Metodología heterogénea para la inspección visual automática basada en técnicas de aprendizaje inexactamente supervisado

Eduardo Villegas Jaramillo

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Industrial
Sede Manizales, Colombia
2024

Metodología heterogénea para la inspección visual automática basada en técnicas de aprendizaje inexactamente supervisado

Eduardo Villegas Jaramillo

Tesis presentada como requisito parcial para optar por el título de:
Doctor en Ingeniería - Industria y Organizaciones

Director:
Prof. Dr. Ing. Mauricio Orozco Alzate

Línea de investigación:
Métodos y modelos de optimización y estadística en Ingeniería Industrial y Administrativa
Grupo de investigación:
Grupo en Ambientes Inteligentes Adaptativos (GAIA)

Universidad Nacional de Colombia
Facultad de Ingeniería y Arquitectura
Departamento de Ingeniería Industrial
2024

Declaración

Me permito afirmar que he realizado esta tesis de manera autónoma y con la única ayuda de los medios permitidos y no diferentes a los mencionados en el presente texto. Todos los pasajes que se han tomado de manera textual o figurativa de textos publicados y no publicados, los he reconocido en el presente trabajo. Ninguna parte del presente trabajo se ha empleado en ningún otro tipo de tesis.

Enero 23 de 2024

Eduardo Villegas Jaramillo

Agradecimientos

En primer lugar, quiero expresar mi agradecimiento por el apoyo, las enseñanzas, el tiempo, la dedicación y la paciencia proporcionados por mi asesor, el profesor Mauricio Orozco Alzate. Sin su ayuda, estoy seguro de que no habría podido completar este trabajo.

Por último, quiero agradecer a mi familia por su paciencia, por estar siempre pendientes, así como por su apoyo y comprensión durante el largo y desafiante proceso de este estudio de doctorado.

Listado de símbolos y abreviaturas

Abreviaturas / Acrónimos

Abre./Acro.	Significado/Término/Traducción
3D-CNN	3 Dimensional Convolutional Neural Network - Red neuronal convolucional en 3 dimensiones
Accu	Accuracy - Exactitud
AITEX	Asociación de Investigación de la Industria Textil
ANN	Artificial Neural Network - Red neuronal artificial
APR	Axis-Parallel Rectangle - Rectángulo paralelo al eje
AUC	Area Under the ROC Curve - Área bajo la curva
BoW	Bag of Words - Bolsa de palabras
CAM	Class Activation Maps - Mapas de activación de clase
CH	Color Histogram - Histograma de color
CNN	Convolutional Neural Network - Red neuronal convolucional
DA	Data Augmentation - Incremento o aumento de datos
DAGM	Deutsche Arbeitsgemeinschaft für Mustererkennung - Sociedad Alemana de Reconocimiento de Patrones
DCNN	Deep Convolutional Neural Network - Red neuronal convolucional profunda
DD	Diverse Density - Densidad diversa
DNN	Deep Neural Network - Red neuronal profunda
DL	Deep Learning - Aprendizaje profundo
ET	Elapsed Time - Tiempo transcurrido
FD	Fourier Descriptors - Descriptores de Fourier
FN	False Negative - Falso negativo
Fast R-CNN	Fast Region-based Convolutional Neural Network Red neuronal convolucional rápida para la detección de regiones
Faster R-CNN	Faster Region-based Convolutional Neural Network Red neuronal convolucional más rápida para la detección de regiones
FP	False Positive - Falso positivo
FP-rate	False Positive rate - Tasa de falsos positivos
GANs	Generative Adversarial Networks - Redes adversarias generativas
GI	Glass Inspector - Inspector de vidrio
GPU	Graphic Process Unit - Unidad de procesamiento gráfico
HoG	Histogram of Oriented Gradient Histograma de gradiente orientado
ICA	Independent Component Analysis - Análisis de componentes independientes
KDE	Kernel Density Estimation - Estimación de la densidad del núcleo
k-NN	K-Nearest Neighbor - K-vecinos más cercanos
LBP	Local Binary Pattern - Patrones binarios locales
LDA	Linear Discriminant Analysis - Análisis discriminante lineal
LSTM	Long Short-Term Memory - Memoria a corto plazo de larga duración

Abre./Acro.	Significado/Término/Traducción
MIL	Multiple Instance Learning - Aprendizaje de instancias múltiples
MILES	Multiple Instance Learning via Embedded Instance Selection - Aprendizaje de múltiples instancias mediante selección de instancias integradas
MIL-KDE	Multi-Instance Kernel Density Estimation - Múltiples instancias con estimación de la densidad basado en el kernel (núcleo)
MI-Boosting	Multiple Instance Boosting - Aprendizaje de múltiples instancias impulsado
MI-DL	Multiple Instance Deep Learning - Aprendizaje profundo de múltiples instancias
MI-NN	Multiple Instance Nearest Neighbor - Vecino más cercano de múltiples instancias
MI-RF	Multiple Instance Random Forest - Bosque aleatorio de múltiples instancias
MI-SVM	Multiple Instance Support Vector Machines - Máquina de vectores de soporte con múltiples instancias
ML	Machine Learning - Aprendizaje de máquina
MLP	Multilayer Perceptron- Perceptrón multicapa
MME	Maximum Mean Estimation - Estimación de la media máxima
MPE	Maximum Probability Estimation - Estimación de probabilidad máxima
NN	Neural Network - Red neuronal
PCA	Principal Component Analysis - Análisis de componentes principales
PR	Pattern Recognition - Reconocimiento de patrones
PU	Process Unit - Unidad de proceso
R-CNN	Region based Convolutional Neural Network - Red neuronal convolucional basada en regiones
RGB	Red Green Blue - Rojo verde azul
RL	Residual Learning - Aprendizaje residual
RNN	Recurrent Neural Network - Red neuronal recurrente
ROC	Receiver Operating Characteristics Graph - Gráfico de características operativas del receptor
R-CNN	Regional Convolutional Neural Network - Red neuronal convolucional de regiones
SD	Standard Deviation - Desviación estándar
SIFT	Scale-Invariant Feature Transform - Transformación de características invariantes de escala
SIVA	Sistema de Inspección Visual Automática
SOM	Self-Organizing Maps - Mapas auto-organizados
SpeedUp	Aceleración
SPP	Spatial Pyramid Pooling - Agrupación piramidal espacial
SSD	Single Shot Multibox Detector - Detector multicaja de disparo único
SSP-net	Spatial Pyramid Pooling Net - Red de agrupación de pirámide espacial
SVM	Support Vector Machine - Máquina de vectores de soporte
TL	Transfer Learning - Aprendizaje por transferencia
TN	True Negative - Verdadero negativo
TP	True Positive - Verdadero positivo
TP-rate	True Positive rate - Tasa de verdaderos positivos
t-SNE	T-Distributed Stochastic Neighbor Embedding - Incrustación estocástica de vecinos distribuida en T
VA	Visión Artificial
ViT	Vision Transformer - Transformador de visión
WSL	Weakly-Supervised Learning Aprendizaje débilmente supervisado
WSOD	Weakly-Supervised Object Detection Detección de objetos con supervisión débil
WSOL	Weakly-Supervised Object Localization Localización de objetos con supervisión débil
YOLO	You Only Look Once - Usted solo lo ve una vez

Uso de acrónimos y términos en inglés

Teniendo en cuenta que muchos acrónimos y términos del idioma inglés son frecuentemente utilizados en el español, y muchos de los cuales no tienen una traducción claramente establecida, o porque en la literatura comúnmente se utilizan en inglés, en esta sección se presenta una lista de aquellos acrónimos y términos del lenguaje inglés que se utilizarán en este documento:

- Algunos términos y medidas de desempeño para aprendizaje de máquina: TP-rate, FP-rate, T, F, TP, FP, TN, FN.
- Términos clásicos en aprendizaje de máquina y de la inteligencia artificial: CNN, RL, TL, DL, DA, WSL, ML, AI.

Resumen

Metodología heterogénea para la inspección visual automática basada en técnicas de aprendizaje inexactamente supervisado

El propósito de la inspección visual automática es detectar y localizar defectos en diferentes tipos de objetos y superficies. Tradicionalmente, estos procesos eran llevados a cabo de manera manual por expertos humanos. Sin embargo, las técnicas de inspección manual suelen ser lentas e ineficientes, encontrando que en muchos casos, no cumplen adecuadamente con las demandas de producción en diversas áreas.

A lo largo del tiempo, se han empleado diferentes soluciones para abordar este problema, centradas principalmente en el procesamiento de imágenes y en técnicas clásicas para la extracción de características, el reconocimiento de patrones, y el uso de clasificadores como máquinas de vectores de soporte, la regla del vecino más cercano y árboles de decisión, entre otros. Recientemente, se ha logrado resolver este problema mediante técnicas de aprendizaje profundo, arrojando resultados muy prometedores. No obstante, se han identificado ciertas limitaciones, tales como la necesidad de contar con un conjunto de datos extenso para el entrenamiento, los elevados requisitos computacionales y la falta de claridad en la interpretación de los resultados.

En esta tesis se explora el empleo de diversas técnicas que incorporan el aprendizaje profundo para abordar problemas de inspección visual automática en la producción de distintos objetos, tales como láminas de vidrio, dulces, telas y conjuntos sintéticos de superficies con textura. Además, ante las limitaciones observadas en las técnicas que hacen uso de aprendizaje profundo, con un enfoque especial en la interpretabilidad, se propone una metodología basada en técnicas de aprendizaje inexactamente supervisado. Esta metodología tiene como objetivo realizar la detección y localización de defectos en diversos problemas de inspección visual automática. La metodología se enfoca en superar y solucionar algunos de los desafíos que surgen al entrenar diferentes modelos cuando no se dispone de información precisa de las etiquetas. Para ello, se integran técnicas provenientes del aprendizaje inexactamente supervisado, como el aprendizaje de múltiples instancias (MIL) y el aprendizaje profundo (DL). Adicionalmente, la utilización de disimilitudes y clasificadores simples, como el del vecino más cercano (k-NN), contribuye al desarrollo y entrenamiento de sistemas capaces de distinguir entre productos defectuosos y no defectuosos, proporcionando la interpretación gráfica correspondiente.

La metodología desarrollada fue evaluada en diversos escenarios con diferentes conjuntos de datos, abarcando tanto conjuntos sintéticos como conjuntos de imágenes reales, mayoritariamente compuestos por superficies texturizadas. Los resultados obtenidos fueron positivos, destacándose varias fortalezas clave de la metodología tales como la capacidad de trabajar con imágenes débilmente etiquetadas, la adaptabilidad para conjuntos de datos con pocas imágenes o desbalanceados, la detección gráfica multiresolución de defectos, la implementación de una ventana deslizante para la generación de bolsas y, finalmente, la habilidad de interpretar de manera gráfica los resultados obtenidos. En cuanto al análisis computacional, es relevante resaltar que las redes neuronales convolucionales (CNN) representan la carga computacional más significativa, ya sea en el entrenamiento del modelo, en la extracción de características o en la predicción de la etiqueta de un objeto de prueba. No obstante, los análisis de desempeño temporal indican que la metodología puede ser aplicada de manera efectiva en diversos contextos.

Palabras clave: inspección visual automática, aprendizaje de múltiples instancias, descomposición en bloques, extracción de características, interpretación gráfica, supervisión inexacta, disimilitudes, localización de defectos

Abstract

Heterogeneous methodology for automatic visual inspection based on inexactly supervised learning techniques

The purpose of automatic visual inspection is to detect and locate defects in different types of objects and surfaces. Traditionally, these processes were carried out manually by human experts. However, manual inspection techniques are usually slow and inefficient, finding that in many cases, they do not adequately meet production demands in various areas.

Over time, different solutions have been used to address this problem, mainly focused on image processing and classical techniques for feature extraction, pattern recognition, and the use of classifiers such as support vector machines, the nearest neighbor rule and decision trees, among others. Recently, this problem has been solved using deep learning techniques, yielding very promising results. However, certain limitations have been identified, such as the need of an extensive dataset for training, high computational requirements, and lack of clarity in the interpretation of results.

This thesis explores the use of various techniques that incorporate deep learning to address automatic visual inspection problems in the production of different objects, such as glass sheets, candies, fabrics, and synthetic sets of textured surfaces. Furthermore, given the limitations observed in techniques that use deep learning, with a special focus on interpretability, a methodology based on inexactly supervised learning techniques is proposed. This methodology aims to detect and localize defects in various automatic visual inspection problems. The methodology focuses on overcoming and solving some of the challenges that arise when training different models when accurate label information is not available. To do this, techniques from weakly supervised learning are integrated, such as multiple instance learning (MIL) and deep learning (DL). Additionally, the use of dissimilarities and simple classifiers, such as the nearest neighbor (k-NN), contributes to the development and training of systems capable of distinguishing between defective and non-defective products, providing the corresponding graphical interpretation.

The developed methodology was evaluated in various scenarios with different datasets, covering both synthetic sets and real image sets, mostly composed of textured surfaces. The results obtained were positive, highlighting several key strengths of the methodology such as the ability to work with weakly labeled images, adaptability for datasets with few or unbalanced images, multi-resolution graphical detection of defects, the implementation of a sliding window for generating bags and, finally, the ability to graphically interpret the results obtained. Regarding computational analysis, it is relevant to highlight that convolutional neural networks (CNN) represent the most significant computational load, whether in model training, feature extraction or in predicting the label of a test object. However, temporal performance analyses indicate that the methodology can be effectively applied in various contexts.

Keywords: automatic visual inspection, multiple instance learning, block decomposition, feature extraction, graphical interpretation, inexact supervision, dissimilarities, defect localization

Lista de figuras

Lista de tablas

Contenido

Parte I

Introducción, antecedentes y propuesta

1 Introducción

En la industria manufacturera, es necesario desarrollar productos que cumplan cada vez más con los requisitos de los consumidores, lo que impulsa la necesidad de contar con sistemas de control de calidad más precisos, más confiables e interpretables. Los procesos productivos demandan, por lo tanto, sistemas automatizados que sean flexibles y permitan mantener controles precisos y eficaces diferentes a los que se presentan en el contexto de la inspección manual tradicional, los cuales son vulnerables a factores como el cansancio, la ansiedad, el estado de ánimo y demás condiciones adversas [Malamas et al.. 2003]. En respuesta a ésta y otras problemáticas, la integración de la Inteligencia Artificial (IA) y el aprendizaje automático (ML) emerge como una solución para mejorar los procesos de inspección visual en las industrias y, por consiguiente, abre oportunidades para una inspección visual automática cada vez más precisa, adaptable y comprensible.

Uno de los principales inconvenientes presentados en los sistemas de inspección visual automáticos (SIVA), es la necesidad de disponer de suficientes imágenes debidamente etiquetadas para entrenar los sistemas de clasificación, donde el proceso de etiquetado de los defectos, por parte de los expertos, se considera una tarea ardua y dispendiosa, siendo en muchos casos casi imposible obtener una buena cantidad de objetos etiquetados a nivel del defecto, contando frecuentemente solo con etiquetas a nivel de la imagen completa, las cuales son más fáciles de conseguir. Otro gran inconveniente se debe a la baja capacidad de clasificación por parte de algunos de los modelos desarrollados, donde varios de los objetos se pueden clasificar de manera errónea, lo que implica que en el proceso de control de calidad artículos defectuosos puedan ser considerados como buenos y viceversa con los costos y consecuencias correspondientes. Adicionalmente, con el fin de verificar la calidad de los resultados, y validar la naturaleza del etiquetado de las imágenes, es importante que los SIVA cuenten con mecanismos de interpretación de los resultados permitiendo así entender y establecer el estado del funcionamiento de los clasificadores y de los métodos utilizados.

Esta tesis se centró en la búsqueda y desarrollo de una metodología heterogénea que —basada en técnicas de aprendizaje inexactamente supervisado (WSL), incluyendo redes neuronales convolucionales (CNN), aprendizaje de múltiples instancias (MIL) y disimilitudes— permitiera mejorar algunos de los aspectos del proceso de clasificación y detección de defectos en problemas enfocados a los SIVA. En concordancia con este propósito, durante el trabajo investigativo se generaron las siguientes publicaciones académicas en las cuales se divulgaron y discutieron los resultados y las contribuciones de esta tesis:

- Ponencia: *Computational Analysis of Multiple Instance Learning-based Systems for Automatic Visual Inspection: A Doctoral Research Proposal''* [Villegas-Jaramillo and Orozco-Alzate. 2019].
- Capítulo de libro: *Convolutional Neural Networks and Deep Learning Techniques for Glass Surface Defect Inspection* [Villegas-Jaramillo and Orozco-Alzate. 2022].
- Ponencia: *Multi-core parallelization of point set dissimilarities for accelerating the comparison of bags with many instances* [Villegas-Jaramillo et al.. 2023].
- Ponencia: *Candy classification using convolutional neural networks, data augmentation and transfer learning: Application and a new public dataset* [Villegas-Jaramillo and Orozco-Alzate. 2023a].

- Arículo: *An Inexactly Supervised Methodology Based on Multiple Instance Learning, Convolutional Neural Networks and Dissimilarities for Interpretable Defect Detection and Localization on Textured Surfaces*, [Villegas-Jaramillo and Orozco-Alzate. 2023b].

Las partes restantes de este documento están organizadas como sigue: en el capítulo 2 se presenta el respectivo planteamiento del problema, las preguntas de investigación, la hipótesis, los objetivos y la justificación del presente trabajo; el marco teórico con su correspondiente revisión de la literatura se presentan en el capítulo 3. En los capítulos 4 y 5, se presenta el desarrollo de soluciones que hacen uso de inspección visual con técnicas de aprendizaje profundo. A continuación, los desarrollos con base en inspección visual utilizando técnicas de aprendizaje inexactamente supervisado se presentan en el capítulo 6. Posteriormente, en el capítulo 7 se presenta la metodología heterogénea que se desarrolló basada en aprendizaje inexactamente supervisado, y finalmente en los capítulos 8 y 9, se presentan las conclusiones y las recomendaciones de trabajos futuros.

2 Planteamiento del problema, objetivos y justificación

2.1 Planteamiento del problema

La inspección visual automática es un aspecto fundamental en una amplia gama de aplicaciones, siendo esencial para la toma de decisiones especialmente en problemas asociados a la industria manufacturera. En los últimos años se han realizado importantes y frecuentes avances en diferentes áreas de la inteligencia artificial (AI), específicamente en las redes neuronales con sus variantes para imágenes basadas en convoluciones, permitiendo así desarrollar mejoras notables en los procesos de inspección visual automática, aunque todavía hay diferentes aspectos donde se puede innovar y mejorar.

Muchos de los problemas encontrados en la industria manufacturera y específicamente en los sistemas de inspección visual, se han resuelto utilizando diferentes técnicas de AI que incluyen: el uso de detectores de bordes, uso de filtros, segmentación de imágenes, extracción de características, clasificadores provenientes del aprendizaje tradicional (máquinas de vectores de soporte (SVM), árboles de decisión y regresión logística), el aprendizaje por transferencia (TL) y el aprendizaje profundo (DL) principalmente. Actualmente, las soluciones basadas en CNN —en sus diferente variantes y en compañía de otras técnicas como DL, TL, aprendizaje residual (RL) y aumento de datos (DA)— son la referencia y el estado del arte para resolver problemas de inspección visual.

Autores como Liu et al. [2022b], han planteado que técnicas basadas en CNN, especialmente el DL a pesar de que tienen una gran capacidad de solucionar problemas enmarcados en el área de inspección visual, presentan algunos inconvenientes especialmente en lo que respecta a la selección e interpretación de los modelos utilizados. Un comentario importante que refuerza lo planteado anteriormente es el de Chollet [2021], donde especifica que *''El aprendizaje profundo no siempre es la herramienta adecuada para el trabajo; a veces no hay suficientes datos para que el aprendizaje profundo sea aplicable y, a veces, el problema se resuelve mejor con un algoritmo diferente''*. Además como lo plantean también Wu and Yan [2023], según los hallazgos por ellos realizados, *''no conocen la existencia de ningún artículo que presente evidencia contundente y reproducible de que el DL supera a los métodos mucho más simples''*. Dado lo anterior, en este trabajo, se busca hacer uso, alternativamente y sin olvidar las CNN, de técnicas provenientes del ML, más simples, más interpretables y que requieran un menor número de ejemplos para el entrenamiento, de tal manera que permitan mejorar aspectos como la eficiencia, la precisión y la interpretación del proceso de clasificación y detección de defectos en problemas enfocados a los SIVA.

En este contexto, el problema que se aborda en este estudio es el de encontrar una metodología que utilice técnicas de aprendizaje inexactamente supervisado para resolver problemas de inspección visual automática. Dicha metodología debe ser capaz de ofrecer buena precisión, utilizar pocas imágenes para el entrenamiento y que sean débilmente etiquetadas, que sea fácil de adaptar a diferentes problemas, así como contar con la

capacidad de interpretación de los resultados.

2.1.1 Preguntas de investigación

- ¿Qué limitaciones y falencias presentan las técnicas para inspección visual automática, basadas en aprendizaje inexactamente supervisado, encontradas en la literatura?
- ¿Qué limitaciones o posibilidades de mejora presentan los algoritmos de inspección visual automática, que implementan técnicas basadas en aprendizaje inexactamente supervisado, encontrados en la literatura?
- ¿Qué combinación, fusión o mezcla de técnicas de aprendizaje, incluyendo aprendizaje multi-instancia y aprendizaje profundo permitiría obtener mejores resultados en los SIVA?
- ¿Qué ganancia —en rendimiento, consumo de recursos, generalización, adaptabilidad e interpretabilidad— se puede obtener con la metodología propuesta?

2.1.2 Hipótesis

La utilización de una metodología heterogénea para la inspección visual automática basada en técnicas de aprendizaje inexactamente supervisado mejora ya sea la eficiencia, la precisión o la interpretación en la detección de defectos en comparación con una metodología homogénea que utiliza técnicas de aprendizaje supervisado.

Además, facilita la interpretabilidad del proceso de inspección al permitir la identificación y comprensión de las características que se utilizan para la detección de los defectos. Asimismo, la metodología heterogénea es más simple que una metodología homogénea al requerir menos datos de entrenamiento y tener una menor dependencia de un experto en la definición de las regiones de interés de los objetos inspeccionados.

2.1.3 Alcance

El trabajo aquí presentado se enmarca específicamente en las etapas de generación de instancias, extracción de características, clasificación, localización e interpretación de defectos en un sistema de inspección visual automática y por lo tanto, sin desconocer su importancia, no se discuten ni se presentan en profundidad los métodos y algoritmos para la adquisición, el preprocesamiento de las imágenes, ni procesos de segmentación propios de éstas y otras etapas de los SIVA.

La metodología desarrollada se evalúa con respecto a diferentes métricas como su exactitud, precisión, valor-F, exhaustividad, matrices de confusión, así como el nivel de aciertos en la localización de defectos y finalmente en términos de la interpretabilidad de los resultados obtenidos. La evaluación se realizó tanto en conjuntos de datos sintéticos como en conjuntos de datos de problemas reales en el contexto de la inspección visual automática; no obstante, la construcción de un prototipo funcional está fuera del alcance de esta propuesta.

2.2 Objetivos

2.2.1 Objetivo general

Proponer una metodología heterogénea para el problema de inspección visual automática, usando técnicas de aprendizaje multi-instancia basado en disimilitudes y en aprendizaje profundo, que incorpore las ventajas individuales de éstas, permitiendo así una detección de defectos más eficiente bajo diferentes medidas y criterios de desempeño.

2.2.2 Objetivos específicos

- Establecer el estado del arte de las técnicas y propuestas algorítmicas más relevantes para la representación y clasificación de objetos, que hagan uso del aprendizaje multi-instancia basado en disimilitudes y técnicas de aprendizaje profundo y que, además, sean las más prometedoras y recomendadas en la literatura reciente para la detección de defectos en la inspección visual automática.
- Evaluar y caracterizar las técnicas, los algoritmos y las implementaciones sobre aprendizaje multi-instancia basado en disimilitudes y en aprendizaje profundo según diversos criterios, con el fin de identificar sus fortalezas, debilidades, limitaciones computacionales y posibilidades de uso para la nueva metodología.
- Identificar estrategias de fusión o combinación de las técnicas y los algoritmos de representación y clasificación seleccionados.
- Determinar las limitaciones técnicas y computacionales de los algoritmos seleccionados para proponer, si es posible, mejoras según diferentes medidas de desempeño.
- Proponer una metodología para sistemas de inspección visual automática, la cual deberá ser evaluada bajo diferentes escenarios de prueba, considerando diversas medidas y criterios de desempeño, tales como eficiencia, precisión, simplicidad, adaptabilidad, interpretabilidad, escalabilidad y otros que se identifiquen durante el desarrollo del proyecto.

2.3 Justificación

Ante la problemática previamente planteada, es importante disponer de metodologías que apoyen los SIVA, de tal manera que puedan reducir los impactos negativos y mejorar así el proceso productivo y, por ende, la calidad de los productos que salen de las líneas de producción. Con relación a este desafío, la investigación que se propone busca realizar un análisis crítico de los algoritmos más representativos de inspección visual automática basados en aprendizaje inexactamente supervisado y, así, disponer de un conjunto de recursos que puedan ser combinados en una solución, a saber: algoritmos, modelos, técnicas de solución, hardware de procesamiento específico y plataforma de desarrollo (lenguajes de programación, librerías), con el fin de encontrar un conjunto de mejoras en el comportamiento de los sistemas de inspección visual.

Con esta investigación se busca entonces contribuir con una mejora en los procesos productivos, y específicamente en los algoritmos de inspección visual automática bajo el aprendizaje inexactamente supervisado desde los siguientes aportes:

- Eficiencia y costo: la metodología propuesta debe tener el potencial de reducir significativamente los costos asociados con la obtención y el etiquetado de grandes conjuntos de datos de entrenamiento, ya que solo requiere etiquetas de clase asignadas al nivel de la imagen completa del objeto y no necesita una gran cantidad de las mismas, lo que hace que la inspección visual automática sea más accesible para una variedad de industrias.
- Precisión y calidad: a pesar de las dificultades inherentes en el aprendizaje inexactamente supervisado, se busca mostrar que esta metodología puede lograr niveles de precisión comparables a los enfoques convencionales en una variedad de situaciones de inspección visual.
- Conjuntos de pruebas: se presentarán diferentes problemas de inspección visual abordados con los respectivos resultados que se alcancen según los enfoques y configuraciones encontradas.
- Interpretabilidad: se presentará una metodología que, con base en un conjunto de algoritmos y métodos, permita dar solución a los requerimientos en cuanto a interpretación de resultados de los SIVA.
- Adaptabilidad: se busca que la metodología desarrollada se pueda adaptar con facilidad para ser utilizada con otros conjuntos de datos.

En resumen, esta tesis se justifica por la necesidad de abordar algunas de las limitaciones de los enfoques tradicionales de inspección visual automática mediante la exploración y desarrollo de una metodología basada en técnicas de aprendizaje inexactamente supervisado. El objetivo final es contribuir con mejoras ya sean en la eficiencia, desempeño y adaptabilidad de los sistemas de inspección visual, lo que puede tener un impacto significativo en la calidad de la producción industrial en una variedad de sectores.

3 Marco teórico/conceptual y revisión de literatura

3.1 Introducción

En este capítulo se introducen los conceptos y referentes teóricos necesarios para el desarrollo de esta investigación sobre una metodología heterogénea para la inspección visual automática basada en técnicas de aprendizaje inexactamente supervisado, comenzando por definir los SIVA, mostrando los diferentes mecanismos para la evaluación de los modelos que allí se utilizan, así como las diferentes técnicas utilizadas en el aprendizaje de máquina, haciendo énfasis en MIL y CNN donde se incluye una sección con las ventajas y desventajas de dichas técnicas, especialmente enfocadas en el desarrollo de SIVA. Adicionalmente, se presentan los esquemas de representación que pueden ser utilizados en un sistema de esta naturaleza, continuando con los métodos utilizados para la clasificación de imágenes y localización de defectos, donde se desarrolla un cuadro a manera de comparación de los métodos de clasificación/localización que permite seleccionar aquellos más prometedores. Posteriormente, se presenta la interpretación e interpretabilidad de los modelos basados en CNN como un elemento de gran relevancia e importancia para el desarrollo del presente trabajo y, finalmente la complejidad algorítmica de los modelos CNN como una herramienta para analizar la viabilidad para implantar las soluciones alcanzadas, seguida de la revisión sistemática de la literatura, donde es posible identificar y recopilar toda la investigación relevante previa en el área específica de los SIVA y así facilitar la comparación de diferentes enfoques y metodologías utilizadas en su implementación.

3.2 Sistemas de inspección visual y estimación de desempeño

Este trabajo de investigación se centra en los SIVA, por ser éstos algunos de los escenarios naturales y típicos en donde se presenta la problemática de reconocer, clasificar, detectar y localizar diferentes elementos en conjuntos de imágenes. Estos escenarios permiten experimentar y aplicar diferentes estrategias y metodologías para su solución.

En el reconocimiento automatizado es importante hacer claridad entre los siguientes conceptos: clasificación, detección, localización, interpretación e interpretabilidad. La *clasificación* es la tarea de asignar una etiqueta de clase; es decir, responder a la pregunta *''¿QUÉ es esto?''*; en la inspección visual automática, esta pregunta puede formularse como: *''¿Es este objeto defectuoso o bueno/funcional?''* —en el caso de dos clases, la tarea se conoce como *detección* [Lei et al.. 2017, p. 4]—. Además, en el caso de varias clases, la clasificación es responder a la pregunta *''¿QUÉ tipo de defecto es éste?''*. La *localización* es la tarea de encontrar la parte concreta del objeto que motiva la asignación de la etiqueta de clase; es decir, para la inspección visual automática, corresponde a responder a la pregunta *''¿DÓNDE está el defecto?''*. Por otro

lado, la *interpretación* corresponde a la tarea de explicar el motivo de la asignación de la etiqueta de clase; es decir, responder a la pregunta *"¿POR QUÉ esa etiqueta de clase se asignó al objeto?"*. Por último, la *interpretabilidad* es la tarea de determinar en qué grado el resultado es predecible o comprensible para los seres humanos [Li et al.. 2021b]; es decir, responder a la pregunta *"¿CÓMO el sistema llegó a asignar esa etiqueta de clase?"*.

3.2.1 Introducción a la inspección visual

La inspección visual es una técnica para detectar defectos a simple vista con el fin de asegurar que los productos se están fabricando de acuerdo con una determinada especificación. Se centra en la detección y localización de defectos en diferentes tipos de objetos o elementos, se realizan tradicionalmente mediante procedimientos manuales desarrollados por expertos humanos, las cuales normalmente son procesos lentos e ineficientes, y muchas veces no permiten cumplir con las necesidades de producción en diferentes áreas [Li and Wang. 2021].

El uso de técnicas para la inspección visual automática en los procesos de fabricación, y especialmente para la búsqueda de defectos en superficies, es cada día más frecuente, conformando lo que se denomina SIVA. Tal como se puede apreciar en la Figura **3-1**, los SIVA se componen de cuatro etapas: (i) la *adquisición* de imágenes digitales de los objetos; (ii) el *pre-procesamiento* de la imagen, mediante la aplicación de filtros y algoritmos para eliminar las distorsiones que obstruyen el reconocimiento de la imagen, por ejemplo, ruido, reflejos de luz y manchas; (iii) la *representación* de la imagen preprocesada, normalmente mediante la extracción de características visuales y/o estructurales, cuyos valores se almacenan en un vector de características también llamado *instancia* y; (iv) la etapa de *clasificación*, en la que un algoritmo —basado en ejemplos proporcionados por un experto— asigna una etiqueta de clase a la representación del objeto; por ejemplo: defectuoso o no defectuoso [Malamas et al.. 2003].

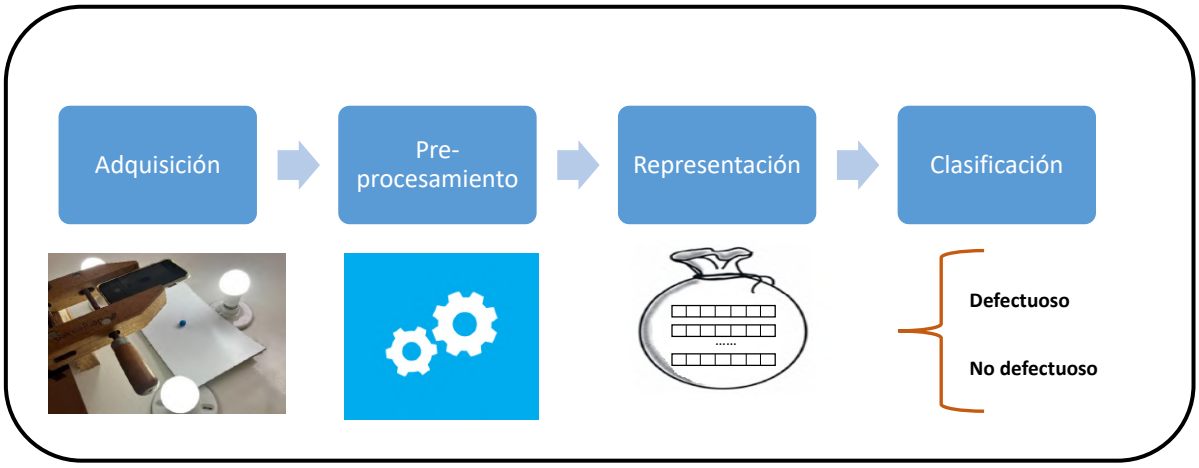


Figura 3-1: Etapas típicas de un SIVA.

Los SIVA permiten controlar la calidad de todo tipo de productos industriales haciendo uso de técnicas relacionadas, entre otras, con la visión y la AI. Para resolver este problema, se utilizan diversas estrategias clásicas, como los métodos de proyección [Chen and Perng. 2011], los enfoques basados en filtros [Lin and Ho. 2007] y los enfoques basados en aprendizaje automático estándar [Chen and Hsu. 2007]. La mayoría de estas estrategias asumen la certeza en las etiquetas de clase asignadas por los expertos a los ejemplos de entrenamiento; sin embargo, la debilidad o ambigüedad de las etiquetas [Zhou. 2017] puede aparecer en

algunos conjuntos de datos que requieren la aplicación de enfoques no estándar [Mera et al.. 2016]. Otros enfoques utilizan redes neuronales, donde el DL emerge como una buena alternativa cuando se aplica en sus diferentes variantes como las CNN, las redes neuronales profundas (DNN) y las redes basadas en RL [Zhou et al.. 2019], que pueden complementarse utilizando otras técnicas como DA [Shorten and Khoshgoftaar. 2019] y el TL [Hafemann et al.. 2015], obteniendo buenos resultados especialmente en cuanto al problema de detección de defectos. Con respecto al problema de localización de defectos, éste ha venido resolviéndose mediante el uso de técnicas más recientes como las CNN basadas en regiones, entre las que se encuentran las compuestas por dos etapas: R-CNN [Girshick et al.. 2014], Fast R-CNN [Girshick. 2015], Faster R-CNN [Ren et al.. 2017] y Mask R-CNN [He et al.. 2017]; por otro lado, los detectores de una sola etapa, como *''You Only Look Once''* (YOLO) [Redmon et al.. 2016] y *''Single Shot Detector''* (SSD) [Liu et al.. 2016], los cuales se caracterizan por su alta velocidad de respuesta.

3.2.2 Evaluación y validación

Dado que es imperativo realizar la evaluación y validación de los diferentes modelos que se desarrollen como resultado de esta tesis, se presentan las diferentes técnicas, métodos y procedimientos que se utilizan para evaluar y validar los SIVA, y en especial el rendimiento de los clasificadores, entre los cuales vale la pena destacar: la clase positiva u objetivo (P), la clase negativa (N), las medidas de exactitud, precisión, exhaustividad, valor-F, la tasa de verdaderos positivos, la tasa de falsos positivos, la matriz de confusión, el reporte de clasificación, la validación cruzada y, las curvas ROC.

La matriz de confusión

La matriz de confusión, es un arreglo de dos dimensiones que permite determinar, como su nombre lo indica, qué tan *''confundido''* está un modelo al momento de realizar el proceso de clasificación, mostrando tanto la cantidad de aciertos como desaciertos cometidos para cada una de las diferentes categorías. La matriz de confusión describe cómo se distribuyen los valores reales y las predicciones que realiza el sistema, como se muestra a continuación:

		Clase predichas		Total
		+	-	
Clase real	+	TP	FN	TP+FN
	-	FP	TN	FP+TN

Un caso especial de las matrices de confusión, según Bramer [2020, Chap. 12] es el clasificador perfecto; es decir, aquél que clasifica bien en todas las categorías, así:

		Clase predichas		Total
		+	-	
Clase real	+	P	0	P
	-	0	N	N

Exactitud

Es la medida más frecuentemente utilizada para estimar el desempeño de un clasificador. Mide qué tan cerca está el resultado de una medición respecto al valor verdadero, indicando cuál es la proporción de instancias que están correctamente clasificadas. Se calcula según la proporción entre el número de predicciones correctas

(tanto positivas como negativas) y el total de predicciones:

$$\text{Exactitud} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{3-1}$$

Precisión

Describe cuántos elementos detectados son realmente relevantes. Se calcula según la proporción entre los positivos reales predichos por el algoritmo y todos los casos positivos:

$$\text{Precisión} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{3-2}$$

Valor-F

Busca combinar las medidas de precisión y exhaustividad en un sólo valor, lo que hace más fácil poder comparar el rendimiento combinado de la precisión y la exhaustividad entre varias soluciones. Se representa como la proporción entre el producto de la precisión por la exhaustividad, y la suma de la precisión y la exhaustividad, multiplicado por 2:

$$\text{Valor-F} = 2 * \frac{\text{Precisión*exhaustividad}}{\text{Precisión+exhaustividad}} \tag{3-3}$$

Especificidad

También conocida como la Tasa de Verdaderos Negativos, o TN.rate. Mide la proporción de verdaderos negativos que el modelo identifica correctamente. Expresa qué tan bien el modelo puede detectar esa clase. Se representa como la proporción entre los verdaderos negativos y la suma de los verdaderos negativos más los falsos positivos:

$$\text{Especificidad} = \frac{\text{TN}}{\text{TN} + \text{FP}} \tag{3-4}$$

Tasa de falsos positivos (FP-rate)

Determina cual es la proporción de casos negativos que se clasifican erróneamente como positivos, y tiene como ventaja que no depende del tamaño relativo de N. Se representa como la proporción entre los falsos positivos y la suma de los verdaderos negativos más los falsos positivos:

$$\text{FP-rate} = \frac{\text{FP}}{\text{TN} + \text{FP}} \tag{3-5}$$

Exhaustividad o tasa de verdaderos positivos (TP-rate)

También conocida como *''Sensibilidad''* o *''Recuperación''*. Se refiere a con qué frecuencia un modelo basado en aprendizaje de máquina es correcto al predecir la clase objetivo. Se representa por la proporción entre los

verdaderos positivos predichos por el algoritmo y la suma de los verdaderos positivos más los falsos negativos:

$$\text{Exhaustividad} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3-6}$$

Reporte de clasificación

Es un resumen global de la evaluación del desempeño de un modelo de aprendizaje automático basado en clasificación. Muestra la precisión, la exhaustividad, el valor-F y los datos de soporte del modelo para cada una de las clases. Proporciona una mejor comprensión del rendimiento general del modelo entrenado[1].

Curva ROC

Se le conoce como curva característica operativa del receptor, o curva ROC. Es un gráfico que ilustra la capacidad de diagnóstico de un sistema clasificador binario a medida que varía su umbral de discriminación. La curva ROC es una representación gráfica del rendimiento del clasificador donde se muestra la distribución de las fracciones de verdaderos positivos y de falsos positivos.

Existe una variación de las curvas ROC propuesta por Bramer [2020, Chap. 12] ubicando cada clasificador como un punto en el plano TP-rate vs. FP-rate y, a continuación, se calcula la distancia de cada uno con respecto a un clasificador perfecto; es decir, respecto a un clasificador con TP-rate = 1.0 y FP-rate = 0.0. El mejor clasificador es aquél con la menor distancia al clasificador perfecto.

Validación cruzada

La validación cruzada es un método estadístico para evaluar y comparar algoritmos de aprendizaje dividiendo los datos en dos conjuntos: el primero utilizado para entrenar el modelo y el segundo utilizado para su validación. En una validación cruzada típica, los conjuntos de entrenamiento y validación deben cruzarse en rondas sucesivas de modo que cada valor entre los datos tenga la posibilidad de ser validado [Refaeilzadeh et al.. 2009]. Se utiliza la validación cruzada para detectar el sobreajuste, es decir, en aquellos casos en los que el modelo no se capaz de generalizar un determinado patrón.

Las principales técnicas de validación cruzada son[2]:

- Dividir en entrenamiento-prueba (Train-Test Split) es una técnica que consiste en dividir de manera aleatoria un conjunto de datos. Una parte de los datos servirá para el entrenamiento del modelo, la otra parte permitirá probarlo para la validación. Por lo general, se reserva entre un 70 % y 80 % de los datos para el entrenamiento, y el 20-30 % restante se usa para la prueba de la validación cruzada.
- Validación cruzada K veces (K-Folds) es una técnica que permite garantizar que todas las observaciones o muestras de la serie de datos original tengan la oportunidad de aparecer en el conjunto de entrenamiento y en el conjunto de prueba. Es muy útil cuando los datos de entrada disponibles son limitados.
- Validación cruzada aleatoria, es una técnica en la cual se divide aleatoriamente el conjunto de datos en entrenamiento y prueba.

[1] https://thecleverprogrammer.com/2021/07/07/classification-report-in-machine-learning/
[2] https://datascientest.com/es/cross-validation-definicion-e-importancia

- Validación cruzada dejando uno afuera, conocida como *''leave one out''*, es una técnica donde se separan los datos de tal forma que para cada iteración se selecciona una sola muestra como dato de prueba, conformando todos los demás el conjunto de entrenamiento.

3.3 Aprendizaje de máquina

Partiendo de que el aprendizaje de máquina es uno de los marcos en que se desarrolla este trabajo, éste se define como el estudio de los diferentes algoritmos computacionales que tienen la capacidad de aprender automáticamente de los datos y de las experiencias previas, con el fin de encontrar patrones y hacer predicciones sin la participación humana. El ML y sus aplicaciones en diferentes áreas de estudio son un componente fundamental de la AI [Soori et al.. 2023] y de gran aplicabilidad en los SIVA.

Algunos autores como Wang et al. [2009], plantean que el ML es un campo interdisciplinario de la informática que se enfoca en el desarrollo de algoritmos y modelos que permiten a los computadores aprender de los datos, de tal manera que sea posible simular las actividades del aprendizaje humano, estudiar los métodos de auto-mejora para obtener nuevos conocimientos y nuevas habilidades, identificar los conocimientos existentes, mejorar su desempeño en tareas específicas y tomar decisiones inteligentes sin la intervención humana explícita.

Entre los principales enfoques que hacen uso del ML están: el aprendizaje supervisado, el aprendizaje no supervisado y el aprendizaje débilmente supervisado, los cuales se especifican a continuación siendo el aprendizaje débilmente supervisado la base para el desarrollo de este trabajo.

3.3.1 Aprendizaje supervisado

El aprendizaje supervisado es un tipo de ML en el que se entrena un modelo utilizando datos etiquetados; es decir, datos que ya han sido previamente categorizados o clasificados. Esto implica que el modelo recibe ejemplos de entrada (imágenes, textos o datos numéricos) junto con las etiquetas correspondientes que indican la categoría, respuesta o clase a la que pertenece cada ejemplo. A partir de estos datos de entrenamiento, el modelo aprende a hacer predicciones para nuevos ejemplos de entrada [Sun et al.. 2023]. Algunos de los algoritmos de aprendizaje supervisado más populares son los árboles de decisión, las redes neuronales, las máquinas de vectores de soporte (SVM) y los algoritmos de regresión lineal y logística [Goodfellow et al.. 2016].

3.3.2 Aprendizaje no supervisado

El aprendizaje no supervisado es una técnica de aprendizaje automático en la que se utilizan algoritmos para analizar conjuntos de datos sin etiquetar y descubrir patrones o agrupaciones de datos sin la necesidad de intervención humana. A diferencia del aprendizaje supervisado, en el que los datos están etiquetados y se espera que el modelo aprenda a predecir la variable dependiente, en el aprendizaje no supervisado los datos no tienen etiquetas y el modelo debe encontrar patrones y estructuras por su propia cuenta [Zipfel et al.. 2023]. Algunas técnicas comunes de aprendizaje no supervisado incluyen el análisis de componentes principales (PCA), la agrupación o *''clustering''*, la reducción de dimensionalidad y las redes neuronales auto-organizativas.

3.3.3 Aprendizaje débilmente supervisado

El WSL es una técnica de ML la cual utiliza una cantidad limitada de información sobre las etiquetas o anotaciones necesarias para entrenar un modelo, de tal manera que se pueda generalizar a datos no etiquetados, permitiendo así, utilizar fuentes de información que son más fáciles de obtener que los datos previamente etiquetados, donde la información o supervisión puede ser incompleta, imprecisa o inexacta [Zhou. 2017].

Supervisión incompleta (aprendizaje activo, aprendizaje semi-supervisado)

La supervisión incompleta se refiere a la situación en la que solo se cuenta con una pequeña cantidad de datos etiquetados, que es insuficiente para realizar un adecuado proceso de entrenamiento, mientras que sí se dispone de abundantes datos sin etiquetar [Zhou. 2017]. Existen tres técnicas principales para resolver el problema de la falta de datos etiquetados como son: el aprendizaje activo, el aprendizaje semi-supervisado y la transferencia de aprendizaje. i) El aprendizaje activo parte del supuesto de que existe un *''oráculo''* como si fuera un experto humano, al que se puede consultar para obtener etiquetas de verdad para las instancias sin etiquetar, además de los datos etiquetados, con el fin de mejorar el rendimiento del aprendizaje, en el que se supone que no hay intervención humana. ii) En el aprendizaje semi-supervisado se emplean unos pocos datos etiquetados y muchos datos no etiquetados como parte del conjunto de entrenamiento. Estos algoritmos tratan de explorar la información estructural de los datos no etiquetados buscando generar modelos predictivos que tengan un mejor comportamiento que aquellos que sólo utilizan datos etiquetados. iii) Finalmente, en el aprendizaje por transferencia, se utilizan modelos previamente entrenados en otro conjunto de datos y se aplican a los datos y la tarea que es de interés. Los datos y su forma serían similares, y se puede afinar el modelo para una tarea específica. Esto también puede implicar la combinación de varias tareas para comprender relaciones de un dominio aproximado que podrían no ser obvias[3].

Supervisión imprecisa

La supervisión imprecisa se refiere a la situación en la cual la información de supervisión no siempre es veraz; en otras palabras, como su nombre indica, en la supervisión imprecisa la información puede contener errores, donde algunas de las etiquetas de verdad no son precisas o no son de buena calidad. Por lo general, esto ocurre cuando hay imprecisiones en el etiquetado, o cuando los datos son difíciles de categorizar [Zhou. 2017].

Un escenario frecuente de supervisión inexacta ocurre con el *''crowdsourcing''*, un paradigma popular para subcontratar el trabajo a individuos, el cual se utiliza para asignar etiquetas a los datos de entrenamiento. Específicamente, las instancias sin etiquetar se subcontratan a un grupo de trabajadores, con o sin experiencia, para que las etiqueten, donde no se tiene una garantía de la fiabilidad y/o precisión de las mismas [Bafti et al.. 2021].

Supervisión inexacta (aprendizaje multi-instancia)

La supervisión inexacta se refiere a la situación en la que se presenta alguna información de supervisión, pero no es tan exacta como se desea. Específicamente, se presenta en escenarios donde se brinda información sobre la presencia de un defecto, pero no se especifica la ubicación del mismo [Zhou. 2017].

La supervisión inexacta puede clasificarse en: supervisión a nivel de imagen y supervisión a nivel de

[3] https://www.oreilly.com/library/view/practical-weak-supervision/9781492077053/ch01.html

caja delimitadora. En la supervisión a nivel de imagen sólo se proporcionan etiquetas de las imágenes de entrenamiento; en contraste, a nivel de caja, además de las etiquetas de la categoría o clase, también se proporcionan las cajas delimitadoras de los objetos para cada imagen de entrenamiento [Shen et al.. 2023].

La supervisión inexacta se centra principalmente en MIL tal como lo presentan en Zhou [2017]; Yue et al. [2022]; Cheplygina et al. [2015], y dado que el aprendizaje inexactamente supervisado y específicamente el aprendizaje de múltiples instancias es el marco del desarrollo de este trabajo, éste se presenta con más detalle en una sección aparte a continuación.

3.4 Aprendizaje de múltiples instancias

El aprendizaje de múltiples instancias (MIL), otro de los referentes para el desarrollo de este trabajo, se considera como un enfoque de aprendizaje automático que se ha utilizado ampliamente en los últimos años, el cual se utiliza en tareas en las que no se dispone de etiquetas precisas para cada instancia individual, sino más bien etiquetas para conjuntos o *''bolsas''* de instancias [Herrera et al.. 2016]. MIL se ha aplicado en una gran variedad de dominios y problemas, como la clasificación de imágenes médicas, la detección de objetos en imágenes, en la predicción de la actividad de fármacos, la clasificación de documentos, entre otros.

Sea $\mathcal{X}$ una bolsa definida como un conjunto de vectores de características $\mathcal{X} = \{X_1, X_2, \ldots, X_i, \ldots X_N\}$, cada instancia (es decir, vector de características) X_i en el espacio de características. $\mathcal{X}$ puede asignarse a una clase mediante un proceso $f(X) \rightarrow \{0, 1\}$, donde las clases negativa y positiva corresponden a 0 y 1 respectivamente [Yue et al.. 2022]. El clasificador de bolsa $g(X)$ se define como:

$$g(\mathcal{X}) = \begin{cases} 1, & \text{si } \exists x \ \in \mathcal{X} : f(x) = 1 \\ 0, & \text{en caso contrario} \end{cases} \tag{3-7}$$

La suposición en el estándar de MIL establece que todas las bolsas negativas contienen sólo instancias negativas, y que las bolsas positivas contienen al menos una instancia positiva [Carbonneau et al.. 2017].

Para efectos de este trabajo, en MIL se pueden diferenciar dos enfoques: MIL clásico o basado en vectores y MIL basado en disimilitudes:

3.4.1 MIL clásico

En el enfoque clásico o estándar de MIL, cada bolsa contiene múltiples vectores de características o instancias. En este escenario, cada bolsa tiene una etiqueta asociada, pero no se conocen las etiquetas de las instancias individuales que conforman la bolsa. Además, otro punto importante, es que no todas las instancias son necesariamente relevantes, es decir, puede haber instancias dentro de una bolsa que no aportan ninguna información sobre su clase, o que están más relacionadas con otras clases, proporcionando información confusa [Amores. 2013]. Para representar imágenes bajo este enfoque, se pueden extraer regiones de interés con sus características y cada región forma un vector de características de la bolsa como se puede ver en la Figura **3-2**.

La suposición estándar para MIL es que las etiquetas de instancia se relacionan con las etiquetas de la bolsa de la siguiente manera: una bolsa es positiva si y sólo si contiene al menos una instancia positiva, de lo contrario la bolsa es negativa [Cheplygina et al.. 2015]. A partir de dicha representación, el objetivo de los

Figura 3-2: Representación de las características de una imagen en una bolsa bajo el enfoque estándar de MIL.

clasificadores basados en MIL es entrenar un modelo que pueda clasificar correctamente las bolsas según las categorías predefinidas, en las siguientes etapas:

Representación: cada objeto se representa por una bolsa, la cual está conformada por uno o varios vectores de características que reúnen la información de la imagen o del objeto a representar. Estos vectores de características pueden ser generados de diversas formas, ya sea mediante la extracción de características, los valores de los píxeles de las imágenes, o como el promedio, suma, máximo, mínimo, de algunos de sus atributos.

Clasificación: El objetivo principal es entrenar un modelo de clasificación que pueda predecir la etiqueta de una bolsa completa en lugar de etiquetar cada instancia de forma individual. El modelo se entrena utilizando un conjunto de bolsas etiquetadas y se ajusta para predecir la categoría de nuevas bolsas. Algunos de los clasificadores que se pueden utilizar bajo este enfoque son:

- Rectángulos paralelos al eje (APR): mediante esta técnica se definen rectángulos (también llamados regiones) en el espacio de características de los datos. Estos rectángulos se utilizan para dividir las instancias y determinar si una bolsa se etiqueta como positiva o negativa. Cada rectángulo se define por un conjunto de características y un valor umbral para esas características, donde el algoritmo busca encontrar los rectángulos óptimos que permiten separar las bolsas positivas de las negativas en el espacio de características [Dietterich et al.. 1997].
- Densidad diversa (DD): busca encontrar áreas en un espacio de características que estén al menos cerca de una instancia positiva y lejos de cada instancia negativa. El algoritmo busca en el espacio de características puntos con alta densidad diversa. Una vez que el punto o los puntos con máxima densidad diversa son encontrados, una nueva imagen se clasifica positiva, si una de sus sub-imágenes esta cerca al punto de máxima densidad diversa [Maron and Ratan. 1998].
- MIL ingenuo (Naive-MIL): el modelo se entrena para clasificar bolsas en función de si al menos una instancia dentro de la bolsa es positiva. Puede ser un modelo de clasificación binaria, como regresión logística o un clasificador basado en árboles de decisión. Una vez que el modelo está entrenado, se utiliza para clasificar nuevas bolsas, donde el modelo calcula si al menos una instancia en la bolsa es positiva o si todas son negativas [Babenko. 2008].
- MILES: utiliza un enfoque de selección de instancias para mejorar la precisión de la clasificación. Para hacerlo, se calcula una puntuación de relevancia para cada instancia en cada bolsa. Esta puntuación de relevancia mide cuánto contribuye cada instancia a la predicción de la etiqueta de la bolsa. El método selecciona las instancias más informativas de cada bolsa, lo que permite construir un clasificador basado en esas instancias seleccionadas. Para clasificar nuevas bolsas, MILES calcula las puntuaciones de relevancia de las instancias en la bolsa desconocida y luego aplica el modelo entrenado para predecir si la bolsa es positiva o negativa [Cheplygina et al.. 2015].

- MI-Boosting: conecta las predicciones para las instancias individuales con una estimación de probabilidad a nivel de bolsa promediando y maximizando la probabilidad a nivel de bolsa; asumiendo que todas las instancias contribuyen de manera igual e independiente a la etiqueta de una bolsa [Xu and Frank. 2004].

- MI-KNN: es una adaptación del algoritmo de vecinos más cercanos k-NN para el aprendizaje de múltiples instancias. Calcula la distancia entre la bolsa a clasificar y todas las bolsas de entrenamiento en el espacio de características. Las distancias pueden calcularse de varias maneras, como la distancia euclidiana, Hausdorff, Manhattan, Mahalanobis entre otras; luego, se seleccionan las K bolsas de entrenamiento más cercanas a la bolsa desconocida según las distancias calculadas. Una vez que se han seleccionado los K vecinos más cercanos, se realiza una votación de mayoría para determinar la etiqueta de la bolsa desconocida [Zafra and Gibaja. 2023].

- Aprendizaje profundo de múltiples instancias (MI-DL): Una extensión del DL. En este caso, una bolsa con múltiples instancias se modela por su disimilitud con otras bolsas, y el cálculo de las disimilitudes se lleva a cabo en una nueva red neuronal, denominada red de similitud o disimilitud de bolsas. Las cuales pueden incluir arquitecturas de CNN y RNN adaptadas para MIL [Wang et al.. 2019b].

- Máquina de vectores de soporte con múltiples instancias (MI-SVM): es una extensión de las SVM para problemas de aprendizaje de múltiples instancias. Se busca encontrar un hiperplano separador en un espacio de características que pueda discriminar entre bolsas positivas y negativas. El objetivo es maximizar la distancia entre el hiperplano y las bolsas positivas más cercanas, mientras se minimiza la distancia con respecto a las bolsas negativas más cercanas [Andrews et al.. 2002].

- Bosque aleatorio de múltiples instancias (MI-RF): es una extensión de los bosques aleatorios para problemas de aprendizaje de múltiples instancias. Combina un conjunto de árboles de decisión con la flexibilidad del aprendizaje de múltiples instancias, donde se definen las etiquetas de clases ocultas dentro de las bolsas de destino como variables aleatorias. Estas variables aleatorias se optimizan entrenando bosques aleatorios, para clasificar las bolsas basándose en las instancias contenidas en ellas [Leistner et al.. 2010].

- MIL con cuadros delimitadores flexibles: es una formulación MIL basada en una transformación polar que funciona tanto para cuadros delimitadores ajustados como para cuadros delimitadores flexibles, en los que una bolsa positiva se define como píxeles en una línea polar de un cuadro delimitador con un punto final ubicado dentro del objeto encerrado por el cuadro y el otro punto final ubicado en uno de los cuatro lados de la caja delimitadora. Además, se introduce una aproximación máxima ponderada para incorporar la observación de que es más probable que los píxeles más cercanos al origen de la transformación polar pertenezcan al objeto delimitado por la caja [Wang and Xia. 2024].

3.4.2 MIL con disimilitudes

El enfoque de MIL con disimilitudes es una extensión del enfoque clásico que incorpora el cálculo de medidas de disimilitud entre las instancias en una bolsa. Busca representar cada bolsa mediante un vector de sus disimilitudes con las demás bolsas en el conjunto de entrenamiento (ver Figura **3-3**), y tratar estas disimilitudes como una representación de características [Cheplygina et al.. 2015]. En el enfoque de MIL con disimilitudes, se deben seguir las siguientes etapas [Huang et al.. 2022]:

Cálculo de disimilitudes: se calculan las disimilitudes entre las instancias en cada bolsa. Estas medidas pueden incluir distancia euclidiana, distancia de Mahalanobis, distancia de Hausdorff, coseno o cualquier otra métrica de disimilitud adecuada para el problema.

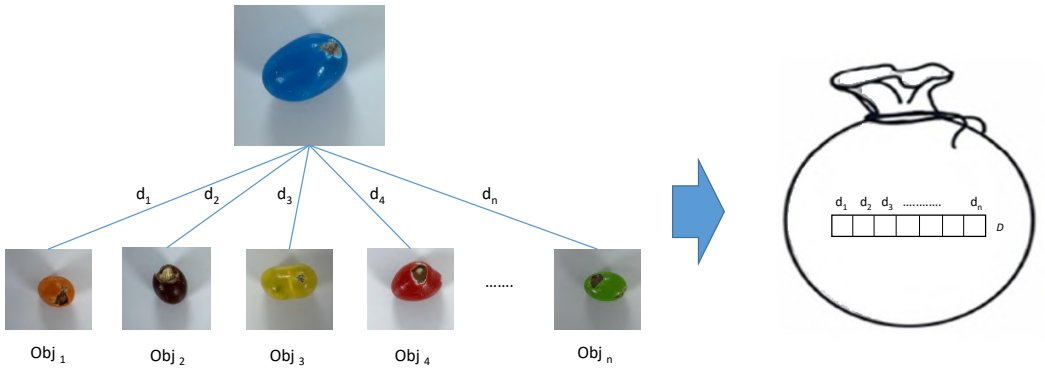


Figura 3-3: Representación de las características de una imagen en una bolsa bajo el enfoque MIL con disimilitudes, en el cual las disimilitudes del objeto con respecto a los demás se almacenan en un vector.

Representación de las bolsas: en lugar de representar las bolsas solo con vectores de características, se incorporan las medidas de disimilitud en la representación. Cada bolsa se describe mediante una matriz de disimilitudes que almacena las relaciones entre las diferentes instancias.

Clasificación de las bolsas: el modelo de clasificación se entrena para aprovechar los vectores y/o las matrices de disimilitudes en la clasificación. Estas matrices proporcionan información adicional sobre las relaciones entre las instancias dentro de una bolsa y se utilizan para mejorar la precisión de la clasificación. En MIL con disimilitudes, podrían usarse los mismos clasificadores que en MIL clásico (ver 3.4.1) puesto que el espacio de disimilitudes es, en sí mismo, un espacio vectorial convencional.

El enfoque clásico y el enfoque basado en disimilitudes tienen aplicaciones en una variedad de problemas en los que las etiquetas se aplican a nivel de bolsa o donde las relaciones entre las instancias en una bolsa son importantes. La elección entre el enfoque clásico y el enfoque con disimilitudes depende de la naturaleza del problema y de la disponibilidad de información adicional sobre las relaciones entre las instancias en las bolsas.

3.4.3 Ventajas y desventajas de MIL

Con respecto a las ventajas y desventajas que presenta MIL es importante tener en cuenta como lo plantea Łukasz Struski et al. [2024], que si bien los modelos basados en instancias pueden presentar un rendimiento en cuanto a resultados ligeramente inferior en comparación con métodos más generales, ofrecen como gran ventaja la interpretabilidad, lo que permite rastrear o seguir el proceso de toma de decisiones hasta cada instancia individual. Otro factor importante es que MIL funciona con imágenes débilmente etiquetadas lo que supone un menor esfuerzo por parte del experto para su etiquetado, así como no necesita una gran cantidad de imágenes para entrenar los respectivos sistemas [Zhou. 2017].

Como una desventaja es necesario analizar que la representación que plantea MIL puede conducir a problemas computacionalmente costosos debido a la demanda computacional resultante si el número de instancias generadas por cada bolsa es grande. Este problema se puede enfrentar mediante estrategias de paralelización

como la desarrollada en el Capítulo 6 de este documento de tesis, donde se aborda el problema de MIL con disimilitudes con buenos resultados, así como la propuesta presentada por Xu et al. [2017] donde se desarrolló un algoritmo de aprendizaje paralelo de instancias múltiples (P-MIL) implementado sobre clústeres de computación de alto rendimiento (HPC) reduciendo significativamente los tiempos de ejecución.

Otro aspecto importante a tener en cuenta es que si el problema a resolver incluye máscaras o información sobre la delimitación de los defectos, MIL pierde su ventaja competitiva frente a las redes neuronales ya que, al contar con máscaras, se tiene la capacidad de conocer exactamente la ubicación de los defectos bajo una estrategia completamente supervisada, para lo cual, en muchos conjuntos de datos públicos ya se están proporcionando las imágenes y sus correspondientes máscaras o anotaciones de delimitación de defecto, tal como se aprecia en algunos conjuntos de datos como: Kolektor [4], MVTEC [5], NEU [6], entre otros.

En contraste, las CNN presentan como gran ventaja con respecto a MIL, la posibilidad de capturar relaciones espaciales entre los píxeles de una imagen, lo que les permite comprender mejor el contexto de los defectos dentro de ella, facilitando así el proceso de detección y localización de defectos.

Finalmente, es importante resaltar trabajos como el presentado por Sun et al. [2016], donde se hace una comparación de las técnicas basadas en MIL y en CNN, encontrando que cada una tiene sus ventajas y desventajas, donde finalmente se considera que utilizar una fusión de las dos técnicas es bastante provechoso y por lo tanto, se proporciona un marco débilmente supervisado para el reconocimiento de imágenes denominado aprendizaje de múltiples instancias basado en redes neuronales convolucionales (MILCNN) cuyo principal objetivo es el reconocimiento de imágenes. Adicionalmente, se presenta una formulación matemática sobre cómo incorporar el concepto de aprendizaje de múltiples instancias a una arquitectura de aprendizaje profundo.

3.5 Esquemas o formas de representación

Dada la necesidad de analizar y seleccionar diferentes formas y métodos para representar la información para ser utilizada en los SIVA, específicamente para los fines de este estudio se analizarán los siguientes: enfoques clásicos basados en probabilidades, uso de características de bajo nivel, uso de las características obtenidas por una CNN y finalmente, las disimilitudes como una forma de representación.

3.5.1 Enfoques clásicos basados en estadística y probabilidades

Se parte de la premisa de que cualquier aplicación en el área de la inspección visual automática pretende resolver un problema con un universo de objetos que son de interés, el cual normalmente, es muy grande y del que realmente sólo se dispone de una pequeña muestra. Es importante extraer la información más relevante del conjunto de datos, esperando que sea aplicable y que represente al gran volumen de datos que aún no se ha revisado. Por tal razón, cada objeto se describe mediante una serie de variables que corresponden a sus propiedades, donde las variables suelen denominarse atributos. En general, hay muchos tipos de variables que se pueden utilizar para representar las propiedades de un objeto, entre las cuales se pueden distinguir al menos seis tipos principales de variables, tales como: nominales, binarias, ordinales, enteras, a escala de intervalo y de escala proporcional [Bramer. 2020].

[4] https://www.vicos.si/resources/kolektorsdd/
[5] https://www.mvtec.com/company/research/datasets/mvtec-ad
[6] https://www.kaggle.com/datasets/kaustubhdikshit/neu-surface-defect-database

3.5.2 Extracción de características

La extracción de características es un proceso que busca reducir la dimensionalidad de un objeto, en el que un conjunto inicial de datos sin procesar se divide y reduce a grupos más manejables o representativos para su posterior procesamiento [Yang et al.. 2007]. La extracción de características es el nombre que reciben los diferentes métodos que seleccionan y/o combinan las variables en características, reduciendo de forma efectiva la cantidad de datos que deben procesarse, sin dejar de describir de forma precisa y completa el conjunto de datos original [Nixon and Aguado. 2012].

La extracción de características de una imagen es una técnica que describe la información ya sea de color, textura o forma de la imagen bajo un conjunto de descriptores de características; así, la obtención de una expresión eficaz y coherente de las características de la imagen es la clave para extraer información ya sea de bajo o alto nivel.

Según Wang et al. [2019a], la extracción de características se divide en dos casos: i) la descripción de las características basada en la aplicación, como el histograma de color (CH), los patrones binarios locales (LBP) y los descriptores de Fourier (FD) que son utilizadas para la descripción de las características de bajo nivel de la imagen; y ii) enfoques matemáticos para la reducción de la dimensión de las características, como el análisis de componentes principales (PCA), el análisis discriminante lineal (LDA) y el análisis de componentes independientes (ICA).

Características de bajo nivel (color, textura y/o forma)

Las características de bajo nivel se pueden definir como aquellas características básicas, que pueden ser extraídas de forma automática a partir de una determinada imagen sin hacer uso de ningún tipo de información sobre la forma y sus relaciones espaciales. Las principales técnicas utilizadas incluyen, la detección de bordes, la cual se puede dar ya sea por operadores de detección de bordes de primer orden, operadores de detección de segundo orden o, el uso de otros operadores como los de Sobel, Canny, Marr-Hildreth, Spacek y Petrou [Nixon and Aguado. 2012].

Es posible clasificar la extracción de características según el objeto de estudio como son el color, la textura y la forma:

- La extracción de características de color es aquella en la que se representa la información relacionada con los niveles de intensidad de gris de una región de interés, donde se incluyen técnicas como el histograma de color, los momentos de color, los vectores de coherencia de color, los conjuntos de color y los correlogramas de color.
- La extracción de características de textura, en la que se describe la textura de una imagen basándose en la relación entre el valor de intensidad de cada píxel en la imagen con sus vecinos, donde se incluyen el método estadístico con la matriz de co-ocurrencia de niveles, el LBP, el método del modelo con el campo aleatorio de Markov y, el método de estructura con primitivas de textura.
- La extracción de características de forma, geométricas o morfológicas, las cuales se basan en calcular la forma de las regiones de interés, a partir del área y el borde de las mismas (área, perímetro, convexidad, momentos de primer, segundo y tercer orden, oblicuidad, etc.); entre estos métodos se incluyen los descriptores de Fourier, la transformada de Hough, los momentos invariantes, los momentos de Zernike, la transformada de HoG, la transformación de características de escala invariable (SIFT) y las Funciones robustas aceleradas (SURF).

CNN como extractor de características

Desde el año 2012, las CNN alcanzaron un nuevo nivel y demostraron su eficacia en el ámbito del reconocimiento de imágenes y especialmente en su utilización para la extracción de características con la invención de *AlexNet*. La razón del éxito de *AlexNet* es que con las DNN se tiene una arquitectura que es flexible, permitiendo añadir capas de neuronas, lo que aumenta la capacidad de aprendizaje de estas redes. La siguiente ventaja es la disponibilidad de equipos de cómputo cada vez más potentes como son las unidades aceleradoras gráficas (GPU) y las unidades de procesamiento tensorial (TPU), que permiten entrenar más rápidamente los modelos. La CNN es una forma de red neuronal artificial que está especializada en detectar patrones y darles sentido; esta detección de patrones hace que la CNN sea especialmente útil para el análisis de imágenes [Jogin et al.. 2018]. La cantidad y calidad de las características que se obtienen de una red CNN varía dependiendo del tipo de red seleccionada, la profundidad de la misma (número de capas), el entrenamiento que se le realice así como los diferentes parámetros e hiper-parámetros que se utilicen [Dara and Tumma. 2018]. Las redes más comúnmente utilizadas para extraer características son: *VGG*, *AlexNet*, *ResNet*, *InceptionResNetV2* y *GoogleNet*.

3.5.3 Las disimilitudes como esquema de representación

La idea detrás de esta técnica es que, en lugar de trabajar con las características específicas de los objetos (como colores, tamaños y/o valores numéricos entre otros), se calculan medidas de disimilitud entre los objetos. Estas medidas pueden ser una distancia euclidiana, una distancia basada en alguna métrica específica o cualquier otra función que capture la relación de *''cercanía''* o *''separación''* entre los objetos.

La representación basada en disimilitudes de un determinado problema produce una matriz cuadrada y en muchos casos simétrica (dependiendo de la función utilizada) donde se almacenan las disimilitudes entre todos los pares de objetos. Tradicionalmente, solo se utilizan las diferencias entre los objetos de prueba y los objetos de entrenamiento, donde dos objetos reciben la misma etiqueta de clase si su diferencia es suficientemente pequeña. Este comportamiento es similar a la regla del vecino más cercano utilizada en los espacios vectoriales [Duin and Pekalska. 2011].

Como las características no describen los objetos completos, es posible que dos objetos que sean diferentes tengan una distancia cero según las características disponibles. Esta es una causa esencial de superposición de clases en espacios de características. Las disimilitudes ofrecen la posibilidad de superar esto, si la medida de disimilitud se define de tal manera que los objetos tienen una distancia cero a sí mismos o a copias completamente idénticas de sí mismos (que por lo tanto deberían pertenecer a la misma clase), no hay superposición de clases [Duin. 2012].

Las disimilitudes permiten definir con mejor exactitud la clase de los objetos que entran al clasificador, puesto que, teóricamente, podrían evitar el solapamiento de las clases, así como también cuentan con la propiedad de consistencia de clase que garantiza que las clases no se sobreponen en un espacio vectorial, y define que la probabilidad de encontrar un objeto que pertenece a la misma clase es muy pequeña y diferente de cero, ya que las vecindades definidas entre objetos de la misma clase son suficientemente pequeñas [Duin et al.. 2014]. Adicionalmente, las medidas de disimilitud tienen en cuenta el objeto en su totalidad mientras que las medidas por características representan el objeto reduciendo la realidad del mismo a un subconjunto de sus características.

3.6 Métodos de clasificación para imágenes

Uno de los principales objetivos de este trabajo es la clasificación de imágenes, lo que implica asignar etiquetas a objetos o regiones dentro de una imagen. Existen varios enfoques y algoritmos para realizar la clasificación, destacándose los métodos de clasificación clásicos y los métodos de clasificación basados en CNNs.

3.6.1 Métodos clásicos para la clasificación de imágenes

Son aquellos métodos o enfoques tradicionales y bien establecidos para resolver problemas de clasificación en los campos de la AI y el ML, que requieren una selección manual de características (sin utilizar redes neuronales), donde se identifican y se eligen las características más relevantes antes del proceso de clasificación. Generalmente son eficaces para resolver problemas con características lineales o relaciones relativamente simples en los datos, y pueden funcionar bien con conjuntos de datos pequeños o medianos, pero pueden tener dificultades para manejar grandes volúmenes de datos. Dada la naturaleza de su funcionamiento, suelen arrojar resultados que pueden ser interpretables. Entre los métodos clásicos es importante presentar los siguientes:

Árboles de decisión

Un árbol de decisión es una estructura jerárquica que se utiliza en el ámbito del análisis de datos, la AI y el ML para tomar decisiones y realizar clasificaciones. Funciona como un modelo predictivo que toma decisiones al seguir una serie de ramificaciones basadas en características específicas de los datos, para lo cual se divide recursivamente el conjunto de datos en subconjuntos más pequeños basados en características específicas, hasta llegar a las hojas del árbol, cada una de las cuales representa una categoría [Smith. 2017].

Los árboles de decisión han influido en un amplio ámbito del ML, que abarca tanto la clasificación como la regresión. En el análisis de procesos decisivos, un árbol de decisión se emplea para representar de manera visual y explícita la dinámica y la ejecución de elecciones determinantes. Como su nombre lo indica, utiliza un modelo de decisiones en forma de árbol, siendo una herramienta comúnmente utilizada en la minería de datos para buscar una estrategia para alcanzar un objetivo particular, también se utiliza ampliamente en el ML, especialmente en la clasificación de objetos. Los algoritmos basados en un árbol de decisión han sido frecuentemente utilizados debido a su alta precisión, bajo coste computacional y alta interpretación de sus resultados [Liu et al.. 2022a].

Bosques aleatorios

Los bosques aleatorios o bosques de decisión aleatorios son un método de aprendizaje en forma de ensamble para la clasificación, la regresión y otras tareas, el cual funciona construyendo un conjunto de árboles de decisión en el momento del entrenamiento. Para las tareas de clasificación, la salida del bosque aleatorio es la clase seleccionada por la mayoría o voto mayoritario de los árboles. Para las tareas de regresión, se devuelve la predicción media o promedio de los árboles individuales [Mohandoss et al.. 2021]. El espacio de búsqueda para la construcción de los nodos de cada árbol de decisión está limitado por un conjunto de características extraídas al azar. El rendimiento del método depende directamente del conjunto seleccionado, donde un conjunto pequeño puede degradar el rendimiento del clasificador. La técnica aleatoria del enfoque ha demostrado su relevancia y eficiencia, especialmente en datos de alta dimensión (con un elevado número de características).

Máquinas de vectores de soporte

Las SVM son un modelo de aprendizaje supervisado utilizado en tareas de clasificación, el cual se basa en el enfoque de núcleo (kernel), y permite realizar una clasificación no lineal que asigna implícitamente los datos de entrada a un espacio de características de alta dimensión. Las SVM generalmente clasifican los datos mediante la construcción de un hiperplano que determina una línea recta que separa el espacio en dos zonas lo más homogéneas posible. Los puntos más cercanos al hiperplano (y que definen la posición de éste) se denominan vectores de soporte. Cuando los datos son linealmente separables en un conjunto, es suficiente utilizar una SVM lineal. Cuando los datos no son linealmente separables, se puede seleccionar una función kernel para mapear los datos en un espacio de mayor dimensión. El objetivo de esto es forzar los puntos de datos para que sean linealmente separables siempre y cuando sea posible [Liu et al.. 2005]. Las funciones de núcleo más frecuentemente utilizadas son la función de núcleo lineal, la función de núcleo polinómico y la función de base radial (RBF) [Song et al.. 2008].

Regla de los k vecinos más cercanos

La regla de los k vecinos más cercanos (k-NN) es un clasificador clásico y muy natural. No necesita estimación de densidades ni optimización de funciones, ya que depende completamente de la medida de distancia definida por el usuario. Una ventaja importante es que proporciona una motivación intuitiva de la etiqueta de clase asignada al mostrar al usuario los vecinos más cercanos. Una segunda ventaja es que la medida de la distancia determina completamente el desempeño de la clasificación, ya que no implica un proceso de aprendizaje [Duin et al.. 2014].

El algoritmo de los k-vecinos más cercanos (k-NN) clasifica los datos no etiquetados con respecto a los datos etiquetados más parecidos según una determinada medida de distancia o similitud. Debido a su simplicidad y eficacia, se utiliza ampliamente para realizar una clasificación supervisada en entornos multivariantes. Existen diferentes formas de medir la distancia entre los atributos en el conjunto de prueba y el conjunto de entrenamiento, encontrando que las más significativas son la distancia euclidiana, la distancia de Hausdorff y la distancia de Manhattan [Bramer. 2020]. Una práctica habitual es tomar valores de k que sean impares y se encuentren entre 3 y 9, otras prácticas buscan estimar k como $k = \sqrt[2]{n_g}$ donde n_g es un tamaño de grupo promedio. Otra posibilidad es probar con distintos valores de k, aplicárselo a los puntos de la muestra cuya clasificación es conocida, obtener el error de clasificación en función de k y escoger aquel valor de k que conduzca al menor error observado [Godoy. 2021].

Adicionalmente, el algoritmo de k-NN se define como una técnica de clasificación fundamental y una de las más sencillas cuando hay poco o ningún conocimiento previo sobre la distribución de los datos. Las ventajas de k-NN sobre otros métodos son que es robusto cuando los datos de entrenamiento presentan ruido, eficaz si los datos de entrenamiento son grandes, no requiere de una fase de entrenamiento previa y aprende modelos complejos con facilidad. Entre las desventajas se encuentra que es difícil determinar el número óptimo de vecinos más cercanos y es difícil de aplicar en representaciones con altas dimensiones, lo que lleva a una baja eficiencia computacional, una gran dispersión de datos y una gran cantidad de requisitos de almacenamiento. Tampoco está claro qué tipo de medida de distancia se debe utilizar para cada problema y el coste computacional puede ser bastante elevado [Li et al.. 2022].

Clasificador Bayesiano ingenuo

El clasificador Bayesiano ingenuo es un enfoque de clasificación simple pero robusto que se basa en el Teorema de Bayes, el cual describe la probabilidad de que ocurra un evento a partir del conocimiento previo de la condición que está potencialmente relacionada con el suceso. El clasificador ingenuo de Bayes calcula las probabilidades de todas las clases (valores) para una característica objetivo y selecciona la que tiene mayor

probabilidad. Además, el clasificador Bayesiano ingenuo supone que los valores asociados a cada clase de cada característica siguen una distribución Gaussiana [Benyahia et al.. 2022].

Perceptrón multicapa

Un algoritmo que clasifica las entradas separando dos categorías con una línea recta se denomina perceptrón o clasificador lineal. Un perceptrón multicapa (MLP) es una red neuronal que se compone de más de un perceptrón. Esta conformado por una capa de entrada para recibir la señal, una capa de salida para tomar la decisión o predicción y varias capas ocultas que realmente hacen la predicción. Los MLP se aplican a menudo a problemas de aprendizaje supervisado. A diferencia de otros algoritmos de clasificación, MLP se basa en una red neuronal subyacente para realizar la tarea de clasificación [Chugh et al.. 2020].

3.6.2 Métodos de clasificación basados en redes neuronales convolucionales

Una CNN es una red neuronal especializada para tareas de procesamiento de imágenes. Esto se debe principalmente a su capacidad para procesar datos representados en forma matricial, ya que los píxeles de una imagen forman matrices bidimensionales, y por lo tanto pueden procesarse directamente. A diferencia de los métodos clásicos de ML, los pasos intermedios del procesamiento de imágenes, como la extracción de características y el preprocesamiento del conjunto de datos, no son necesarios al utilizar las CNN, ya que se realizan automáticamente dentro de ellas [Anding et al.. 2019].

Las redes neuronales artificiales (ANN) están formadas por una serie de unidades elementales, denominadas neuronas artificiales, o nodos, las cuales constituyen dispositivos simples de cálculo que, bien a partir de unos valores proporcionados a la entrada, o bien a partir de estímulos recibidos de otras neuronas, proporcionan una respuesta única o salida. Hay tres tipos de unidades en las redes neuronales: de entrada, salida y ocultas. Las unidades de entrada reciben señales desde el entorno (entradas a la red), las unidades de salida envían la señal fuera del sistema (salidas de la red), las unidades ocultas son aquellas cuyas entradas y salidas se encuentran dentro del sistema y no tienen contacto con el exterior [Origel-Rivas et al.. 2020]. Una CNN es una arquitectura de red que se utiliza en DL y que aprende directamente de los datos, sin necesidad de extraer las características manualmente [Origel-Rivas et al.. 2020].

Estas redes son particularmente útiles para encontrar patrones en imágenes, los cuales son utilizados para reconocer objetos, rostros, defectos y escenas entre otros; pero también resultan eficaces para clasificar datos en diferentes formatos, tales como datos de audio, series de tiempo y diferentes tipos de señales.

Típicamente las CNN están conformadas por algunas de las siguientes capas [Khan et al.. 2018; Venkatesan and Baoxin. 2017]:

- Capa Convolucional (Convolutional Layer): Ésta es la capa principal que da nombre a la red convolucional. Utiliza operaciones de convolución para aplicar filtros o núcleos a la entrada. Estos filtros ayudan a detectar características locales, como bordes, texturas y patrones en la imagen.
- Capa de Agrupación (Pooling Layer): También conocida como capa de submuestreo, tiene como objetivo reducir la dimensionalidad de la representación espacial al seleccionar valores prominentes de una región en la entrada. El submuestreo reduce la cantidad de información a procesar y ayuda a lograr cierta invarianza ante las transformaciones.
- Capa de Normalización por lotes (Batch Normalization Layer): Esta capa normaliza las activaciones de una capa anterior para acelerar el entrenamiento y estabilizar el proceso de aprendizaje. Contribuye a que el proceso de entrenamiento sea más rápido y estable.

- Capa Totalmente Conectada (Fully Connected Layer): También conocida como capa densa, son capas en las que cada neurona o nodo está conectado a todos los nodos de la capa anterior. Toma las características aprendidas por las capas anteriores y realiza una combinación lineal de ellas. Esta capa se asemeja a una red neuronal tradicional y típicamente se utiliza para la clasificación final.

- Capa de Activación (Activation Layer): Introduce una no linealidad en la red neuronal al aplicar una función de activación a las salidas de las capas anteriores. Las capas de activación se aplican después de las capas de transformación lineal (como capas densas o capas convolucionales) y son responsables de introducir relaciones no lineales en los datos, lo que permite a la red neuronal aprender funciones más complejas y realizar tareas más avanzadas. Ejemplos de funciones de activación incluyen ReLU (Rectified Linear Unit), sigmoide y tangente hiperbólica.

- Capa de Aplanamiento (Flatten Layer): Esta capa es la que convierte datos de entrada multidimensionales, como una matriz de características, en un vector unidimensional para que puedan ser conectados a una capa totalmente conectada.

- Capa de Salida (Output Layer): Genera las predicciones finales de la red neuronal para una tarea específica. La arquitectura y el número de neuronas en esta capa dependen del tipo de problema que la red esté tratando de resolver (clasificación, regresión, segmentación, etc.). En clasificación, su tamaño normalmente coincide con el número de clases a clasificar.

Entre las ventajas de un clasificador CNN se destaca que contiene muy pocas capas diferentes y cada capa transforma la entrada en la salida a través de un conjunto de funciones; así generan resultados de reconocimiento altamente precisos, y se pueden volver a entrenar para nuevas tareas de reconocimiento, lo que permite aprovechar las redes preexistentes y/o previamente entrenadas.

Como principal desventaja se encuentra que, de forma natural, no codifican la posición y la orientación del objeto en sus predicciones. Adicionalmente, una convolución es una operación significativamente más lenta que otras operaciones de las redes neuronales como por ejemplo, una agrupación, una normalización o una combinación lineal como en una capa totalmente conectada. Si la red es profunda, cada paso de entrenamiento va a tomar mucho más tiempo y requerirá más recursos de máquina tanto para la fase de entrenamiento como para la fase de prueba [Jogin et al.. 2018].

3.6.3 Clasificación basada en distancias y/o disimilitudes

La noción de similitud se originó en la psicología y se estableció para determinar por qué y cómo se agrupan las entidades en categorías, y responder a la pregunta *¿por qué algunas categorías son comparables entre sí y otras no?*. Los diferentes enfoques para modelar la similitud incluyen los basados en características, en redes y los geométricos. Más recientemente, la comunidad de la AI comenzó a investigar los modelos de similitud computacional como un nuevo método para la recuperación de información [Keshavarzi et al.. 2009]. La similitud y la disimilitud están estrechamente relacionadas. Dos objetos que tienen una gran similitud tienen una disimilitud pequeña. El procedimiento a seguir para clasificar un objeto no etiquetado parte de calcular el vector de características asociado a ese objeto y posteriormente, calcular la distancia, similitud o disimilitud, según sea el caso, de ese vector a cada uno de los centroides u objetos de entrenamiento. A partir de la representación obtenida se pueden utilizar diferentes algoritmos como el de k-NN o las SVM para establecer la clase de un nuevo objeto [Trosset et al.. 2008]. En otros enfoques, podría no haber vectores de características asociados, de modo que la (di)similitud se calcule directamente a partir de los datos crudos, es decir, sin caracterizar [Yen and Cios. 2008].

Medidas de disimilitud entre imágenes

Estas medidas proporcionan una medición cuantitativa del grado de coincidencia entre dos imágenes, o parches de imágenes. Las medidas de similitud entre imágenes desempeñan un papel importante y son la base de funcionamiento de muchos algoritmos y aplicaciones de imágenes, incluida la recuperación, clasificación, detección de cambios, evaluación de calidad y registro.

Es importante aclarar que se utiliza el término ‘‘medida de similitud entre dos imágenes A y B’’ como un término general que incluye tanto medidas de similitud (que alcanzan su valor máximo cuando A = B) como medidas de disimilitud o distancia (que alcanzan su valor mínimo cuando A = B) [Piotrowski. 2009].

La verificación de imágenes se utiliza ampliamente en diferentes escenarios para establecer correspondencia entre dos imágenes y es un problema fundamental en diversos campos. Se calcula una transformación espacial que alinea una imagen de origen con una imagen de destino. Anteriormente se hacía coincidir las características correspondientes en las imágenes, pero más recientemente el interés se ha dirigido hacia medidas de similitud cuya naturaleza es de correspondencia global obtenida a partir de intensidades en las imágenes. En este caso, el rendimiento de la verificación de imágenes depende directamente de la eficacia de la función utilizada para medir la similitud entre ellas [Razlighi et al.. 2013].

Entre las medidas de disimilitud que permiten comparar imágenes directamente es importante destacar: Chernoff, Bhattacharyya, Jeffrey's-Matusita, Kullback-Leibler, Chi cuadrado (χ^2), error cuadrático medio, error medio absoluto y coeficiente de correlación cruzada [Mitchell. 2010].

Medidas de disimilitudes entre conjuntos de puntos

Sean $\mathcal{X}_{(i)} = \left\{\vec{X}_{(i,1)}, \ldots, \vec{X}_{(i,k)}, \ldots \vec{X}_{(i,N_i)}\right\}$ y $\mathcal{X}_{(j)} = \left\{\vec{X}_{(j,1)}, \ldots, \vec{X}_{(j,l)}, \ldots \vec{X}_{(i,N_j)}\right\}$ dos bolsas a comparar. Las siguientes son, según Cheplygina and Tax [2015], cuatro opciones para calcular las disimilitudes entre conjuntos de puntos:

$$d_{maxmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \max_k \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{3-8}$$

$$d_{minmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \min_k \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{3-9}$$

$$d_{meanmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \frac{1}{N_i}\sum_{k=1}^{N_i} \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{3-10}$$

$$d_{meanmean}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \frac{1}{N_i N_j}\sum_{k=1}^{N_i}\sum_{l=1}^{N_j} d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{3-11}$$

La medida de disimilitud de la Ec. (3-8) no es más que la adaptación al caso de las bolsas de la conocida distancia de Hausdorff entre conjuntos. Las medidas de disimilitud de la Ec. (3-9), Ec. (3-10) y Ec. (3-11), a saber d_{minmin}, $d_{meanmin}$ y $d_{meanmean}$ son medidas simétricas útiles para resolver el problema de encontrar la distancia entre bolsas. Las anteriores medidas son no métricas, pero se pueden utilizar en el enfoque de disimilitudes, porque la matriz de disimilitud se utiliza como un conjunto de características y la relación entre las características no está restringida como las distancias [Cheplygina et al.. 2015]. Se ha señalado en Cheplygina and Tax [2015] que las medidas de disimilitud más populares y utilizadas con frecuencia son

d_{maxmin} y d_{minmin}; sin embargo, la última es menos sensible a los valores atípicos que la primera, lo que la hace útil cuando los conjuntos de datos contienen datos atípicos.

3.7 Métodos de localización

La localización de un defecto en los SIVA consiste en encontrar la ubicación exacta de un defecto en un objeto o en una imagen. Entre los principales métodos para localizar defectos es importante destacar: detección de objetos tradicional (con búsqueda selectiva), detección de objetos en una etapa, detección de objetos en dos etapas y, detección de objetos basada en mapas de calor.

3.7.1 Detección de objetos tradicional con búsqueda selectiva

Los algoritmos tradicionales de visión artificial y específicamente los que realizan detección de objetos, se centran en la extracción de características de la imagen, por ejemplo, la detección de bordes. Los bordes son puntos característicos de la imagen ya que son zonas con mucha textura, con cambios de color y que son relativamente sencillos de identificar. Es importante determinar las características de una imagen ya que son las que nos permitirán obtener la información y comprensión que necesitamos sobre la misma. A partir de estas características, se pueden aplicar diferentes tipos de algoritmos para identificar los posibles objetos que puedan estar al interior de la imagen.

Otra herramienta es la segmentación de imágenes. Esta aproximación parte de que, además de la etiqueta de la imagen, se cuenta con un método de segmentación para extraer las regiones de interés en la imagen, el cual consiste en aplicar un filtro de mediana para la extracción del fondo de la imagen y después aplicar un método de umbralización para así aislar los defectos potenciales [Mera. 2017].

3.7.2 Detección de objetos en una etapa:

Son un conjunto de métodos de detección que se caracterizan por revisar las imágenes una sola vez a través de una única etapa que se encarga de todas las tareas de detección. Entre los principales métodos de detección en una sola etapa están:

OverFeat: Es un modelo de CNN que fue desarrollado para la tarea de clasificación de imágenes y detección de objetos, el cual combina las tareas de clasificación y detección en un único marco de trabajo [Sermanet et al.. 2013].

YOLO: Es el acrónimo de *''You Only Look Once''* (sólo se mira una vez), y que cubre un conjunto de algoritmos para la clasificación de objetos, cuya principal característica es que analizan cada imagen una sola vez para predecir qué objetos están presentes y dónde se encuentran [Redmon et al.. 2016]. Su funcionamiento se basa en que una red neuronal divide la imagen en regiones, prediciendo los cuadros o cajas de identificación y las probabilidades por cada región; a partir de allí, las cajas delimitadoras se ponderan mediante las probabilidades predichass. Las versiones YOLOv2 y YOLOv3 introducen arquitecturas más profundas y complejas, lo que se traduce en más capas y parámetros, pero también en un aumento en la capacidad de representación.

SSD: Es el acrónimo de *''Single Shot Detector''* (detector de disparo único), y son un conjunto de algoritmos que se basan en una arquitectura de CNN que procesa cada imagen una sola vez y realiza predicciones en

múltiples niveles de resolución para detectar objetos de diferentes tamaños en una sola pasada utilizando cajas delimitadoras. Esto los hace eficientes y adecuados para aplicaciones en tiempo real donde se requiere una detección rápida y precisa de objetos en imágenes. El núcleo de SSD consiste en predecir las puntuaciones de las categorías y los desplazamientos de las cajas delimitadoras para un conjunto fijo mediante pequeños filtros convolucionales aplicados a los mapas de características [Liu et al.. 2016].

3.7.3 Detección de objetos en dos etapas:

A diferencia de los algoritmos en una sola etapa, como YOLO y SSD, que realizan la detección en una sola pasada, los algoritmos en dos etapas dividen el proceso de detección en dos fases distintas: primero la generación de regiones candidatas y posteriormente, la clasificación y refinamiento de dichas propuestas.

La familia de los algoritmos R-CNN (Region-based Convolutional Neural Networks) son una familia de modelos y variantes diseñadas para la detección de objetos en imágenes utilizando CNNs y enfoques basados en regiones de interés. Estos algoritmos necesitan imágenes etiquetadas o anotadas con las ubicaciones y las clases de los objetos que se desea detectar. Cada anotación debe incluir las coordenadas de la caja delimitadora alrededor del objeto y la etiqueta de la clase a la que pertenece [Hmidani and Ismaili Alaoui. 2022].

R-CNN: Se denomina algoritmo de CNN basado en regiones (Region-based Convolutional Neural Networks), el cual se desarrolló con el fin de solucionar el problema de seleccionar un número muy grande de regiones para el proceso de clasificación de objetos. Utiliza una búsqueda selectiva que permite extraer 2000 regiones de cada imagen que se denominan propuestas de regiones. Para cada región de interés (RoI), se extraen las características que la definen utilizando una CNN previamente entrenada. Las características extraídas de cada RoI alimentan a un clasificador, que determina la probabilidad de que la región contenga un objeto de una clase específica. Finalmente, además de la clasificación, se realiza un proceso de regresión para ajustar las coordenadas de la caja delimitadora que rodea al objeto en la región, lo que ayuda a refinar la precisión de la ubicación del objeto en ella [Dong and Wang. 2016].

Fast RCNN: Se le conoce como el método rápido de R-CNN, el cual mejora la eficiencia de R-CNN al utilizar una única extracción de características para todas las regiones propuestas, y optimiza simultáneamente la clasificación y la regresión de las cajas delimitadoras, lo que resulta en un algoritmo de detección de objetos más rápido y preciso. Se utiliza una CNN para generar características en toda la imagen, luego, las RoIs potenciales se obtienen utilizando diferentes técnicas, y para cada RoI propuesta, se extraen características utilizando una RoI y una CNN pre-entrenada. Esto permite que las RoIs se ajusten a un tamaño fijo y se conviertan en entradas compatibles con la red. Las características de cada RoI se pasan a dos ramas paralelas en la red: una rama para la clasificación y otra para la regresión de la caja delimitadora. La rama de clasificación predice las probabilidades de pertenencia a cada clase, mientras que la rama de regresión ajusta la ubicación y tamaño de la caja delimitadora alrededor del objeto [Girshick. 2015]. La razón por la que Fast R-CNN es más rápida que R-CNN es porque no hay que alimentar cada vez la CNN con 2000 propuestas de regiones. En su lugar, la operación de convolución se realiza una sola vez por imagen y a partir de ella se genera el mapa de características [Hsu et al.. 2018].

Faster R-CNN: Se denominan como las redes R-CNN más rápidas, las cuales a diferencia de los algoritmos anteriores de R-CNN y Fast R-CNN no utilizan la búsqueda selectiva para encontrar las propuestas de regiones, sino como lo plantean en [Ren et al.. 2017], utilizan un algoritmo de detección de objetos que permite que la red directamente aprenda las propuestas de región. Las denominadas redes R-CNN más rápidas constan de dos etapas. La primera etapa, denominada Red de Propuesta de Región (RPN), donde una CNN se encarga de proponer cajas delimitadoras de los objetos candidatos. La segunda etapa, utiliza otra CNN que extrae características de cada caja o región candidata y realiza la clasificación de los objetos y la posterior regresión de las cajas delimitadoras [Marin et al.. 2021].

Mask RCNN: Se denominan redes R-CNN con máscaras. Es una CNN de última generación utilizada para la segmentación de imágenes y de instancias. Mask R-CNN se desarrolló sobre las Faster R-CNN, como una CNN basada en regiones y adopta el mismo procedimiento de dos etapas, con una primera etapa que se encarga de generar las RPN, mientras que en la segunda etapa, el algoritmo además de predecir la clase también genera una máscara binaria para cada región de interés. Faster R-CNN tiene dos salidas para cada objeto candidato, una etiqueta de clase y un desplazamiento de la caja delimitadora, mientras que, Mask R-CNN añade una tercera salida que da origen a la máscara del objeto [Vemula and Frye. 2020].

SPP-net: Se denomina red de agrupación de pirámide espacial (Spatial Pyramid Pooling Net), la cual es una arquitectura de CNN que emplea una capa de agrupación piramidal espacial (SPP) para eliminar la restricción de tamaño fijo de las redes. Específicamente, se agrega una capa SPP sobre la última capa convolucional. La capa SPP agrupa las características y genera resultados de longitud fija, que luego se introducen en las capas completamente conectadas (u otros clasificadores).

Antes de que existiera la agrupación de pirámide espacial, el mapa de características extraído de una imagen generalmente se aplanaba o se agrupaba, lo que proporcionaba una salida de tamaño variable y dificultaba los procesos de detección. La agrupación de pirámides espaciales mantiene la información en contenedores espaciales locales, donde el número de contenedores y su tamaño es fijo [Wu and Yan. 2023].

Las capacidades de SPP-net son más notorias en los procesos de detección de objetos. Con SPP-net es posible calcular los mapas de características de una imagen completa una sola vez y luego se agrupan las características en regiones arbitrarias (subimágenes) para generar representaciones de longitud fija y a partir de allí entrenar los diferentes detectores [He et al.. 2014].

3.7.4 Detección de objetos basada en mapas de calor

Los mapas de activación de clase (CAM) son una familia popular de técnicas que genera mapas de calor (heatmaps) específicos de clase como subproductos del proceso de predicción dentro los modelos. Estas técnicas se basan en los principios de MIL que establecen que cada decisión tomada por el modelo debe ser atribuible a al menos una evidencia en la imagen de entrada. Por diseño, la metodología de los CAM se enfoca en explicar o interpretar las capas superiores de un modelo profundo e indican la probabilidad de que un objeto esté presente en cada región de la imagen y luego, se establece un umbral para detectar objetos basados en estos mapas. Los métodos CAM se inventaron para explicar modelos convolucionales profundos en los dominios generales de imágenes, donde una de las principales características de este método de detección es que no requiere de una anotación experta de la ubicación del objeto, sino que se puede entrenar usando solo una etiqueta a nivel de imagen bajo el WSL. Esta capacidad es especialmente importante para el dominio de los problemas donde las anotaciones por parte de los expertos a menudo no están disponibles o son costosas de obtener [Preechakul et al.. 2022].

3.8 Comparación de algoritmos de clasificación/localización

Una vez presentados los diferentes conceptos, formas de representación, métodos y algoritmos requeridos para desarrollar SIVAs, en la Tabla **3-1** se presenta una comparación de diferentes métodos que pueden ser utilizados para la clasificación y detección en imágenes, indicando el método, una breve descripción, las ventajas y desventajas de su uso, y finalmente algunos ejemplos de algoritmos que utilizan el método.

Tabla 3-1: Comparación de métodos de representación y clasificación/detección de objetos en imágenes

Método	Descripción	Ventajas	Desventajas	Ejemplos de Algoritmos
Histogramas de color	Representa objetos según la distribución de colores en la imagen.	Simple y rápido de calcular.	No considera la estructura espacial.	k-NN
Descriptores SIFT	Detecta puntos de interés y describe sus alrededores con vectores.	Robusto a cambios de escala y rotación.	Sensible a cambios de iluminación y oclusión.	SVM
Árboles de decisión	Representaciones gráficas para la resolución de problemas.	Simples de entender y de interpretar	Tienden al sobre entrenamiento y pueden ser inestables.	AdaBoost, bosques aleatorios.
Bosques aleatorios	Utilizan múltiples árboles de decisión.	Alta precisión y escalabilidad.	Alto consumo de recursos.	XGBoost RF, Máquina potenciadora de gradientes de luz.
Bolsa de palabras	Representa una imagen como un conjunto de palabras visuales.	Efectivo para clasificación de categorías generales.	Ignora la estructura espacial y la relación entre palabras.	Bosques aleatorios
MIL	Las imágenes se presentan en forma de bolsas de instancias y posteriormente se clasifican diferenciando entre bolsas positivas y negativas.	Solo requiere información sobre la presencia o ausencia de la clase de interés en una imagen y requiere pocos datos.	Dificultad para proporcionar localizaciones precisas de los defectos dentro de las imágenes.	MILES, mi-SVM, SimpleMI, EM-DD.
Redes neuronales artificiales	Redes neuronales que siguen una arquitectura donde la información fluye en una sola dirección.	Buena capacidad para modelar funciones no lineales.	Requiere grandes conjuntos de datos para el entrenamiento.	Perceptrón multicapa
Redes neuronales convolucionales	Aprenden automáticamente características jerárquicas de una imagen.	Altamente efectivas en tareas de visión por computador.	Requieren grandes conjuntos de datos y alta capacidad de cómputo.	CNN
Redes neuronales recurrentes	Modelan secuencias en imágenes, útiles para tareas de seguimiento.	Considera relaciones temporales en secuencias.	Pueden ser costosas de entrenar y sensibles a la longitud de la secuencia.	Redes de memoria larga a corto plazo (LSTM)
Redes neuronales convolucionales 3D	Extienden las CNN para datos volumétricos, como vídeo.	Capturan información espacial y temporal.	Requieren datos 3D, lo que puede ser escaso.	3D-CNN
Redes neuronales siamesas	Comparan la similitud entre pares de imágenes u objetos.	Útiles para tareas de verificación y emparejamiento.	Requieren datos emparejados para el entrenamiento.	Redes neuronales siamesas

Continúa en la siguiente página

Tabla 3-1: Comparación de métodos de representación y clasificación/detección de objetos en imágenes (continuación)

Método	Descripción	Ventajas	Desventajas	Ejemplos de Algoritmos
Redes neuronales de atención	Modelan interacciones a larga distancia en imágenes.	Eficientes para capturar relaciones a larga distancia.	Pueden requerir recursos significativos.	Transformadores de visión (ViT)
Redes siamesas con atención	Combinan elementos de redes Siamesas y redes Transformer para tareas de clasificación.	Buen desempeño en una variedad de tareas.	Requieren grandes modelos y conjuntos de datos.	Convoluciones sobre características jerárquicas
Algoritmos de una etapa	Realizan la detección y clasificación en una sola pasada.	Eficientes y rápidos para tareas de detección.	Pueden ser menos precisos que los de dos etapas.	OverFeat, YOLO, SSD
Algoritmos de dos etapas	Realizan primero la detección y luego la clasificación.	Tienden a ser más precisos en la detección de objetos.	Pueden ser más lentos que los de una etapa.	R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, SPPNet.
Uso de disimilitudes	Mide la diferencia entre objetos y referencia en lugar de asignar etiquetas directamente.	Útil en tareas de búsqueda, recuperación, agrupación y clasificación.	Requiere una métrica de disimilitud adecuada.	MIL con disimilitudes, k-NN con disimilitudes
Uso de probabilidades	Calculan la probabilidad condicional de que un objeto con un conjunto de características particulares pertenezca a una etiqueta.	Solo se requiere una pequeña cantidad de datos de entrenamiento.	Requiere independencia entre las características.	Bayesiano ingenuo.
Mapas de calor	Identifican regiones de interés en función de la distribución de valores de intensidad.	Identifica patrones y facilita la interpretación.	Se dificulta la elección de parámetros.	k-Medias, KDE.

3.9 Interpretación e interpretabilidad

Otro aspecto de gran interés en el desarrollo de este trabajo es lograr la interpretación de los resultados obtenidos, donde la interpretación, también conocida como explicación, se utiliza para explicar o revelar la forma en que los modelos toman las decisiones, así como para identificar las características discriminatorias utilizadas para las decisiones de un modelo, o para establecer la importancia de cada muestra de entrenamiento como contribución a la inferencia o el resultado. Por otro lado, la interpretabilidad del modelo se refiere a identificar las propiedades intrínsecas de un modelo que miden en qué grado el resultado de la inferencia del modelo es predecible o comprensible para los seres humanos [Molnar. 2019].

3.9.1 Interpretación de modelos basados en CNN

Las interpretaciones se calculan mediante procesos que tienen como objetivo explicar o revelar las diferentes formas y estrategias en que los modelos toman las decisiones [Li et al.. 2021b]. Interpretar los modelos basados en CNN es un nuevo desafío debido a su naturaleza compleja y de caja negra que los caracteriza. Las CNN son altamente efectivas para resolver diferentes tareas, pero llegar a entender cómo toman decisiones puede ser difícil debido a la gran cantidad de parámetros y operaciones involucradas.

Un modelo profundo necesita interpretaciones porque la salida o inferencia del modelo no muestra el razonamiento interno que se realizó. Por tanto, un algoritmo de interpretación está diseñado para producir interpretaciones que expliquen las decisiones que tomó el modelo y permitan comprender su razonamiento interno. Todos los resultados producidos por los algoritmos de interpretación que ayudan a comprender el modelo se consideran interpretaciones [Liang et al.. 2021].

Es importante analizar las diferentes categorías de los algoritmos de interpretación, ya que proporcionan información adicional para ayudar a entender los modelos profundos. Siendo importante tener en cuenta qué es lo que se quiere entender o comprender, de tal manera un algoritmo que busca las dificultades del proceso de entrenamiento ayuda a inspeccionar el proceso de entrenamiento del modelo; o un algoritmo que calcula la importancia de las características ayuda a darse cuenta de cuáles son las características más importantes que el modelo utiliza para tomar decisiones; un algoritmo que investiga los resultados intermedios de una red neuronal ayuda a comprender el proceso de toma de decisiones del modelo; y así se podrían definir algoritmos que ayudarán a entender o comprender cada una de las partes y elementos que constituyen un modelo de aprendizaje profundo [Li et al.. 2021b].

Con el fin de interpretar un modelo se pueden utilizar diferentes herramientas como las redes adversarias generativas (GANs) [Cheng et al.. 2023], los mapas de activación de clases (CAM) [Batchuluun et al.. 2023], los estimadores locales [Botari et al.. 2020], entre otros.

3.9.2 Interpretabilidad de los modelos basados en CNN

Tomando como base las necesidades de los SIVA, la interpretabilidad de un modelo basado en CNN es a veces más importante que otras métricas como la exactitud, la precisión o la exhaustividad, debido a diversos aspectos que involucran la calidad de los procesos, el valor económico de los rechazos y muchos otros.

Partiendo de un enfoque matemático, se puede especificar la definición de interpretabilidad de un modelo de la siguiente manera: es la capacidad del modelo de explicar o presentar los procesos que realizó en términos comprensibles para un ser humano.

Una buena definición para los fines de este trabajo es la que presentan autores como [Miller. 2019], *''la interpretabilidad de un modelo es mayor si es más fácil para una persona razonar y averiguar por qué el modelo ha hecho una predicción. Comparativamente, un modelo es más interpretable que otro si las decisiones del modelo anterior son más fáciles de entender que las decisiones de este último''*.

Dentro de las diferentes herramientas que pueden ser utilizadas para determinar la interpretabilidad de un modelo, vale la pena resaltar:

- CAM: los mapas de activación de clase son una herramienta que mejora la interpretabilidad al proporcionar una visualización intuitiva de las decisiones del modelo. Al mostrar qué partes de una imagen son relevantes para una clasificación, los CAM pueden ayudar a verificar si el modelo está tomando decisiones coherentes y razonables [Payer et al.. 2019].

- Gradiente ponderado de activación máxima: es una técnica que resalta las regiones de una imagen que fueron más importantes para la clasificación de la red. Proporciona una interpretación visual de las decisiones del modelo [Selvaraju et al.. 2020].

- Técnicas de visualización de filtros: permiten visualizar los filtros en las diferentes capas convolucionales de la red para entender qué tipo de características o patrones en las imágenes está buscando la red [Mohamed et al.. 2022].

3.10 Complejidad algorítmica de los modelos CNN

Una vez que se cuente con los diferentes modelos para realizar la detección y localización de defectos en imágenes, es necesario disponer de herramientas que permitan medir los requerimientos en cuanto complejidad computacional, definida esta como la cantidad de recursos (temporales) que necesita un algoritmo para resolver un problema y, por tanto, permite determinar la eficiencia de dicho algoritmo [Brassard and Bratley. 1997].

Con respecto al análisis de la complejidad algorítmica de los modelos CNN, varios autores como Makkar et al. [2017] y Shah and Bhavsar [2022] lo desarrollan mediante el cálculo de la complejidad de tiempo y lo que buscan es determinar los tiempos de ejecución de los diferentes algoritmos, mientras que otros autores como Naeem et al. [2020] se refieren al análisis de las dificultades intrínsecas de los modelos CNN con sus complejidades y buscan presentar el análisis de rendimiento en cuanto a las métricas de clasificación. Autores como Oyedare et al. [2023], recalcan la falta de estudios en el tema y enfatizan las dificultades para establecer una complejidad algorítmica, explorando la compensación entre el tamaño del conjunto de datos, la complejidad del modelo CNN y la precisión de la clasificación en varias configuraciones.

Teniendo en cuenta los anteriores enfoques y aportes, los criterios que se van a emplear para evaluar la complejidad algorítmica de las CNN en este trabajo se regirán por la notación clásica como se presenta en [Brassard and Bratley. 1997], donde no se proporcionan medidas absolutas sino medidas relativas al tamaño del problema y se evaluarán siempre en el peor de los casos bajo la notación O grande.

3.11 Revisión sistemática de literatura

En esta sección, se presentan brevemente las estrategias de revisión sistemática de la literatura sobre los trabajos que enmarcan el desarrollo de una *metodología heterogénea para la inspección visual automática bajo técnicas de aprendizaje inexactamente supervisado*, con el objetivo de reunir y evaluar críticamente la documentación disponible, identificar brechas en el conocimiento y proporcionar una base sólida para los diferentes desarrollos y prácticas en esta temática.

Este trabajo tiene como marco de referencia la AI y en especial el ML, donde la intersección de dos de sus áreas como son el DL y el WSL, es el lugar donde se centra el desarrollo de esta metodología heterogénea tal como se aprecia en la Figura **3-4**.

Es importante aclarar que esta revisión de literatura se actualizó al finalizar este trabajo de doctorado, razón por la cual la mayor parte de referencias y trabajos que fueron utilizados en el desarrollo de las ponencias, capítulos de libro y artículos tienen fechas anteriores y muchos no figuran en las referencias que resultaron al hacer esta búsqueda sistemática actualizada. Los resultados se utilizaron como base para establecer el estado del arte, determinar los métodos más frecuentemente utilizados, comparar resultados y para próximos desarrollos.

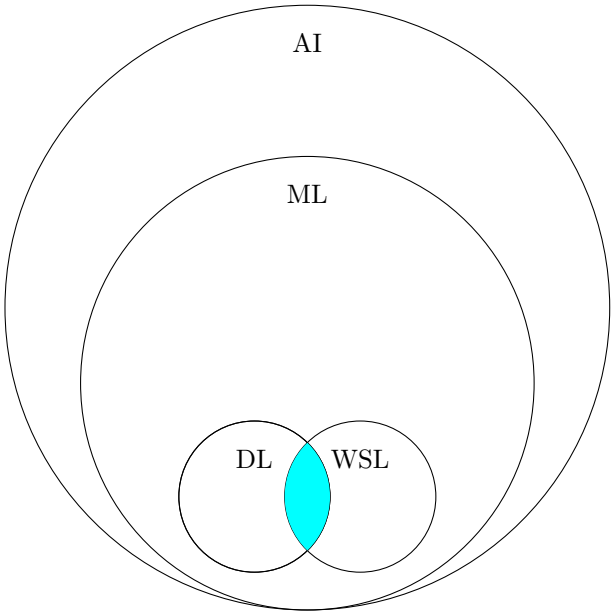


Figura 3-4: Ubicación del área de desempeño de la metodología heterogénea para la inspección visual automática, al formar parte del DL y el WSL con respecto a la AI y el ML.

Para la revisión sistemática se utilizaron las herramientas que proporciona la base de datos de referencias bibliográficas y citas *Scopus*, el uso de diferentes artículos a manera de resúmenes, revisiones de literatura y compendios, así como herramientas para establecer relaciones bibliométricas como *Bibliometrix* [Aria and Cuccurullo. 2017] y VOSviewer. Para delimitar el alcance de la revisión, se seleccionaron las siguientes temáticas de interés:

- Tendencias en el desarrollo de metodologías para la inspección visual automática.
- Evolución de las técnicas del aprendizaje inexactamente supervisado.
- Trabajos más relevantes en la inspección visual automática bajo técnicas de aprendizaje inexactamente supervisado utilizando CNN.

Las búsquedas se limitaron a áreas afines a la problemática de los sistemas de inspección visual automáticos, especialmente a la detección y localización de defectos incluyendo: ingeniería, ciencias de la computación, física y ciencia de materiales; excluyendo principalmente áreas afines con la medicina, agricultura, ciencias humanas, química, biología. Las diferentes búsquedas se realizaron sobre el título, el resumen y las palabras clave de los artículos, conferencias, libros, capítulos de libro y revisiones de literatura.

3.11.1 Tendencias en metodologías para la inspección visual automática

En cuanto al uso y aplicación de la inspección visual automática, marco de referencia sobre el cual se desarrolla la problemática abordada por este trabajo, se realizaron búsquedas con el fin de determinar cuáles son las metodologías más frecuentemente utilizadas, así como sus autores. Para realizar dicha labor, se utilizó la siguiente ecuación de búsqueda:

```
(‘‘visual’’ OR ‘‘optical’’ OR ‘‘vision’’)
AND
```

```
  (‘‘quality assurance OR quality control’’ OR ‘‘process optimization’’ OR
  ‘‘predictive maintenance’’ OR ‘‘inspection’’)
AND
  (‘‘automated’’ OR ‘‘automatic’’ OR ‘‘computerized’’)
```

Luego del respectivo proceso de depuración, donde se acotó principalmente el área de búsqueda, se encontraron 21965 documentos pudiendo constatar que la cantidad de estos aumenta cada año, especialmente en los últimos 5 años, lo que refleja el progresivo interés en esta temática, como se puede ver en la Figura **3-5** que muestra su evolución desde el año 1980.

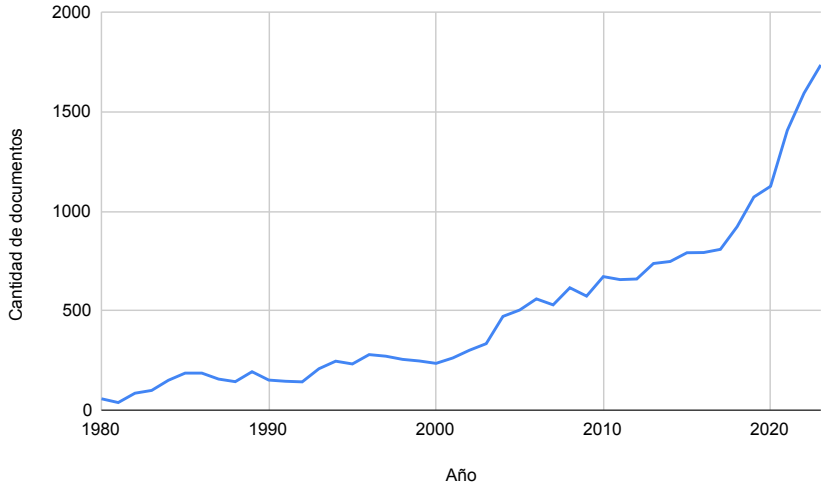


Figura 3-5: Cantidad de documentos publicados durante las últimas décadas sobre los sistemas de inspección visual según Scopus.

Usando los metadatos de los documentos resultantes de esta búsqueda, se procedió a determinar cuáles son los países que más contribuyen en la generación de documentos en el área de inspección visual automática, encontrando que están encabezados principalmente por los Estados Unidos, China y Alemania como se aprecia en la Figura **3-6**.

Finalmente, se buscó determinar cuáles son las metodologías, métodos o algoritmos más utilizados para implementar soluciones para los sistemas de inspección visual en los últimos años, utilizando como referencia el trabajo de Hoffmann and Reich [2023], cuyos resultados se presentan en la Figura **3-7**. Como se puede contemplar las CNN, las SVM y los árboles de decisión son las estrategias más utilizadas en cuanto al desarrollo de soluciones para los SIVA, donde como lo plantea Huang et al. [2020], las CNN estimulan y fomentan el desarrollo de este tipo de sistemas, así como son hoy en día los métodos de vanguardia y el estado del arte para su solución. Con respecto a las SVM, Wen et al. [2023] afirman que son ampliamente utilizadas por los métodos tradicionales de aprendizaje automático, donde generalmente combinan métodos de extracción de características con clasificadores SVM para realizar el reconocimiento de defectos con buenos resultados.

Con respecto a los autores que más publican sobre el tema y las relaciones que se establecen entre ellos, se presenta la correspondiente red de relaciones en la Figura **3-8**, donde se puede establecer que autores como Liu, Y., Zhang, J., Zhang, H., se destacan por ser los autores con mayor número de publicaciones en el área y por formar las mayores agrupaciones.

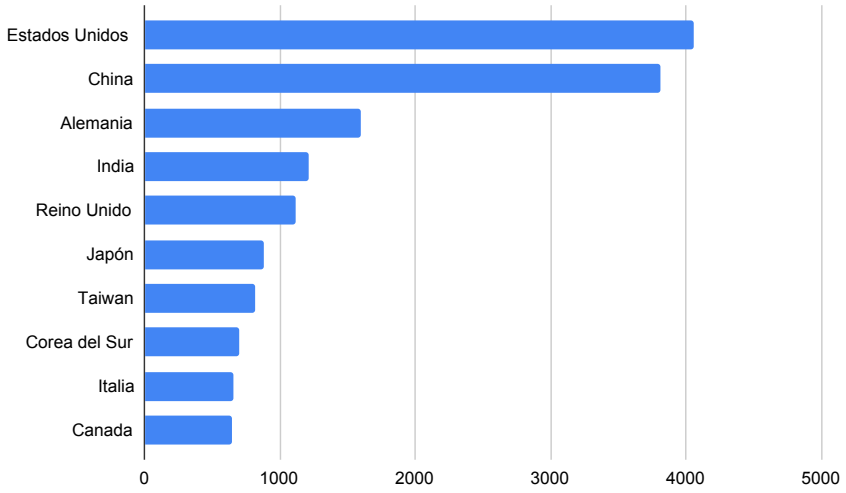


Figura 3-6: Países con mayor número de publicaciones sobre sistemas de inspección visual automática según Scopus.

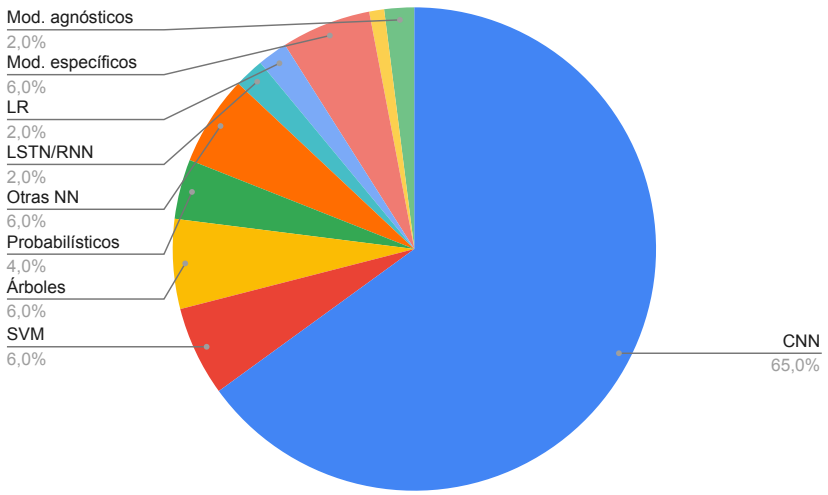


Figura 3-7: Participación de los diferentes métodos para el desarrollo de soluciones en los SIVA según Hoffmann and Reich [2023].

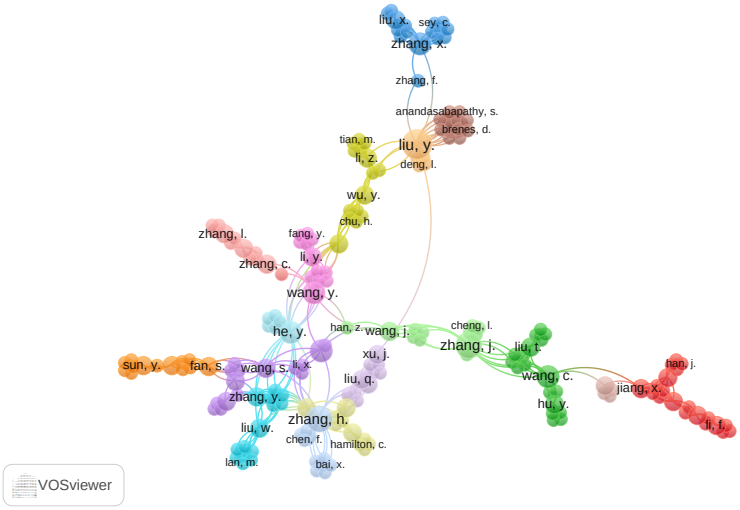


Figura 3-8: Relaciones entre autores de documentos relacionados con sistemas de inspección visual según Scopus.

3.11.2 Evolución de las técnicas de aprendizaje inexactamente supervisado

Las técnicas de aprendizaje inexactamente supervisado son otro marco de referencia para el desarrollo de este trabajo, en el cual las imágenes utilizadas cuentan con etiquetas pero éstas no proporcionan información de la ubicación de los defectos. Mediante la búsqueda bibliográfica se buscó determinar la evolución de las técnicas empleadas, con el objetivo de analizar cuál es el estado del arte y la tendencia. Para tal fin se utilizó la siguiente cadena de búsqueda:

```
(‘‘inexact learning’’ OR ‘‘fuzzy learning’’ OR ‘‘imprecise learning’’ OR
   ‘‘unsupervised learning’’ OR ‘‘semi-supervised learning’’ OR
   ‘‘weakly supervised learning’’ OR ‘‘partially supervised learning’’)
AND
   (‘‘machine techniques’’ OR ‘‘methods’’ OR ‘‘algorithms’’)
```

Como resultado y después de un minucioso proceso de depuración, se obtuvieron 48719 documentos. En la Figura **3-9** es evidente que la cantidad de documentos por año está en un aumento progresivo, especialmente desde el año 2017.

En cuanto al país de origen de las publicaciones, los países que más publican en la temática del aprendizaje inexactamente supervisado son en su orden: China, Estados Unidos y la India, como se pueden ver en la Figura **3-10**. Nótese que China, el país que más publica, duplica en cantidad de publicaciones al segundo país que es Estados Unidos, evidenciando el dominio en el tema por parte de los investigadores de origen chino, lo que se verifica al revisar los diferentes documentos y autores que resultaron de la revisión bibliográfica. Los autores más referenciados están encabezados por Wang, J. , Wang, Z y Li, Y. (todos de origen chino). Las relaciones que desarrollan estos autores se pueden visualizar en la Figura **3-11**.

Con respecto a las metodologías más frecuentemente utilizadas para implementar técnicas de aprendizaje inexactamente supervisado y tomando como referencia el trabajo realizado por Jiang et al. [2023], se presentan dos grandes variantes: los métodos basados en MIL y los que allí denominan, métodos de aprendizaje más allá de MIL. En cuanto a aquellas técnicas que van más allá de MIL, como lo plantea Shao et al. [2022], se basan en el uso de técnicas tradicionales que incluyen extractores de características basados en transformadas de

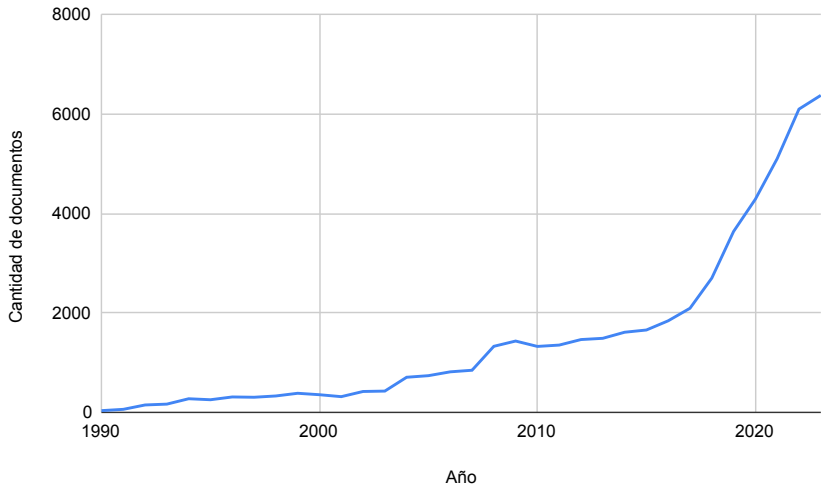


Figura 3-9: Cantidad de documentos publicados sobre aprendizaje débilmente supervisado por año según Scopus.

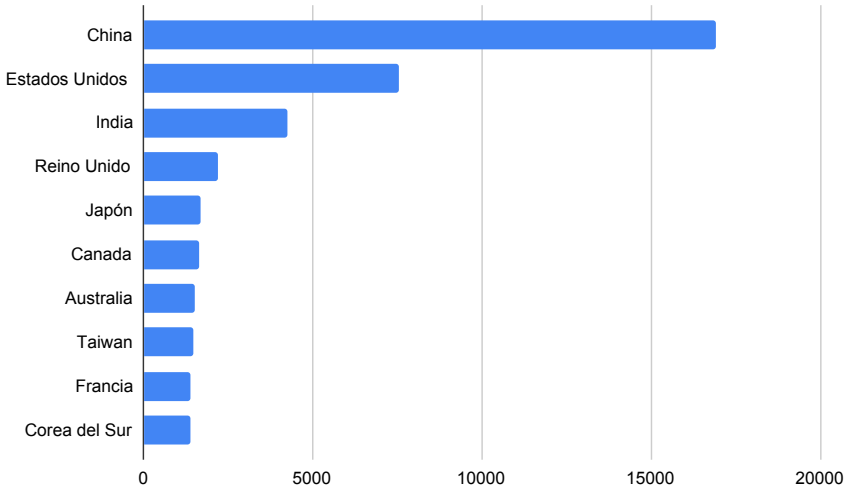


Figura 3-10: Países que más publican sobre aprendizaje débilmente supervisado según Scopus.

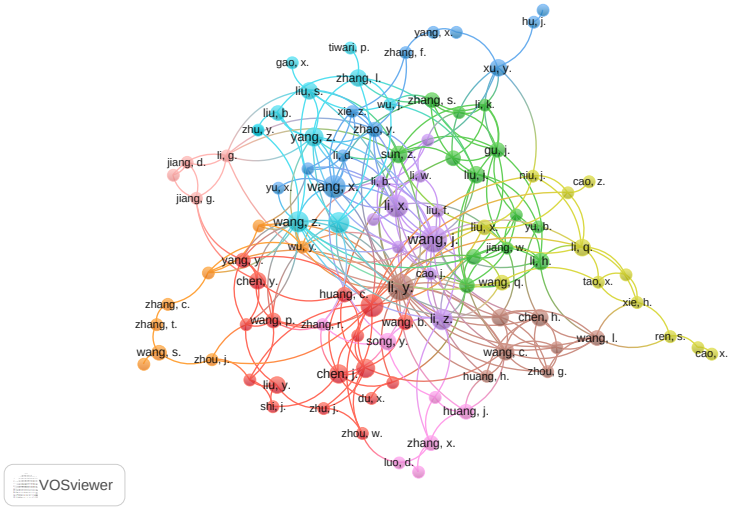


Figura 3-11: Relaciones entre autores sobre aprendizaje débilmente supervisado según Scopus.

HoG, descriptores de SIFT e histogramas de color, pasando a representaciones de niveles superiores como las bolsas de palabras (BoW) y la representación de Fisher. Como clasificador, el más frecuentemente utilizado son las SVM.

3.11.3 Documentos, autores y temáticas más relevantes en inspección visual automática bajo aprendizaje inexactamente supervisado utilizando CNN

Con el fin de acotar y determinar el estado de avance del tema específico de desarrollo de esta tesis, se busca conocer qué trabajos se han realizado que incluyan la inspección visual automática, el aprendizaje inexactamente supervisado y el uso de las CNN. Para tal fin se decidió buscar aquellos documentos que incluyeran las tres áreas, para lo cual se definió la siguiente cadena de búsqueda:

```
( (‘‘visual’’ OR ‘‘optical’’ OR ‘‘vision’’) AND
  ( ‘‘quality assurance’’ OR ‘‘quality control’’ OR ‘‘inspection’’ OR
  ‘‘machine’’ ) )
AND
  ( ‘‘inexact learning’’ OR ‘‘imprecise learning’’  OR ‘‘unsupervised learning’’
   OR ‘‘semi-supervised learning’’ OR  ‘‘weakly supervised learning’’ OR
  ‘‘partially supervised learning’’ )
AND
  ( ‘‘machine techniques’’ OR ‘‘methods’’ OR ‘‘algorithms’’ )
AND
  ( ‘‘convolutional neural networks’’ OR ‘‘cnn’’ )
```

Luego de la respectiva depuración se logró recopilar un total de 145 documentos, pudiéndose establecer la cantidad de documentos publicados sobre la temática año por año (ver Figura **3-12**). Se puede apreciar, tal como en las áreas anteriores, que el tema es de alta relevancia y que cada año, con la excepción de 2023, se aporta cada vez un número mayor de documentos al acervo bibliográfico.

En cuanto al país de origen de las publicaciones, los países que más publican documentos que incluyen las tres temáticas y en total concordancia con las técnicas de aprendizaje inexactamente supervisado, son

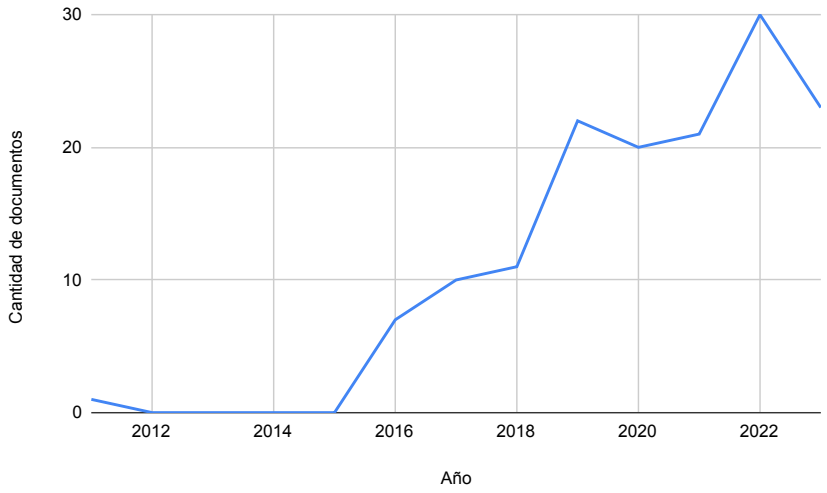


Figura 3-12: Cantidad de documentos publicados sobre sistemas de inspección visual, aprendizaje inexactamente supervisado y CNN, por año según Scopus.

también en su orden: China, Estados Unidos y la India, como se pueden ver en la Figura **3-13**. Con respecto a los autores y las redes que conforman, en la Figura **3-14** se puede contemplar que autores como Wang, Y., Yang, Z., Chen, J. y Huang, Y. son los que predominan, encontrando que autores como Wang et al. [2023] desarrollan métodos que podrían ser similares a los propuestos en este trabajo.

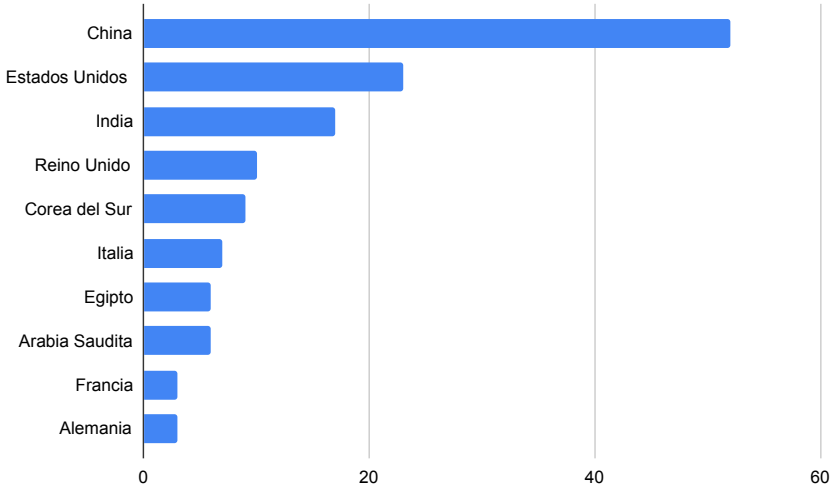


Figura 3-13: Países que más publican sobre inspección visual, aprendizaje inexactamente supervisado y CNN según Scopus.

Tomando como base la información recopilada de los diferentes documentos, se buscó determinar cuáles son las palabras clave más frecuentemente utilizadas, tal como se aprecia en la Figura **3-15**, encontrando que sobresalen *convolutional neural network*, *deep learning*, *machine learning* y *learning systems*. En cuanto a las técnicas utilizadas, se destacan: *feature representation*, *extraction*, *image segmentation*, *support vector machines*, *textures* y *transfer learning*.

Por último, se exploró la disponibilidad de documentos en los cuales se desarrollan búsquedas sistemáticas o

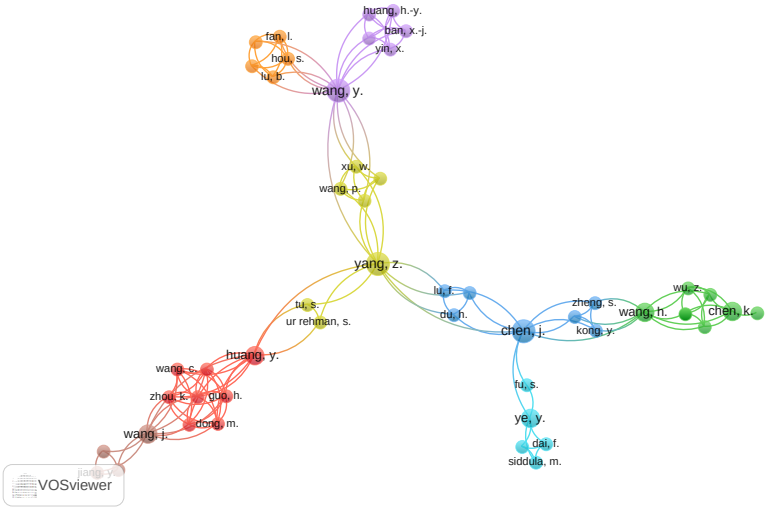


Figura 3-14: Relaciones entre autores sobre inspección visual, aprendizaje inexactamente supervisado y CNN según Scopus.

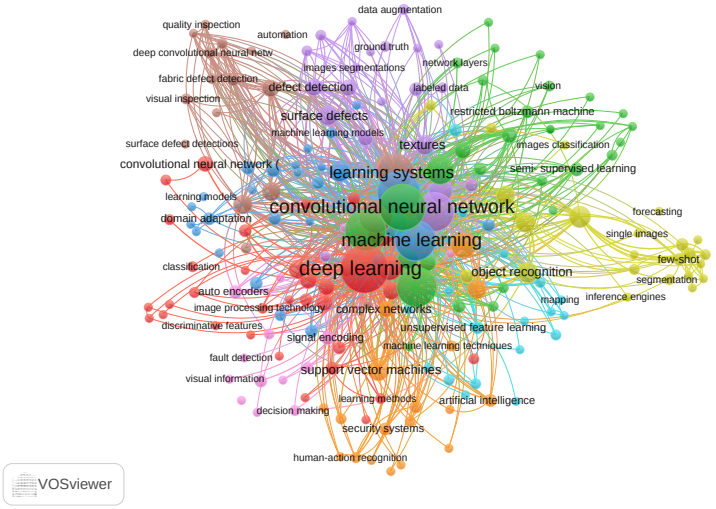


Figura 3-15: Términos más frecuentemente utilizados sobre inspección visual, aprendizaje inexactamente supervisado y CNN según Scopus.

recopilaciones sobre el tema, encontrándose el artículo presentado por Cumbajin et al. [2023], el cual puede tomarse como referencia actualizada de revisión bibliográfica. Dicho trabajo cuenta con una recopilación de trabajos que utilizan DL y CNN hasta el año 2021, evidenciando que arquitecturas de red como U-Net, ResNet, VGG16, VGG19, Xception, MobileNet, Inception, InceptionResNetV2, DenseNet121, AlexNet, R-CNN, Fast R-CNN, Faster R-CNN, Mask R-CNN, YOLO y SSD son frecuentemente utilizadas para resolver este tipo de problemas. En cuanto a clasificadores se destacan las SVM y la regla k-NN; de igual forma, las técnicas más frecuentemente utilizadas como apoyo de las CNN incluyen DA y TL. Finalmente, trabajos como el presentado por Cheplygina et al. [2015], que aunque no involucran el uso de CNN, presentan un aspecto de gran importancia para el desarrollo de este trabajo como lo es el uso de disimilitudes bajo un ambiente MIL.

Otra temática de gran interés para el desarrollo de este trabajo es el uso de disimilitudes, encontrando en la revisión bibliográfica realizada por Costa et al. [2020], autores y trabajos donde se presentan las diferentes posibilidades en términos de representación bajo disimilitudes, resaltando que afirmaciones como la de Pekalska and Duin [2005]: *''En algunos casos, es difícil obtener una descripción eficiente de patrones basada en características con fines de aprendizaje. A veces, ni siquiera los expertos son capaces de describir características de forma inequívoca, obteniendo una representación de alta dimensión o incluso una representación con variables continuas y categóricas mezcladas''* demuestran el potencial de las disimilitudes en el proceso de representación, clasificación y reconocimiento de patrones. Adicionalmente, a continuación, se listan algunos trabajos de alta relevancia para el desarrollo de este trabajo, los cuales se han tomado como referencia, por ejemplo los siguientes: [Keshavarzi et al.. 2009], [Porro-Munoz et al.. 2011], [Duin and Pekalska. 2011], [Duin et al.. 2014], [Alpaydin et al.. 2015], [Cheplygina et al.. 2016] y [Huang et al.. 2022] principalmente.

Parte II

Inspección visual usando técnicas de aprendizaje profundo

4 Clasificación de defectos en láminas de vidrio

Este capítulo se basa en la siguiente publicación (capítulo de libro):

- Eduardo Villegas-Jaramillo y Mauricio Orozco-Alzate. *''Convolutional Neural Networks and Deep Learning Techniques for Glass Surface Defect Inspection'', publicado en IGI Global, pp 67—99., 2022.* [Villegas-Jaramillo and Orozco-Alzate. 2022].

4.1 Introducción

El propósito de este estudio radicó en la necesidad de aprender y aplicar los diferentes conceptos sobre el manejo de las redes neuronales y su aplicación en la solución de problemas reales en el campo de la inspección visual automática. Encontrando que el problema de inspección de láminas de vidrio, por su naturaleza y dificultad, permitía la aplicación de los diferentes conceptos, técnicas y conocimientos adquiridos para llegar a una solución del mismo.

La búsqueda de defectos en superficies como las láminas de vidrio es una tarea que puede realizarse desde diferentes enfoques, entre los que se destaca el DL como una alternativa que ofrece buenos resultados cuando se aplica en sus diferentes variantes como las CNN, las redes profundas y las RL [Zhou et al.. 2019], que pueden complementarse utilizando técnicas como el DA [Shorten and Khoshgoftaar. 2019] y la TL [Hafemann et al.. 2015]. El número de alternativas de configuración de un modelo puede convertirse en un problema combinatorio, lo que dificulta el diseño de un sistema que cumpla tanto los requisitos de rendimiento como las restricciones prácticas. Por lo tanto, el objetivo de este trabajo es presentar un estudio comparativo de diferentes modelos que, basados en DL, abordan el problema de la clasificación de trece categorías de defectos que aparecen en las imágenes de un conjunto de datos público para la inspección de superficies de vidrios (Glass Inspector - GI). El conjunto de datos de GI parece ser difícil de clasificar debido a su heterogeneidad y a las posibles ambigüedades en el etiquetado original; además, hasta donde los autores saben, al momento de escribir este capítulo, el conjunto de datos de GI no se ha utilizado en otros estudios, ya que no se encontraron citas al mismo. El estudio comparativo consiste en la elaboración de diez experimentos con diversas variantes del conjunto de datos, arquitecturas de CNN, redes basadas en RL, TL, DA y el ajuste de los diferentes parámetros e (hiper)parámetros, incluyendo combinaciones de diversas alternativas de configuración. La evaluación se realizó según métricas globales como la exactitud (reportando sus valores mínimos, máximos y promedio) y las medidas micro y promedio de precisión, exhaustividad y valor-F; también se consideraron otras medidas de rendimiento más específicas, incluyendo la matriz de confusión y el informe de clasificación por clase, que se complementaron con curvas de aprendizaje-validación y gráficos de clasificación predictiva. En cuanto al diseño del sistema de clasificación, se adoptaron dos arquitecturas

de redes neuronales, a saber: i) una CNN y ii) una RL que hace uso de TL incorporando una red *''Inception ResNet''* previamente entrenada [Deng et al.. 2009]. Las redes elegidas se combinaron con técnicas de DA que se aplicaron ya sea en una fase previa o en el momento del entrenamiento. La mejor solución se obtuvo con un modelo que utilizaba una CNN basada en RL y DA, reportando una exactitud media de 0.8365, que los autores han considerado como buena dada la naturaleza desafiante del problema.

4.2 Métodos

Se exploró un conjunto de soluciones que requieren inicialmente una etapa de procesamiento de las imágenes, seguida de la aplicación de diferentes enfoques de DL que permitan encontrar una solución adecuada al problema de clasificación de los defectos encontrados en las superficies de vidrio. Se probaron varios tipos de modelos de CNN, desde un modelo de línea base que utiliza una CNN con pocas capas, hasta redes residuales profundas en las que se aplica la técnica de TL; además, se construyeron tres variantes del conjunto de datos de GI, la primera con una partición aleatoria de los conjuntos de entrenamiento y prueba, la segunda que busca garantizar la participación de todas las categorías de defectos tanto en la etapa de entrenamiento como en la de prueba y, finalmente, una variante que busca equilibrar la cantidad de imágenes en las diferentes categorías. Los experimentos propuestos se definen bajo la combinación de las variantes de modelo de red, el conjunto de datos, el uso de TL y las técnicas de DA; las cuales se ilustran en la Figura **4-1**, y se resumen en la Tabla **4-1**.

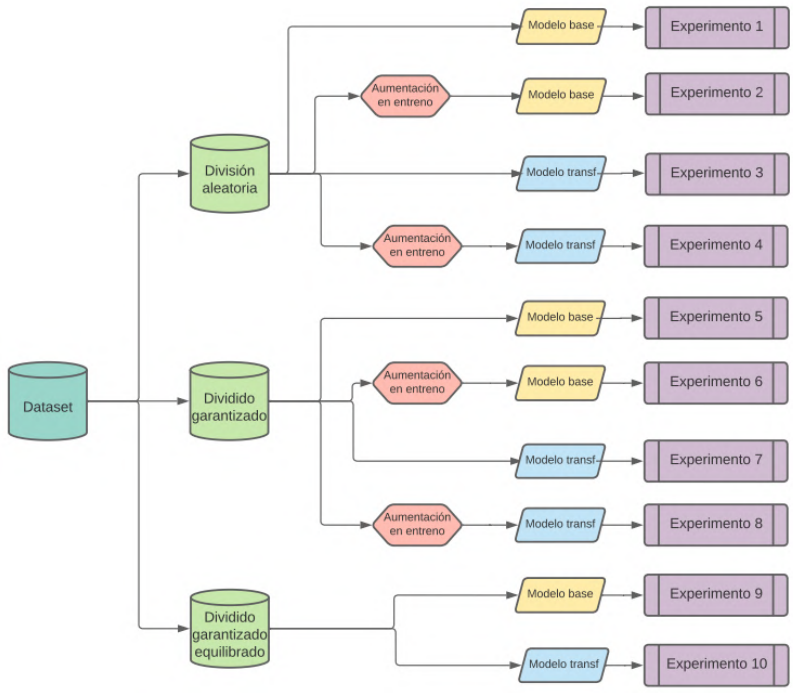


Figura 4-1: Diagrama de árbol que ilustra los diez experimentos considerados, según las diferentes combinaciones del conjunto de datos (original, garantizado, equilibrado), la inclusión o exclusión de DA en el entrenamiento y el diseño del modelo (ya sea el de referencia o por TL).

Tabla 4-1: Resumen de los diez experimentos considerados, según las diferentes combinaciones del conjunto de datos (original, garantizado, equilibrado), la inclusión de DA y el diseño del modelo (ya sea el de referencia/base o el de transferencia).

#	Conjunto de datos	Aumentación	Diseño CNN
1	Original	Ninguna	Base
2	Original	En entrenamiento	Base
3	Original	Ninguna	Transferencia
4	Original	En entrenamiento	Transferencia
5	Garantizado	Ninguna	Base
6	Garantizado	En entrenamiento	Base
7	Garantizado	Ninguna	Transferencia
8	Garantizado	En entrenamiento	Transferencia
9	Balanceado	Diferencial	Base
10	Balanceado	Diferencial	Transferencia

4.2.1 Preprocesamiento de los datos

Las imágenes del conjunto de datos de GI se descargaron del sitio [Deltamax Automazione . 2016], las cuales forman parte del proyecto RISOLVI [Plotegher et al.. 2016]. El conjunto de datos contiene imágenes de 660 defectos presentes en láminas de vidrio con sus respectivas etiquetas para un total de 1160 imágenes; en la Figura **4-2** se muestran algunos ejemplos. Los tamaños de las imágenes son heterogéneos variando entre 61 y 346 píxeles de ancho y entre 39 y 455 píxeles de largo. Cada imagen tiene una etiqueta de clase única según la categoría de defecto que contiene. Los defectos encontrados en el conjunto de datos difieren levemente de los reportados en el documento complementario [Plotegher et al.. 2016], no solo en el número de imágenes por clase, sino también en las propias clases; es decir, algunas de las clases reportadas en [Plotegher et al.. 2016] no están presentes en [Deltamax Automazione . 2016] y viceversa. La Tabla **4-2** muestra las distribuciones nominales según [Plotegher et al.. 2016] y reales según [Deltamax Automazione . 2016] de las imágenes contenidas en el conjunto de datos.

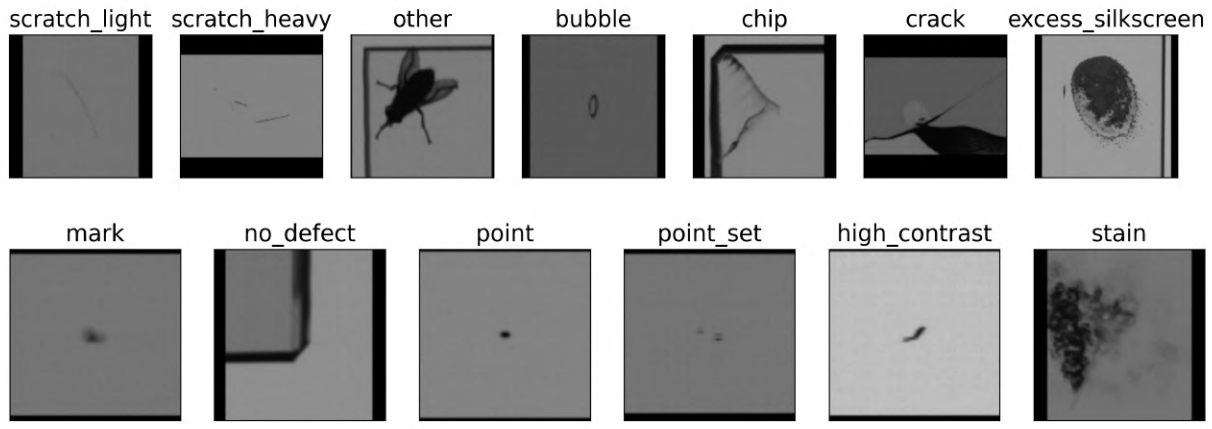


Figura 4-2: Ejemplos de defectos del conjunto de datos GI, con sus correspondientes etiquetas.

Los sistemas de reconocimiento de imágenes basados en CNNs requieren que los tamaños de las muestras de entrada sean iguales; por lo tanto, fue necesario modificar el tamaño de todas las imágenes y, dados los requisitos de algunos modelos, especialmente los basados en TL, se seleccionó un tamaño de 224×224 píxeles con los tres canales (RGB) para todas las imágenes buscando homogeneizar las condiciones de las pruebas. Adicionalmente, con el fin de facilitar los diferentes cálculos, los píxeles de cada una de las imágenes, que

inicialmente tenían valores entre 0 y 255, se normalizaron al rango $[0.0, 1.0]$. Finalmente, en cuanto a las etiquetas, se definieron con números enteros, según la correspondencia mostrada en la Tabla **4-3**.

Para los experimentos 1 a 4, la totalidad de las imágenes se dividió aleatoriamente en dos conjuntos: entrenamiento y prueba, en una proporción de 70 % y 30 % respectivamente; esa partición se denominará en adelante conjunto de datos original, y tendrá 812 imágenes para el entrenamiento y 348 para las pruebas.

Tabla 4-2: Categorías de defectos, con sus nombres en inglés y en español y número de defectos, nominales (según [Plotegher et al.. 2016]) y reales (según [Deltamax Automazione . 2016]), en el conjunto de datos de GI.

Defecto		Nominal	Real
scratch_light	rayón suave	103	103
scratch_heavy	rayón fuerte	51	51
other	otro	1	1
bubble	burbuja	52	52
chip	astilla	81	82
crack	grieta	14	14
excess_silkscreen	exceso serigrafía	0	22
mark	marca	74	74
no_defect	sin defecto	12	13
point	punto	57	57
point_set	conjunto de puntos	107	107
high_contrast	alto contraste	0	3
stain	mancha	82	81
printing_error	error de impresión	26	0
Total		660	660

En cambio, en los demás experimentos, se garantizó la participación de imágenes pertenecientes a todas las categorías tanto en la fase de entrenamiento como en la de prueba para tratar con el problema del desequilibrio de clases. Obsérvese que, como se aprecia en la Tabla **4-2**, algunas categorías tienen un número muy bajo de imágenes (*other* con dos imágenes y *high_ contrast* con cuatro imágenes) mientras que otras clases están mucho mejor representadas (*stain* con 239 imágenes y *scratch_ light* con 201 imágenes).

Tabla 4-3: Etiquetas numéricas y cantidad de imágenes por categoría en el conjunto de datos GI.

Etiqueta	Defecto	# de imágenes
0	scratch_light	201
1	scratch_heavy	103
2	other	2
3	bubble	70
4	chip	133
5	crack	40
6	excess_silkscreen	36
7	mark	132
8	no_defect	13
9	point	63
10	point_set	124
11	high_contrast	4
12	stain	239
Total		1160

De acuerdo con las condiciones antes mencionadas, se construyeron tres variantes del conjunto de datos, a saber:

- **Conjunto de datos original:** este conjunto está compuesto por las 1160 imágenes originales donde la partición en entrenamiento y prueba se realiza de forma aleatoria, en una proporción 70 % - 30 %, justo antes de efectuar cada experimento. Este conjunto de datos se utilizó en los experimentos 1, 2, 3 y 4.

- **Conjunto de datos garantizado:** es una versión del conjunto de datos desarrollada para evitar que algunas de las categorías no tengan imágenes que las representen en el conjunto de entrenamiento. Está conformado por las 1160 imágenes que se distribuyen previamente en un 70 % para el entrenamiento y un 30 % para la prueba, de tal forma que se garantiza la presencia de al menos una imagen de todas las categorías tanto en el conjunto de entrenamiento como en el conjunto de prueba, resultando en 812 imágenes para el entrenamiento y 348 imágenes para la prueba. Este conjunto de datos se utilizó en los experimentos 5, 6, 7 y 8.

- **Conjunto de datos equilibrado:** esta variante del conjunto de datos tiene dos objetivos; el primero, busca garantizar la presencia de imágenes de todas las categorías en los conjuntos de entrenamiento y prueba, y el segundo, busca equilibrar el número de imágenes en las diferentes categorías. Para desarrollar este conjunto de datos se utilizó el proceso de DA, el cual se aplicó de forma diferencial para cada categoría utilizando los siguientes parámetros: rango de rotación de hasta 20 grados, rango de zoom de máximo 15 %, traslación lateral y superior de hasta un 10 % y la posibilidad de realizar operaciones de espejo horizontal y vertical (flip). Tras el proceso de incremento en el número de las imágenes, cuyo objetivo era alcanzar cerca de 400 imágenes por categoría, el tamaño del conjunto de entrenamiento pasó de 812 a 4841 imágenes, mientras que el conjunto de prueba se mantuvo con las 348 imágenes sin aumentar. La composición por clase del conjunto de datos equilibrado se muestra en la Tabla **4-4**. Esta variante del conjunto de datos se utilizó en los experimentos 9 y 10.

Tabla 4-4: Cantidad de imágenes por categoría para entrenamiento (versión original y versión aumentada) y para prueba.

Defecto	Entren. orig.	Entren. aume.	Prueba
scratch_light	141	423	60
scratch_heavy	72	360	31
other	1	400	1
bubble	49	392	21
chip	93	372	40
crack	28	392	12
excess_silkscreen	25	400	11
mark	92	368	40
no_defect	9	396	4
point	44	396	19
point_set	87	435	37
high_contrast	3	399	1
stain	168	504	71
Total	812	4841	348

Modelo base

Como primera opción se diseñó una CNN basada en el repositorio de ejemplos de Keras [1] modificada y adecuada según los requerimientos del problema, de forma tal que permitiera considerar diferentes configuraciones, arquitecturas, número y tipos de capas, así como el ajuste de los diferentes parámetros e hiper-parámetros necesarios para su entrenamiento. Entre las principales propiedades que se tuvieron en cuenta para seleccionar y adaptar este modelo se encuentra el uso de capas convolucionales como núcleo de las CNNs y principal herramienta para el procesamiento de imágenes, así como poder experimentar con

[1] https://keras.io/examples/vision/image_classification_from_scratch/

diferentes tipos de capas incluyendo versiones convolucionales, densas, de regularización, de agrupación y de aplanación; además de explorar opciones para la cantidad de filtros y el número de capas. Tras diferentes ensayos, se seleccionó una arquitectura de CNN con las siguientes especificaciones: 19 capas, comenzando por una capa convolucional que recibe las imágenes con un tamaño de 224×224 píxeles $\times$ 3 canales, aplicando 32 filtros de tamaño 3×3, con el fin de extraer información y reducir su dimensión, seguida por dos secuencias de capas convolucionales, con su debida normalización, que se repiten para 64, 128, 256, 512, 728 y 1024 filtros y tienen como objetivo reducir el tamaño de la imagen y aumentar la información hasta obtener imágenes con un tamaño de 7×7; Posteriormente, se aplica una capa de agrupación global para reducir el tamaño de la imagen, seguida de una capa de regularización así como una capa de abandono que busca eliminar el 50 % de las neuronas. Finalmente, una capa densa que conecta todas las neuronas de la capa anterior con un arreglo de probabilidades cuya dimensión corresponde con el número de clases. El modelo resultante, véase la Tabla **4-5**, se utilizó en los Experimentos 1, 2, 5, 6 y 9.

Tabla 4-5: Arquitectura de la CNN del modelo de referencia.

Capa (tipo)	Modelo
Input	(None, 224, 224, 3)
Conv2D	(None, 112, 112, 32)
Batch normalization	(None, 112, 112, 32)
Conv2D 1	(None, 112, 112, 64)
Batch normalization 1	(None, 112, 112, 64)
Separable conv2D	(None, 112, 112, 128)
Batch normalization 2	(None, 112, 112, 128)
Separable conv2D 1	(None, 112, 112, 128)
Batch normalization 3	(None, 112, 112, 128)
Max pooling 2D	(None, 56, 56, 128)
Separable conv2D 2	(None, 56, 56, 128)
Batch normalization 4	(None, 56, 56, 128)
Separable conv2D 3	(None, 56, 56, 256
Batch normalization 5	(None, 56, 56, 256
Max pooling 2D 1	(None, 56, 56, 256
Separable conv2D 4	(None, 28, 28, 512)
Batch normalization 6	(None, 28, 28, 512)
Separable conv2D 5	(None, 28, 28, 512)
Batch normalization 7	(None, 28, 28, 512)
Max pooling 2D 2	(None, 14, 14, 512)
Separable conv2D 6	(None, 14, 14, 728)
Batch normalization 8	(None, 14, 14, 728)
Separable conv2D 7	(None, 14, 14, 728)
Batch normalization 9	(None, 14, 14, 728)
Max pooling 2D 3	(None, 7, 7, 728)
Separable conv2D 8	(None, 7, 7,1024)
Batch normalization 10	(None, 7, 7,1024)
Global average pooling2D	(None, 1024)
Dropout	(None, 1024)
Dense	(None, 13)

Modelo de aprendizaje por transferencia con aprendizaje residual

Como segunda opción, se seleccionaron algunas de las diferentes arquitecturas de redes basadas en RL que utilizan TL y que, según la literatura [Abubakar et al.. 2020], ofrecen los mejores resultados para problemas de clasificación de imágenes. Donde se destacan las siguientes: *''ResNet50''*, *''ResNet101''*, *''ResNet152''* e *''InceptionResNetV2''* [Xiao et al.. 2018]. Tras haber realizado diferentes pruebas exploratorias con las arquitecturas antes mencionadas y según las recomendaciones dadas en [Hershey et al.. 2017; Kornblith et al.. 2019], se utilizó la red *''InceptionResNetV2''* —previamente entrenada con el conjunto de datos *''ImageNet''*— pero modificada sustituyendo la última capa totalmente conectada por un bloque clasificador acorde con las necesidades del problema. Este bloque incluye una capa de agrupación global media para minimizar el sobreajuste, disminuyendo el número de parámetros, una capa de abandono para desactivar un porcentaje de neuronas, y una última capa densa como clasificador softmax, donde se calcula el valor de la probabilidad normalizada de las trece categorías. Esta arquitectura de red se utilizó en los experimentos 3, 4, 7, 8 y 10.

Incremento de datos (DA)

El proceso de DA puede mejorar la fase de entrenamiento de una CNN ya que, al proporcionar ejemplos de entrenamiento adicionales derivados artificialmente de los originales, los modelos de DL se regularizan ayudando así a evitar el sobreajuste [Mormont et al.. 2018; Voulodimos et al.. 2018]. Teniendo en cuenta la reducida cardinalidad de algunas de las categorías en el conjunto de datos GI y, sobre todo, el desequilibrio entre ellas, se decidió aplicar dos tipos de procesos de incremento para las imágenes: el primero buscando aumentar el número de imágenes en el momento del entrenamiento; el segundo, realizado en un proceso previo, tiene como objetivo equilibrar el número de imágenes entre las categorías haciendo uso de la técnica de DA diferencial [Takase et al.. 2020], que garantiza que todas las categorías estén representadas en el conjunto de entrenamiento, calculando después la cantidad de imágenes adicionales que se requieren. El primer tipo de DA se aplicó en los experimentos 2, 4, 6 y 8, mientras que el incremento diferencial se aplicó a los experimentos 9 y 10.

4.2.2 Configuración experimental

A continuación se detallan el ajuste de los (hiper)parámetros, el entrenamiento de los modelos y la estimación del rendimiento de los modelos utilizados.

Ajuste de (hiper)parámetros y entrenamiento del modelo

Una vez definidos los dos modelos, ya sea con un diseño de CNN base o utilizando RL y TL, se ajustó su funcionamiento mediante la selección y sintonización de los diferentes (hiper)parámetros. Se hizo una revisión de las diferentes funciones de optimización que se pueden utilizar en este tipo de problemas, encontrando que las más adecuadas son: Gradiente Estocástico Descendiente, Estimación Adaptativa de Momentos y Propagación de la Raíz Cuadrada Media (RMSprop) [De et al.. 2018]. De igual manera, en cuanto a la función de pérdida, se tomaron en cuenta las siguientes opciones: Error Cuadrático Medio y Entropía Cruzada Categórica [Chollet. 2021]. Finalmente, como métricas de rendimiento: Exactitud y Exactitud Categórica fueron exploradas. Después de los experimentos de prueba, se encontraron los mejores resultados utilizando RMSprop, Entropía Cruzada Categórica y Exactitud. Para el entrenamiento de los modelos se tuvieron en cuenta las siguientes consideraciones en cuanto a los parámetros: un máximo de 100 épocas para el entrenamiento, incluyendo una opción de parada anticipada cuando el proceso de validación supere las 10 épocas sin mejora y un tamaño de lote de 32 con los cuales se obtuvieron los mejores resultados.

Evaluación del rendimiento

Se realizaron diez repeticiones de cada experimento teniendo en cuenta la naturaleza estocástica de los modelos CNN. Se calcularon las exactitudes, las cuales se presentaron en términos de los valores mínimos, máximos y promedio, junto con la desviación estándar correspondiente; además, para ilustrar y analizar mejor los resultados, se presenta la matriz de confusión acumulada de los experimentos más representativos, así como un informe de clasificación que incluye precisión, exhaustividad y valor-F para cada categoría, y finalmente las métricas globales (macro y media) para precisión, exhaustividad y valor-F.

4.3 Resultados experimentales

Como resultados globales, las exactitudes obtenidas en los diez experimentos se muestran en la Tabla **4-6**, junto con sus correspondientes barras de error en la Figura **4-3**.

Tabla 4-6: Exactitud de los experimentos. Se indican los valores mínimos, máximos y promedios junto con la desviación estándar calculada a partir de diez repeticiones de cada experimento.

Exp.	Modelo	Min.	Max.	Prom.	SD
1	Base	0.6868	0.7557	0.7129	0.0225
2	Base	0.6839	0.7902	0.7394	0.0303
3	Transferencia	0.7500	0.8592	0.8112	0.0358
4	Transferencia	0.7557	0.8649	0.8316	0.0296
5	Base	0.6983	0.7529	0.7181	0.0184
6	Base	0.6954	0.7730	0.7425	0.0287
7	Transferencia	0.8075	0.8391	0.8207	0.0115
8	Transferencia	0.7931	0.8621	0.8336	0.0169
9	Base	0.7414	0.7931	0.7698	0.0156
10	Transferencia	0.8218	0.8592	0.8365	0.0119

En un primer análisis de los resultados globales, se observa que los modelos presentan una variabilidad significativa en el rendimiento, mostrando diferencias para la exactitud que, en promedio, oscilan entre 0.7129 ± 0.0225 (en el Experimento 1), y 0.8365 ± 0.0119 (en el Experimento 10).

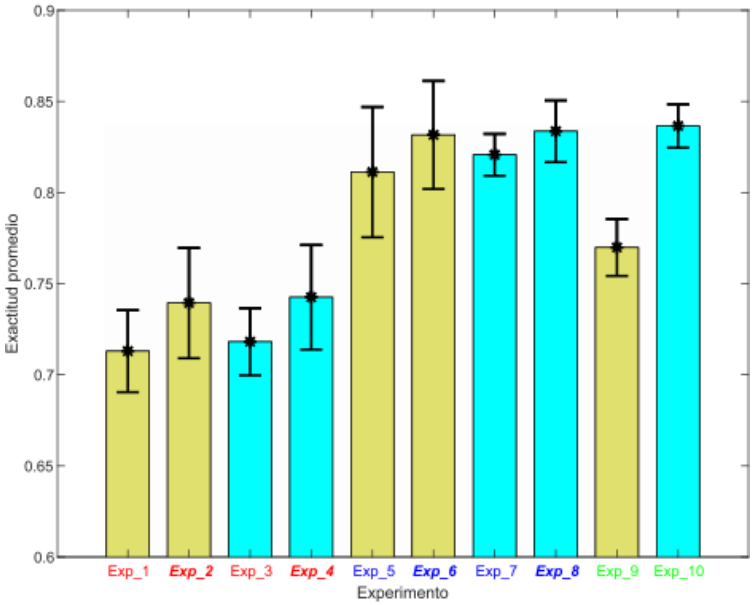


Figura 4-3: Barras de error para la exactitud por experimento. Los experimentos 1, 2, 5, 6, 9 (barras amarillas) corresponden al modelo de referencia y los experimentos 3, 4, 7, 8 y 10 (barras azules) corresponden al modelo de transferencia; los experimentos 1, 2, 3, 4 (leyendas en color rojo) utilizan el conjunto de datos original, los experimentos 5, 6, 7, 8 (leyendas en color azul) utilizan el conjunto de datos garantizado y los experimentos 9, 10 (leyendas en color verde) utilizan el conjunto de datos equilibrado; por último, los experimentos 2, 4, 6, 8 utilizan DA en el entrenamiento.

Las métricas de rendimiento general, incluyendo la precisión macro, la precisión media, la exhaustividad y el valor-F, se presentan en la Tabla **4-7**.

Tabla 4-7: Métricas de rendimiento global por experimento

# Exp.	Macro			Ponderada		
	Precisión	Exhau.	Valor-F	Precisión	Exhau.	Valor-F
1	0.6144	0.5604	0.5759	0.7138	0.7129	0.7076
2	0.6084	0.5732	0.5792	0.7356	0.7394	0.7324
3	0.7137	0.6689	0.6823	0.8114	0.8112	0.8072
4	0.6955	0.6693	0.6767	0.8243	0.8316	0.8249
5	0.6354	0.5705	0.5914	0.7198	0.7181	0.7126
6	0.6804	0.5921	0.6063	0.7475	0.7425	0.7330
7	0.7200	0.6907	0.7004	0.8179	0.8207	0.8168
8	0.7030	0.6741	0.6853	0.8292	0.8336	0.8295
9	0.7118	0.7099	0.7083	0.7699	0.7698	0.7687
10	0.8199	0.7331	0.7526	0.8372	0.8365	0.8354

En la Figura **4-3** se observa que entre los pares de experimentos cuya única diferencia es la aplicación de DA (Experimentos $1-2$, $3-4$, $5-6$, $7-8$), se observan mejoras en el rendimiento medio, pero que no son significativas según el criterio simple de no superposición de intervalos Avg $\pm$ SD. En cuanto al macro valor-F, se puede observar que los mayores valores se logran en los experimentos 7, 9 y 10. Por el contrario, considerando las métricas ponderadas, que podrían ser más útiles dado el desequilibrio entre las clases, destacan los resultados obtenidos en los Experimentos 3, 4, 7, 8 y 10 utilizando el modelo TL, que ofrece valores superiores a 0.8 en el valor-F ponderado. Dado que se utilizaron dos modelos diferentes (línea de base y transferencia), a continuación se analizan con más detalle los resultados experimentales por separado para los dos modelos.

Resultados con el modelo base/referencia

Los experimentos 1, 2, 5, 6 y 9 se desarrollaron con el modelo de referencia, lo que permitió comprobar el funcionamiento y la capacidad de clasificación de los defectos con un modelo basado en CNN con pocas capas sobre las tres variantes del conjunto de datos y, alternativamente, haciendo uso de las técnicas de DA.

Las Figuras: **4-4** y **4-5** muestran la evolución de las exactitudes de los procesos de entrenamiento y de prueba con respecto a las épocas de entrenamiento en los Experimentos 1 y 2, en particular para las ejecuciones en las que se obtuvo la máxima exactitud con el conjunto de prueba (0.7557 y 0.7902 respectivamente), encontrando que la exactitud de entrenamiento es casi 0.1 mayor que la de prueba para el Experimento 1, mostrando que la falta de imágenes perjudica la generalización; en cambio, las curvas en el Experimento 2 son similares, mostrando que la red aprende mejor y es capaz de generalizar un poco más.

Al revisar el informe de clasificación acumulada del Experimento 2 (véase la Tabla **4-8**), se observa que la categoría *stain* es en la que se obtienen los mejores resultados, con un valor-F de 0.8709, mientras que los peores resultados se obtienen en las categorías *other* y *high_ contrast*, donde el número de imágenes es inferior a 10. Esto significa que para algunas de las diez ejecuciones no había imágenes disponibles durante el entrenamiento, lo que afecta notablemente los resultados del clasificador, ya que esta categoría no fue entrenada y, por lo tanto, algunas medidas de rendimiento obtienen valores de 0.0 para dichas clases.

En el Experimento 5, que utiliza el conjunto de datos garantizado con el modelo de referencia, se alcanzó una exactitud media de 0.7181 ± 0.0184, con una pequeña mejora de 0.0052 respecto al Experimento 1. Esta mejora se explica por el hecho de que, en once de las trece categorías, el número de imágenes a entrenar era el mismo que en el experimento anterior, pero en categorías como *other* y *high_ contrast*, la participación de las imágenes está asegurada, garantizando la intervención de al menos una imagen por cada clase. El siguiente paso en el Experimento 6, fue entrenar el modelo con el conjunto de datos garantizado aplicando

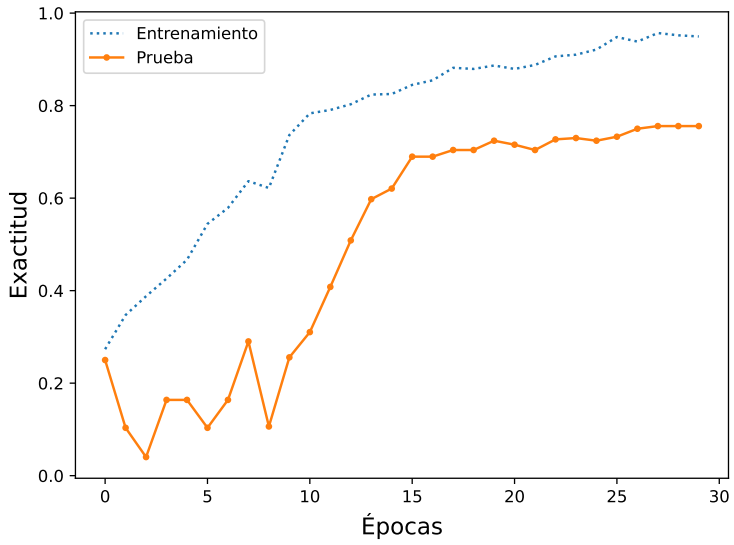


Figura 4-4: Experimento 1: Curva de aprendizaje del modelo base sin aumentación - exactitud vs. épocas.

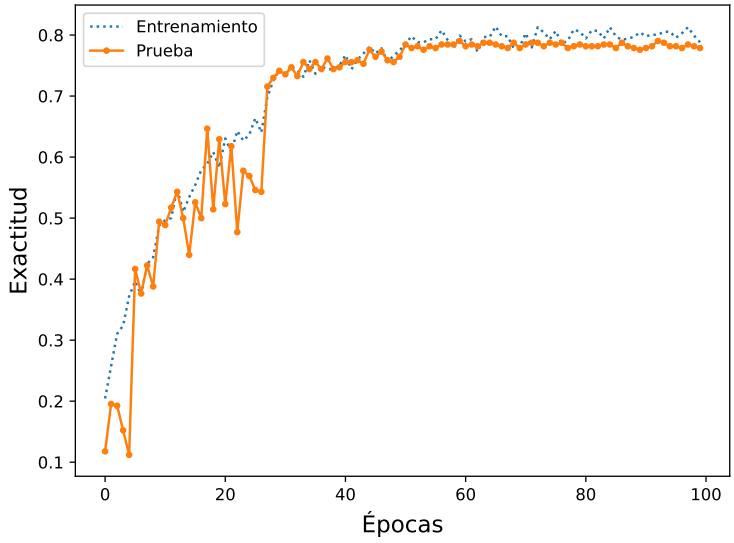


Figura 4-5: Experimento 2: Curva de aprendizaje del modelo base con aumentación - exactitud vs. épocas.

Tabla 4-8: Informe de clasificación acumulado del Experimento 2.

Categoría	Precisión	Exhau.	Valor-F	Soporte
scratch_light	0.6788	0.7598	0.7170	587
scratch_heavy	0.5350	0.5733	0.5535	307
other	0.0000	0.0000	0.0000	7
bubble	0.8861	0.8524	0.8689	210
chip	0.8175	0.8281	0.8228	384
crack	0.7053	0.5630	0.6262	119
excess_silkscreen	0.5417	0.1238	0.2016	105
mark	0.6738	0.6429	0.6580	392
no_defect	0.7667	0.6571	0.7077	35
point	0.7336	0.8441	0.7850	186
point_set	0.6995	0.7367	0.7176	376
high_contrast	0.0000	0.0000	0.0000	8
stain	0.8715	0.8704	0.8709	764

DA, consiguiendo así una exactitud media de 0.7425 ± 0.0287. En las Figuras **4-6** y **4-7** se muestra la evolución de las exactitudes de los procesos de entrenamiento y prueba de los Experimentos 5 y 6, presentada para las ejecuciones en las que se obtuvo la máxima exactitud (0.7529 y 0.7730 respectivamente). Obsérvese que la exactitud del conjunto de entrenamiento es aproximadamente 0.2 mayor que la exactitud del conjunto de prueba para el Experimento 5, lo que demuestra que la falta de imágenes perjudica la generalización y, que en el Experimento 6 las curvas de entrenamiento y prueba se emparejan, mostrando que la red es capaz de aprender y generalizar razonablemente bien.

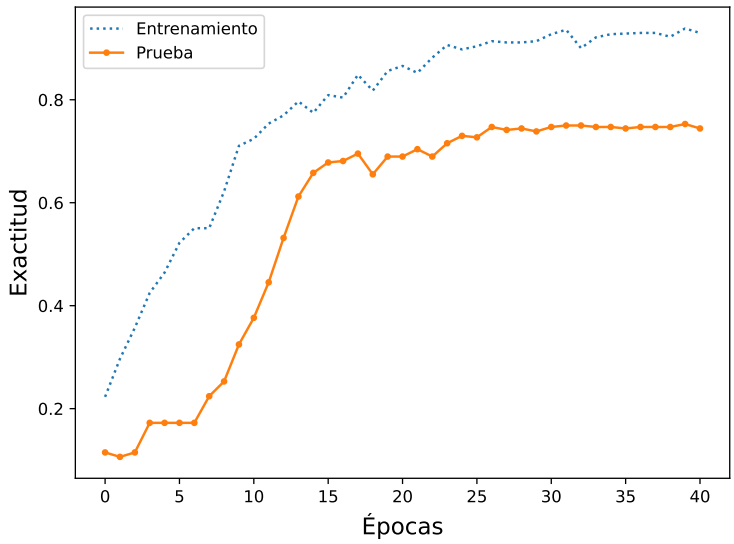


Figura 4-6: Experimento 5: Curvas de aprendizaje del modelo base sin aumentación - exactitud vs. épocas.

Es importante analizar la matriz de confusión acumulada del Experimento 6 que se presenta en la Tabla **4-9**, así como el reporte de clasificación en la Tabla **4-10**, donde se muestran los mejores resultados alcanzados hasta el momento, encontrando que algunas categorías de defectos como *scratch_light* y *scratch_heavy* son propensas a confundirse entre sí, lo que explica sus clasificaciones erróneas: 42 imágenes (7.0 %) de *scratch_light* que se clasifican como *scratch_heavy*, y 71 imágenes (11, 8 %) de *scratch_heavy* que se clasifican como *scratch_light*. Obsérvese también que 49 imágenes de *scratch_light* (8, 2 %) se clasifican como *point_set*, que pueden ser visualmente similares.

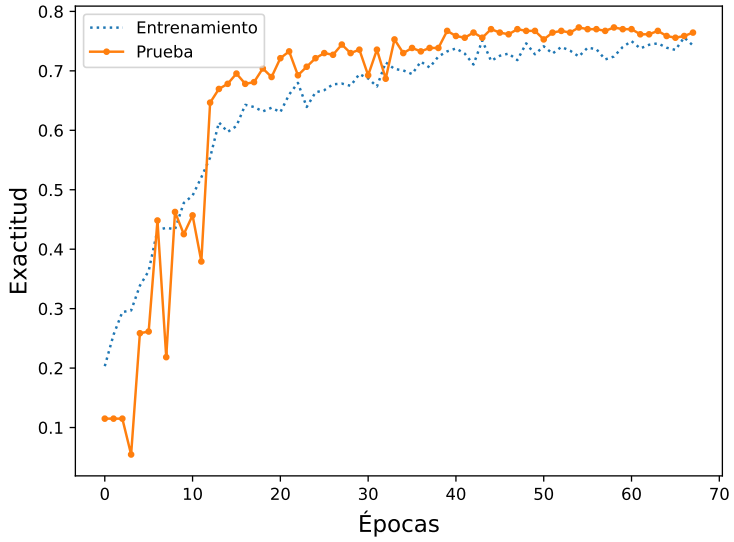


Figura 4-7: Experimento 6: Curvas de aprendizaje del modelo base con aumentación - exactitud vs. épocas.

Tabla 4-9: Matriz de confusión acumulada del Experimento 6

Categoría	0	1	2	3	4	5	6	7	8	9	10	11	12	Total
0 - scratch_light	468	42	0	0	6	0	0	12	0	1	49	0	22	600
1 - scratch_heavy	71	184	0	1	14	5	0	13	0	4	12	0	6	310
2 - other	0	0	0	0	10	0	0	0	0	0	0	0	0	10
3 - bubble	3	0	0	170	0	0	0	12	0	24	1	0	0	210
4 - chip	4	2	0	0	348	9	1	2	7	0	2	0	25	400
5 - crack	2	27	0	0	11	73	0	5	0	0	2	0	0	120
6 - excess_silkscreen	1	3	0	1	16	8	11	32	0	5	4	0	29	110
7 - mark	23	35	0	7	8	0	0	275	0	19	14	1	18	400
8 - no_defect	5	0	0	0	11	0	0	0	23	0	0	0	1	40
9 - point	4	0	0	5	0	0	0	20	0	149	10	2	0	190
10 - point_set	54	8	0	2	3	0	0	9	0	17	270	0	7	370
11 - high_contrast	1	0	0	0	0	0	0	1	0	5	0	3	0	10
12 - stain	33	10	0	0	16	0	0	22	0	0	19	0	610	710

Además, para categorías como *other* y *high_ contrast* se puede observar que con un total de 10 imágenes analizadas en diez experimentos, sólo contaban con una imagen para el entrenamiento y entre una y tres imágenes para la prueba; además, el proceso de DA no garantiza suficientes imágenes en esas categorías, lo que se refleja en los bajos resultados de la métrica combinada valor-F con 0.0 y 0.3750 respectivamente. Las categorías que muestran los mejores resultados en términos de precisión son *excess_ silkscreen* con 0.9167 y *bubble* con 0.914 mientras que, en términos de exhaustividad, son *chip* con 0.87 y *stain* con 0.8592. En cuanto al valor-F, los mejores resultados corresponden a *bubble* con 0.8586 y *stain* con 0.8543, lo que demuestra que el clasificador se comporta bien con esas clases. La mayor diferencia entre precisión y exhaustividad se observa en *excess_ silkscreen* (0.9167 frente a 0.1), lo que indica que el modelo no detecta muy bien esta clase, pero cuando lo hace, su predicción es bastante confiable.

Tabla 4-10: Informe de clasificación acumulado de los Experimentos 6 y 9

	Experimento 6			Experimento 9			
Categoría	Precisión	Exhau.	Valor-F	Precisión	Exhau.	Valor-F	Soporte
scratch_light	0.6996	0.7800	0.7376	0.7320	0.7467	0.7393	600
scratch_heavy	0.5916	0.5935	0.5926	0.6536	0.6452	0.6494	310
other	0.0000	0.0000	0.0000	0.6667	0.6000	0.6316	10
bubble	0.9140	0.8095	0.8586	0.9146	0.8667	0.8900	210
chip	0.7856	0.8700	0.8256	0.8722	0.8700	0.8711	400
crack	0.7684	0.6083	0.6791	0.7500	0.7750	0.7623	120
excess_silkscreen	0.9167	0.1000	0.1803	0.4425	0.4545	0.4484	110
mark	0.6824	0.6875	0.6849	0.7311	0.6050	0.6621	400
no_defect	0.7667	0.5750	0.6571	0.7234	0.8500	0.7816	40
point	0.6652	0.7842	0.7198	0.7285	0.8474	0.7835	190
point_set	0.7050	0.7297	0.7171	0.7538	0.8027	0.7775	370
high_contrast	0.5000	0.3000	0.3750	0.4286	0.3000	0.3529	10
stain	0.8496	0.8592	0.8543	0.8565	0.8662	0.8613	710

Las categorías con peores resultados en precisión y exhaustividad utilizando el Experimento 6 son *other* (con 0.0 y 0.0 respectivamente) y *high_ contrast* (con 0.5 y 0.3 respectivamente). Es importante tener en cuenta que estas clases son las que cuentan con un menor número de imágenes para el entrenamiento, lo que implica que el modelo no puede aprenderlas correctamente. En el Experimento 9, que utiliza el conjunto de datos equilibrado, se obtuvo una exactitud de 0.7698 ± 0.0156 (véase de nuevo la Tabla **4-6**), lo que supone un ligero pero no significativo aumento de 0.0273 con respecto a los mejores resultados del Experimento 6. Un análisis más detallado de la información de la Tabla **4-10** permite hacer las siguientes observaciones: los mejores resultados se obtienen para las categorías *bubble*, *chip* y *stain*, cuyos valores-F son superiores a 0.8, lo que sugiere, por tanto, que pueden clasificarse con fiabilidad. Por el contrario, los valores-F más bajos (0.3529 y 0.4484) corresponden a *high_ contrast* y *excess_ silkscreen*, lo que indica que esas categorías son poco fiables y difíciles de clasificar. Las curvas de aprendizaje de entrenamiento y prueba y la matriz de confusión acumulada para el Experimento 9 se muestran en la Figura **4-8** y en la Tabla **4-11** respectivamente. Se puede observar que el modelo no es lo suficientemente bueno para generalizar la clasificación, ya que la diferencia entre las exactitudes de entrenamiento y de prueba es superior a 0.1; por lo tanto, un conjunto de datos equilibrado no compensa la escasa variedad de imágenes en algunas de las categorías, lo que se corrobora con el valor-F para categorías como *high_ contrast* que sólo alcanza 0.3529. Otro factor importante es la afinidad morfológica entre algunas de las categorías que se observa en la Figura **4-2**, encontrando que las categorías *scratch_ light*, *scratch_ heavy* y *point_ set* tienden a confundirse entre sí, así como con *mark* que morfológicamente comparte rasgos con otras categorías; en este caso, el clasificador acaba discriminando 400 imágenes entre 11 de las 13 categorías disponibles.

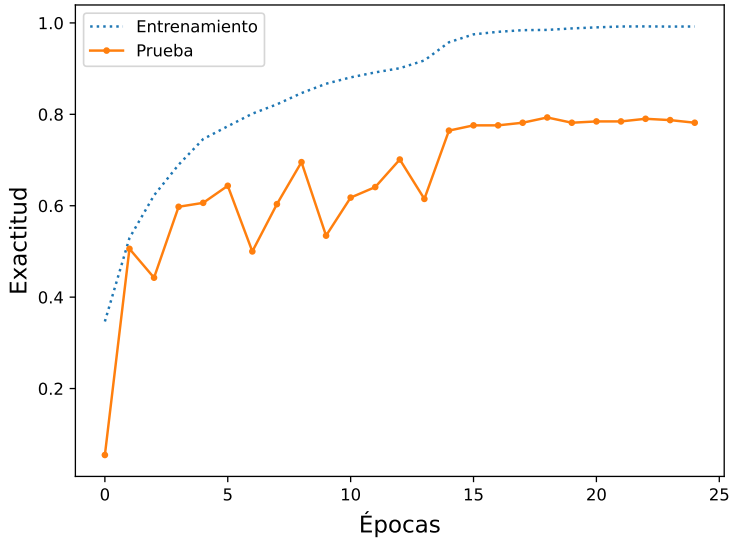


Figura 4-8: Experimento 9: Curvas de aprendizaje del modelo base con aumentación - exactitud vs. épocas.

Tabla 4-11: Matriz de confusión acumulada del Experimento 9.

Categoría	0	1	2	3	4	5	6	7	8	9	10	11	12	Total
0 - scratch_light	448	53	0	0	6	1	3	18	1	4	43	0	23	600
1 - scratch_heavy	61	200	0	0	9	10	4	8	0	3	8	0	7	310
2 - other	0	0	6	0	1	0	2	0	0	0	0	0	1	10
3 - bubble	3	0	0	182	1	0	7	7	0	7	3	0	0	210
4 - chip	9	7	2	0	348	7	6	2	6	0	1	0	12	400
5 - crack	0	10	0	0	9	93	1	6	0	0	0	0	1	120
6 - excess_silkscreen	0	1	1	1	8	3	50	17	0	5	1	0	23	110
7 - mark	19	21	0	9	6	8	28	242	0	27	14	3	23	400
8 - no_defect	3	0	0	0	0	0	0	0	34	0	0	0	3	40
9 - point	1	1	0	6	0	0	0	11	0	161	9	1	0	190
10 - point_set	37	7	0	1	1	2	2	5	0	8	297	0	10	370
11 - high_contrast	0	0	0	0	0	0	0	1	0	6	0	3	0	10
12 - stain	31	6	0	0	10	0	10	14	6	0	18	0	615	710

Resultados del aprendizaje por transferencia para un modelo de aprendizaje residual

Los experimentos 3, 4, 7, 8 y 10 hacen uso de TL sobre un modelo CNN residual. Las exactitudes alcanzadas se presentan en la Tabla **4-6**, donde el Experimento 10 obtuvo los mejores resultados medios con 0.8365 ± 0.0119. En cuanto al Experimento 3, obtuvo una exactitud media de 0.8112 ± 0.0358 para el conjunto de datos original, lo que representa una mejora de 0.0414 con respecto al mejor resultado con el modelo de referencia. Posteriormente, en el Experimento 4, se probó la capacidad de la red utilizando DA, con el objetivo de caracterizar mejor las categorías de defectos, con un resultado de 0.8316 ± 0.0296 para la exactitud. Obsérvese que, según la Figura **4-3**, los resultados del uso de DA en los Experimentos 3 y 4 no se solapan con los resultados de los Experimentos 1 y 2; por tanto, puede decirse que el uso de DA es significativamente útil. Los resultados de clasificación del Experimento 4 se ilustran con la matriz de confusión acumulada en la Tabla **4-12**, y el correspondiente informe de clasificación en la Tabla **4-13**. Según la matriz de confusión acumulada, las categorías *scratch_light*, *scratch_heavy*, así como *point* y *point_set* siguen confundiéndose. Las categorías con altos valores de exhaustividad y valor-F, superiores a 0.8, como en las categorías *scratch_light*, *bubble*, *chip*, *crack*, *no_defect*, *point*, *point_set* y *stain*, muestran que éstas pueden ser clasificadas con buena fiabilidad, mientras que en las categorías *other* y *high_contrast*, que tienen un bajo número de imágenes, se puede apreciar que no está garantizada la disponibilidad de más de una imagen para el entrenamiento, lo que se refleja en un valor de exhaustividad nulo para las dos categorías.

Tabla 4-12: Matriz de confusión acumulada del Experimento 4.

Categoría	0	1	2	3	4	5	6	7	8	9	10	11	12	Total
0 - scratch_light	520	32	0	0	4	0	0	14	0	0	32	0	13	615
1 - scratch_heavy	58	224	0	0	0	2	0	12	0	3	10	0	5	314
2 - other	0	0	0	0	7	0	0	0	0	0	0	0	0	7
3 - bubble	1	0	0	171	2	0	0	12	0	8	2	0	1	197
4 - chip	2	0	0	0	362	5	4	0	2	0	2	0	3	380
5 - crack	7	8	0	0	4	103	1	2	0	0	0	0	0	125
6 - excess_silkscreen	0	6	0	0	15	7	31	27	0	1	3	0	29	119
7 - mark	16	12	0	7	1	4	11	316	0	11	15	0	15	408
8 - no_defect	0	0	0	0	5	0	0	0	38	0	0	0	4	47
9 - point	1	0	0	2	0	0	0	18	0	149	7	1	0	178
10 - point_set	11	5	0	0	0	0	0	3	0	7	341	0	6	373
11 - high_contrast	3	0	0	0	0	0	0	0	0	6	0	0	0	9
12 - stain	22	3	0	0	2	2	7	8	0	0	25	0	639	708

Tabla 4-13: Informe de clasificación acumulado del Experimento 4.

Categoría	Precisión	Exhau.	Valor-F	Soporte
scratch_light	0.8112	0.8455	0.8280	615
scratch_heavy	0.7724	0.7134	0.7417	314
other	0.0000	0.0000	0.0000	7
bubble	0.9500	0.8680	0.9072	197
chip	0.9005	0.9526	0.9258	380
crack	0.8374	0.8240	0.8306	125
excess_silkscreen	0.5741	0.2605	0.3584	119
mark	0.7670	0.7745	0.7707	408
no_defect	0.9500	0.8085	0.8736	47
point	0.8054	0.8371	0.8209	178
point_set	0.7803	0.9142	0.8420	373
high_contrast	0.0000	0.0000	0.0000	9
stain	0.8937	0.9025	0.8981	708

Para el Experimento 7, se probó el conjunto de datos garantizado para comparar su comportamiento con el modelo basado en TL, obteniendo una exactitud media de 0.8207 ± 0.0115 que es 0.0095 mejor que

la obtenida con el conjunto de datos original. Este resultado se explica debido a que sólo debería haber una mejora en aquellas categorías que podrían quedarse sin imágenes como *other* y *high_contrast*, que tienen muy pocas imágenes; sin embargo, a pesar de garantizar que esas categorías estarán representadas en el conjunto de entrenamiento, no contribuyen mucho a los resultados globales. En el Experimento 8, se aplicó DA en el conjunto de datos garantizado, obteniendo una exactitud media de 0.8336 ± 0.0169 (ver Tabla **4-6**), que es 0.0013 mejor que cuando los datos no están aumentados (Experimento 7). Sin embargo, según la Figura **4-3**, el uso de DA no es suficiente, ya que algunas repeticiones del Experimento 8 no alcanzan la media del Experimento 7, lo que confirma que al proporcionar imágenes adicionales durante el entrenamiento, los resultados sólo mejoran ligeramente. La matriz de confusión acumulada y el correspondiente informe de clasificación para el Experimento 8 se presentan en las Tablas **4-14** y **4-15**, permitiendo observar que: a pesar de que las categorías *other* y *high_contrast* disponen de imágenes para el entrenamiento en todas las repeticiones del experimento, los valores de exhaustividad son 0.0 ya que no es posible garantizar que el aumento en el número de las imágenes proporcione suficientes imágenes para mejorar significativamente el rendimiento global del clasificador. Persisten las confusiones entre las categorías *scratch_light*, *scratch_heavy* y *point_set*. Las categorías como *chip* y *stain* alcanzan valores de exhaustividad superiores a 0.9, lo que aumenta la fiabilidad del clasificador para estas categorías. Todas las categorías, excepto *other*, *excess_silkscreen* y *high_contrast*, muestran resultados altos y similares en precisión y exhaustividad, lo que indica que el modelo funcionará bien para clasificar las imágenes de estas categorías.

Tabla 4-14: Matriz de confusión acumulada del Experimento 8.

Categoría	0	1	2	3	4	5	6	7	8	9	10	11	12	Total
0 - scratch_light	514	27	0	0	4	1	0	14	0	0	26	0	14	600
1 - scratch_heavy	59	220	0	0	0	7	1	12	0	0	5	0	6	310
2 - other	0	0	0	0	9	0	1	0	0	0	0	0	0	10
3 - bubble	2	0	0	180	0	0	1	13	0	9	4	0	1	210
4 - chip	8	0	0	0	374	5	2	0	3	0	3	0	5	400
5 - crack	6	7	0	0	5	98	0	4	0	0	0	0	0	120
6 - excess_silkscreen	0	1	0	0	6	4	48	22	0	5	3	0	21	110
7 - mark	16	4	0	5	3	0	12	313	0	13	14	0	20	400
8 - no_defect	0	0	0	0	3	0	0	0	31	0	0	0	6	40
9 - point	0	0	0	5	0	0	0	18	0	155	10	2	0	190
10 - point_set	21	2	0	0	1	2	0	4	0	7	321	0	12	370
11 - high_contrast	0	0	0	0	0	0	0	1	0	9	0	0	0	10
12 - stain	24	3	0	0	1	0	6	12	0	0	17	0	647	710

Tabla 4-15: Informe de clasificación acumulado de los Experimentos 8 y 10.

	Experimento 8			Experimento 10			
Categoría	Precisión	Exhau.	Valor-F	Precisión	Exhau.	Valor-F	Soporte
scratch_light	0.7908	0.8567	0.8224	0.8376	0.8250	0.8312	600
scratch_heavy	0.8333	0.7097	0.7666	0.8147	0.7516	0.7819	310
other	0.0000	0.0000	0.0000	1.0000	0.2000	0.3333	10
bubble	0.9474	0.8571	0.9000	0.9118	0.8857	0.8986	210
chip	0.9212	0.9350	0.9280	0.9474	0.9450	0.9462	400
crack	0.8376	0.8167	0.8270	0.7744	0.8583	0.8142	120
excess_silkscreen	0.6761	0.4364	0.5304	0.5377	0.5182	0.5278	110
mark	0.7579	0.7825	0.7700	0.7807	0.7475	0.7637	400
no_defect	0.9118	0.7750	0.8378	0.9429	0.8250	0.8800	40
point	0.7828	0.8158	0.7990	0.7824	0.7947	0.7885	190
point_set	0.7965	0.8676	0.8305	0.7650	0.8622	0.8107	370
high_contrast	0.0000	0.0000	0.0000	0.6667	0.4000	0.5000	10
stain	0.8839	0.9113	0.8974	0.8979	0.9169	0.9073	710

Por último, en el Experimento 10 que utiliza el conjunto de datos equilibrado, se obtiene una exactitud media de 0.8365 ± 0.0119 (Tabla **4-6**), la más alta alcanzada, que es 0.0029 superior a la del Experimento 8, con el conjunto de datos garantizado. Del análisis de la matriz de confusión (Tabla **4-16**) y del informe de clasificación (Tabla **4-15**) del Experimento 10 se desprenden las siguientes observaciones:

Tabla 4-16: Matriz de confusión acumulada del Experimento 10.

Categoría	0	1	2	3	4	5	6	7	8	9	10	11	12	Total
0 - scratch_light	495	35	0	0	3	0	2	11	0	0	43	0	11	600
1 - scratch_heavy	43	233	0	0	3	13	2	7	0	0	6	0	3	310
2 - other	0	0	2	0	5	0	3	0	0	0	0	0	0	10
3 - bubble	0	0	0	186	2	0	3	14	0	3	2	0	0	210
4 - chip	9	0	0	0	378	3	3	1	2	0	0	0	4	400
5 - crack	4	2	0	0	4	103	2	5	0	0	0	0	0	120
6 - excess_silkscreen	0	0	0	0	4	4	57	18	0	6	4	0	17	110
7 - mark	10	8	0	10	0	2	18	299	0	21	14	1	17	400
8 - no_defect	1	0	0	0	0	0	0	0	33	0	0	0	6	40
9 - point	1	2	0	7	0	0	1	18	0	151	9	1	0	190
10 - point_set	13	4	0	0	0	5	4	2	0	7	319	0	16	370
11 - high_contrast	0	0	0	1	0	0	0	0	0	5	0	4	0	10
12 - stain	15	2	0	0	0	3	11	8	0	0	20	0	651	710

Todas las categorías, excepto *other*, *excess_ silkscreen* y *high_ contrast*, muestran resultados superiores a 0.75 en precisión, exhaustividad y valor-F. Categorías como *other* y *high_ contrast*, que tenían valores de exhaustividad nulos en experimentos anteriores, alcanzan ahora valores de 0.3333 y 0.5 respectivamente, dando la posibilidad de clasificar correctamente algunas de sus imágenes. Se mantienen las apreciaciones en cuanto a la afinidad de algunas categorías como *scratch_ light*, *scratch_ heavy* y *point_ set*. La Figura **4-9**, muestra las curvas de entrenamiento y de prueba para el Experimento 10. Se puede observar que, a pesar de alcanzar una exactitud de 1.0 en el entrenamiento, el modelo no es lo suficientemente bueno como para generalizar todo, ya que el valor máximo alcanzado con el conjunto de prueba es de 0.8365 ± 0.0119. Otro factor importante que limita los buenos resultados de las pruebas es la afinidad morfológica que existe entre algunas de las categorías, encontrando que *scratch_ light*, *scratch_ heavy* y *point_ set* comparten muchas de sus características físicas. También es importante destacar la similitud de rendimiento de los experimentos que incluyen DA; véanse de nuevo los resultados de los experimentos 3, 4, 7, 8 y 10 en la Tabla **4-6** y la Figura **4-3**. Obsérvese que las medias de exactitud oscilan entre 0.8112 en el Experimento 3 y 0.8365 en el Experimento 10.

Fiabilidad de las etiquetas de clase originales

Con el fin de verificar la fiabilidad de las etiquetas de clase que estaban previamente asignadas a cada una de las imágenes, se llevó a cabo una revisión manual de las imágenes y de sus etiquetas de clase originales, comprobando que aunque muchas de las imágenes con defectos quedan bien clasificadas, tal como se aprecia en las confidencias de las etiquetas predichas para algunas imágenes en la Figura **4-10**, una gran cantidad de ellas contenían más de un defecto o podían estar mal etiquetadas. La Figura **4-11** muestra grupos de imágenes ejemplares que son visualmente muy similares y, por tanto, propensas a ser fácilmente confundidas o incluso etiquetadas de forma ambigua o incorrecta por los expertos (analizar la similitud por cada fila).

Otra comprobación de la ambigüedad y dificultad encontrada con el conjunto de datos se presenta en la Figura **4-12**, donde se muestran las confidencias de las etiquetas predichas para algunas imágenes utilizando la mejor ejecución del Experimento 10. Obsérvese que una imagen de la clase *scratch_ heavy* se predijo principalmente como *crack* (categoría 5) con una probabilidad posterior de aproximadamente 1.0; de forma similar, una imagen de la categoría *mark* se predice principalmente como *bubble* (categoría 3) con una probabilidad posterior de 1.0; y en las siguientes figuras se muestran casos similares entre las clases *bubble* y

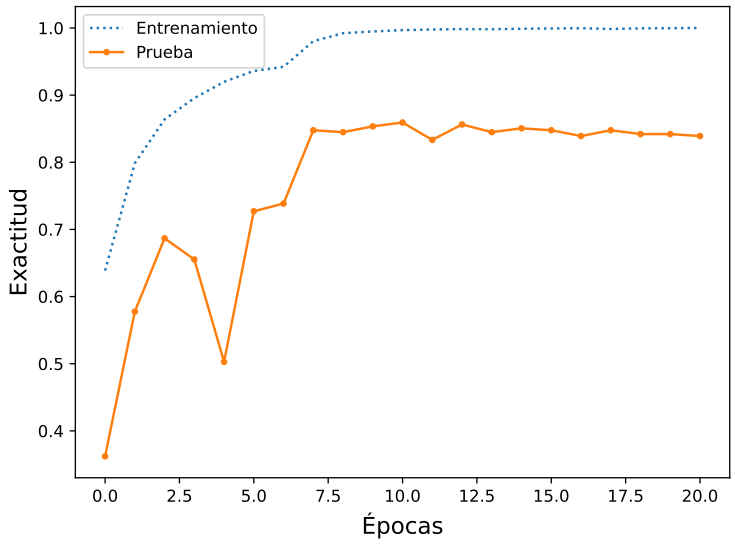


Figura 4-9: Experimento 10: Curvas de aprendizaje del modelo por transferencia con aumentación - exactitud vs. épocas.

mark, *excess_silkscreen* y *stain*, *mark* y *stain*, y finalmente entre *scratch_light* y *point_set*, demostrando la gran afinidad entre ellas, así como las posibles inconsistencias en el etiquetado original.

4.4 Conclusiones

En este capítulo se presentó un análisis comparativo de varias estrategias basadas en CNN para clasificar defectos en láminas de vidrio. La comparación incluyó diez experimentos que consideraron diferentes opciones de preprocesamiento del conjunto de datos y arquitecturas de CNN, así como técnicas de aprendizaje profundo como TL, RL, DA y ajustes de (hiper)parámetros; además, se utilizaron diversas métricas para medir e interpretar los rendimientos resultantes. Los mejores resultados se obtuvieron cuando se utilizaron redes basadas en RL, en particular la arquitectura *"InceptionResNetV2"* junto con la aplicación combinada de las técnicas TL y DA, alcanzando una exactitud media de 0.8365 ± 0.0119, una precisión de 0.8372, una exhaustividad de 0.8365 y un valor-F combinado de 0.8354. Estos rendimientos se consideran aceptables dada la naturaleza desafiante del conjunto de datos de GI y las posibles ambigüedades encontradas en el etiquetado original de las imágenes. El análisis exhaustivo para comprender el comportamiento de las soluciones también incluyó matrices de confusión, informes de clasificación detallados, curvas de aprendizaje de entrenamiento/prueba y confidencias de las etiquetas de clase predichas. Finalmente, quedó claro que la afinidad morfológica entre las categorías, las posibles ambigüedades encontradas en las etiquetas originales y la baja representación de algunas de las categorías de defectos son inconvenientes que dificultan en gran medida encontrar una mejor solución al problema.

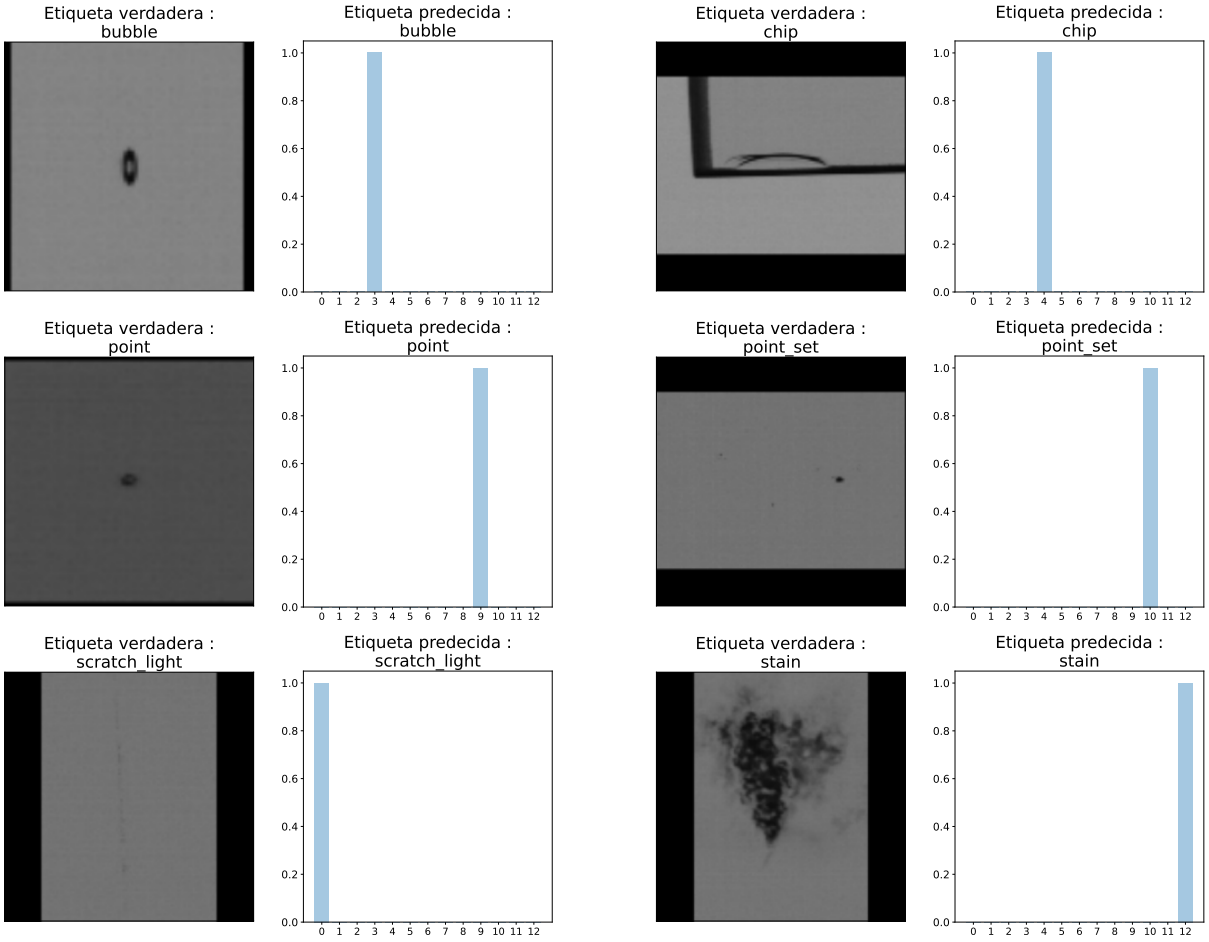
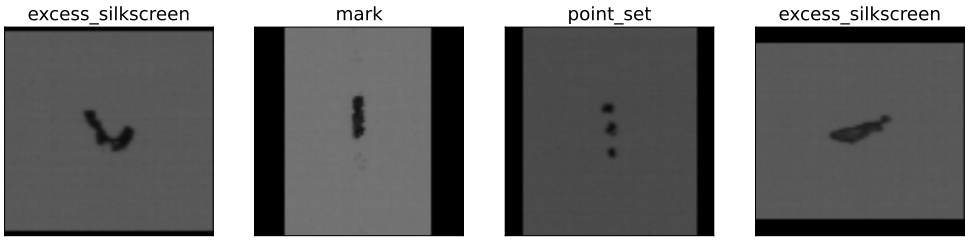
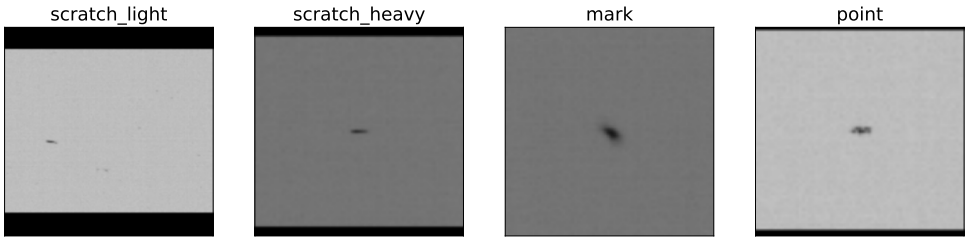


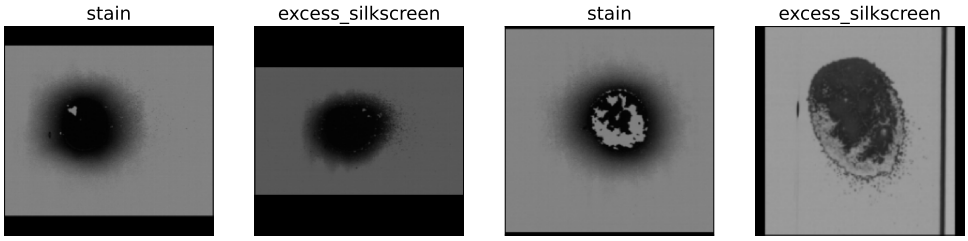
Figura 4-10: Probabilidades posteriores/confidencias de las etiquetas de clase predichas en la última ejecución del Experimento 10, para defectos bien clasificados. Las correspondencias entre los nombres de las categorías y los índices de las etiquetas predichas son las mismas que se indicaron en la Tabla **4-3**.



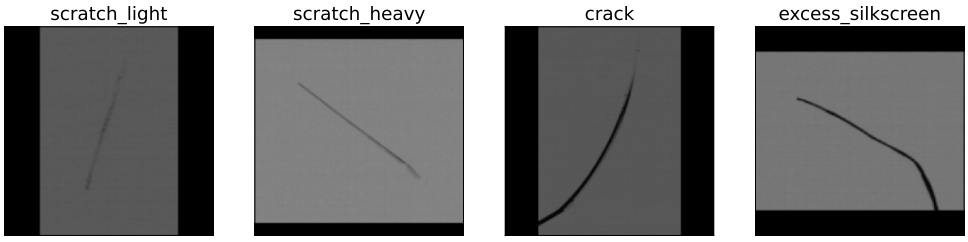
(a) Imágenes con defectos que parecen conjuntos de puntos gruesos



(b) Imágenes con defectos que parecen puntos aislados



(c) Imágenes con defectos que parecen manchas grandes y redondeadas



(d) Imágenes con defectos que parecen rayones

Figura 4-11: Ejemplos de imágenes agrupados por fila con defectos similares y con etiquetado original ambiguo.

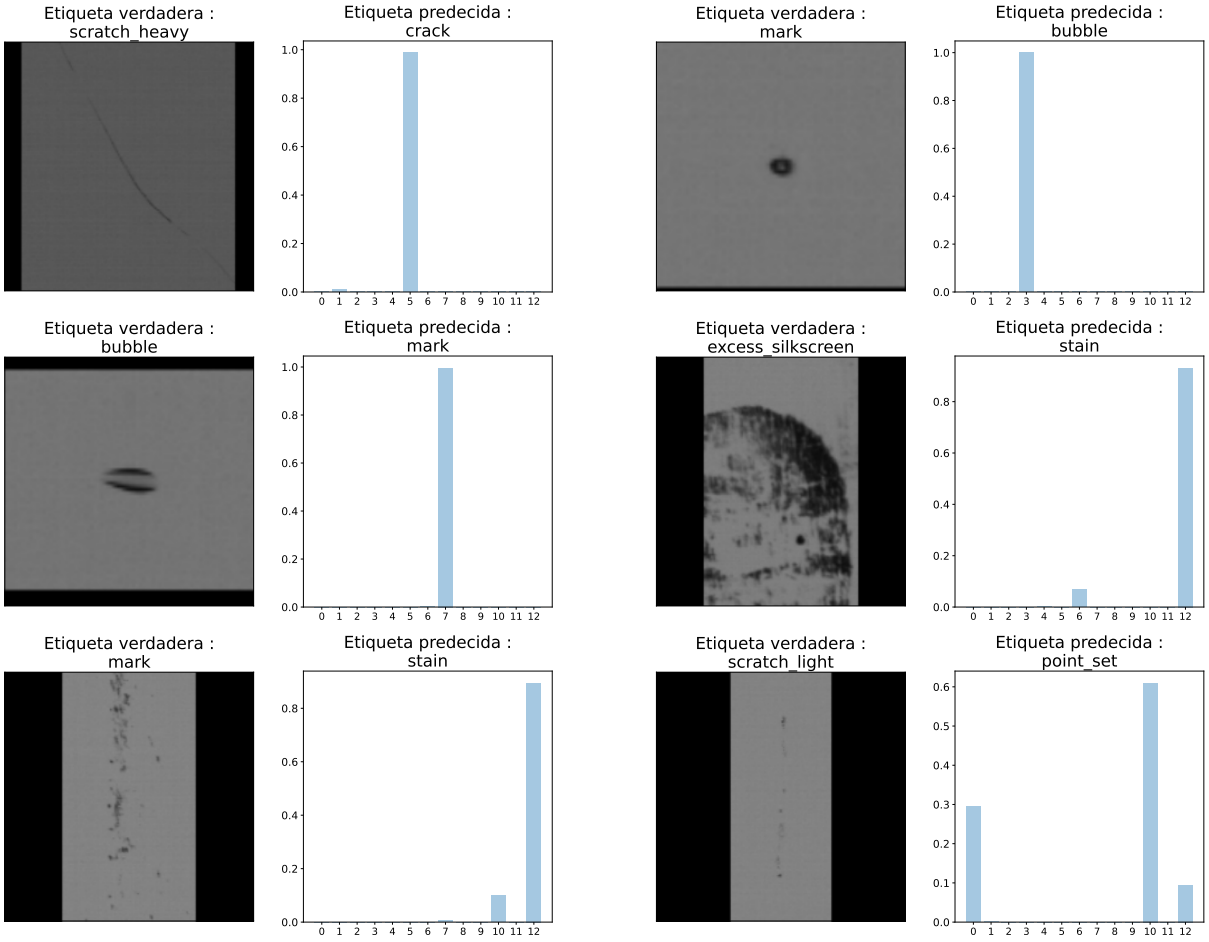


Figura 4-12: Probabilidades posteriores/confidencias de las etiquetas de clase predichas en la última ejecución del Experimento 10. Las correspondencias entre los nombres de las categorías y los índices de las etiquetas predichas son las mismas que se indicaron en la Tabla **4-3**.

5 Clasificación de defectos en dulces

Este capítulo se basa en la ponencia:

- Eduardo Villegas-Jaramillo, y Mauricio Orozco-Alzate *''Candy classification using convolutional neural networks, data augmentation and transfer learning: Application and a new public dataset''* [Villegas-Jaramillo and Orozco-Alzate. 2023a].
  Presentada en ''The IEEE 13th International Conference on Pattern Recognition Systems (ICPRS-2023)'', llevada a cabo entre el 4 y el 7 de julio de 2023 en la ciudad de Guayaquil - Ecuador.

5.1 Introducción

La confitería es una de las principales industrias establecidas en Manizales, Colombia. En esta ciudad, la empresa más grande de ese sector obtuvo —según [Becerra Elejalde. 2020]— un ingreso total de $358.216 millones de COP en el año 2021; es decir, cerca de 90 millones de USD a la tasa de cambio de finales de ese año. Tal cantidad de ingresos ilustra la importancia de esta industria para la producción anual de bienes en una ciudad latinoamericana de tamaño medio como la antes mencionada.

Los dulces de maní recubiertos con chocolate son uno de los muchos tipos de productos que se fabrican en esta industria. Se comercializan en diferentes presentaciones y están hechos de maní recubierto con chocolate en varios colores (rojo, verde, azul, café, naranja y amarillo), con formas elipsoidales y diámetros entre 8 y 12 mm. Realizar el control de calidad de este tipo de producto es una tarea difícil, dados los diferentes problemas que pueden surgir en el momento de la fabricación como: grietas y resquebrajados en la cubierta, formas irregulares y tamaños fuera de las tolerancias previamente definidas.

Ante esta problemática, este trabajo presenta una aplicación de técnicas de aprendizaje profundo para la clasificación de dulces de maní recubiertos de chocolate, discriminando entre los que cumplen con las condiciones de calidad (*No defectuosos*) y los que no cumplen (*Defectuosos*). Se han elegido técnicas de DL, en particular basadas en CNN, por considerarse el estado del arte para la clasificación de imágenes. Además, el conjunto de datos desarrollado y utilizado para los experimentos se ha hecho público junto con el artículo, para fomentar nuevos trabajos e investigaciones [Villegas-Jaramillo. 2023]. Es importante tener en cuenta que para los fines de este trabajo, sólo se considera la apariencia de los dulces, sin incluir problemas relacionados con el sabor, el aroma, la textura o la composición.

Este trabajo consta de diferentes etapas, a saber: i) comienza con la adquisición de una muestra de los productos, comprobándose que los supuestamente no defectuosos son los que se venden como dulces de primera categoría mientras que, por el contrario, los que no cumplen algunas de las condiciones de calidad (es decir, los defectuosos) se venden como dulces de segunda categoría y, por tanto, más baratos. ii) A continuación, se toman fotografías de los productos y se construye el conjunto de datos. iii) Las imágenes se

etiquetan manualmente de acuerdo con un examen visual de las muestras y los tipos de defectos observados. iv) Se crean variantes del conjunto de datos teniendo en cuenta opciones como el modo de color (color verdadero o escala de grises), así como la opción de utilizar o no DA. v) A continuación, se desarrollan dos modelos de CNNs basados en TL y DA para resolver el problema y, vi) se prueban los dos modelos con las variantes del conjunto de datos estimando una serie de métricas como la exactitud, la precisión, la exhaustividad, el valor-F, las matrices de confusión y algunas medidas de rendimiento adicionales.

5.2 Configuración experimental

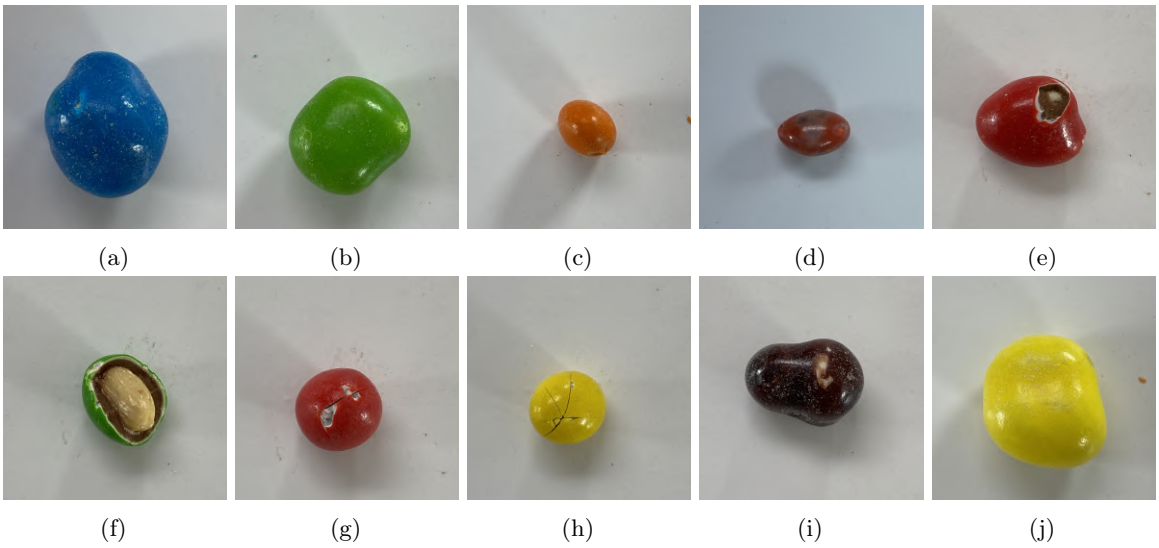
5.2.1 Conjunto de datos

Se consideró una colección de 200 dulces de maní recubiertos con chocolate de colores. Se tomaron dos imágenes por dulce para construir un conjunto de datos equilibrado de 400 imágenes con una resolución de 512×512 píxeles, de forma tal que 200 de ellas pertenecen a la clase *''No defectuosa''* (véase Figura **5-1**) y las 200 restantes pertenecen a la clase *''Defectuosa''*. En esta última clase, se pueden encontrar diferentes tipos de defectos, como grietas, deformaciones y problemas de tamaño, como se muestra en la Figura **5-2**.

Figura 5-1: Ejemplos de dulces de maní etiquetados como *No Defectuosos*.

Una vez definido el conjunto de datos con su respectiva proporción por clase y considerando la pequeña cantidad de imágenes resultantes, se procedió a desarrollar un proceso de DA, en el cual se generaron 5 nuevas imágenes a partir de cada imagen original mediante el uso de transformaciones como escala, espejo, rotación y traslación. Como resultado, se obtienen 2000 imágenes: 1000 por cada categoría, como se muestra en la Tabla **5-1**.

Un ejemplo del proceso de DA para una imagen etiquetada como *Defectuosa* se muestra en la Figura **5-3**. La imagen original se muestra en la parte superior de la figura y las cinco imágenes resultantes del proceso de aumento en la parte inferior.



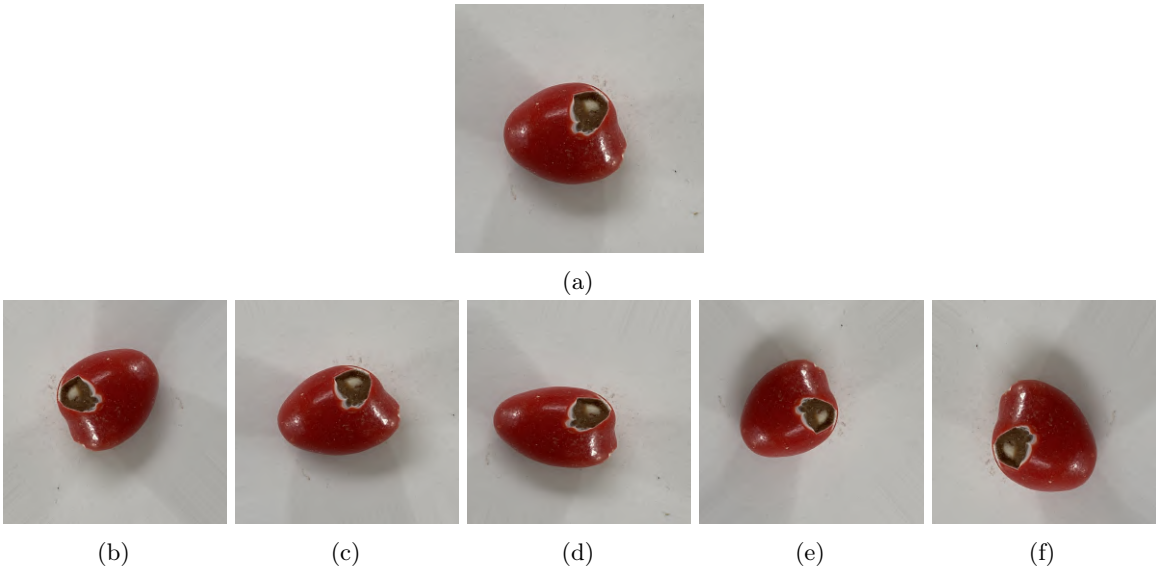
(a) (b) (c) (d) (e)

(f) (g) (h) (i) (j)

Figura 5-2: Ejemplos de dulces de maní etiquetados como *Defectuosos* donde (a)—(b) son demasiado grandes, (c)—(d) son demasiado pequeños, (e)—(f) están resquebrajados, (g)—(h) tienen grietas, o (i)—(j) están deformados.

Tabla 5-1: Número de imágenes del conjunto de datos según las etiquetas de clase, sin DA (Original) o con DA.

Categoría	No-defectuoso	Defectuoso	Total
Original	200	200	400
Aumentado	1000	1000	2000



(a)

(b) (c) (d) (e) (f)

Figura 5-3: Ejemplos del proceso de aumento de datos para una imagen etiquetada como *Defectuosa*, donde (a) corresponde a la imagen original, y (b)—(f) son las imágenes resultantes tras aplicar el proceso de DA.

5.2.2 Adquisición de las imágenes

Las imágenes del conjunto de datos se obtuvieron utilizando la cámara de un teléfono móvil iPhone 11, con un zoom 3X y una distancia focal de 14 cm. En cuanto a la iluminación, se colocaron de forma equidistante tres lámparas con bombillas LED de 14 vatios y 1200 lúmenes a 30 cm de los dulces, como puede verse en la Figura **5-4**.

Figura 5-4: Montaje para la adquisición de las imágenes con el soporte para el teléfono móvil, una muestra de un dulce y la disposición de la iluminación.

5.2.3 Plataformas de computación y métricas de desempeño

Los experimentos se ejecutaron en un servidor DELL con sistema operativo Ubuntu versión 18.04, donde se utilizaron como librerías de soporte Python 3.8.5, con Keras 2.3.1 y TensorFlow 2.4.1.

Se consideraron dos CNNs diferentes, con todas las variantes de la configuración experimental y diez permutaciones diferentes —para un proceso de entrenamiento y prueba repetidos— en cada uno de los conjuntos de datos. Se estimaron diferentes métricas de rendimiento [Bramer. 2020, Chap. 12] para los modelos de clasificación, incluyendo exactitud, precisión, exhaustividad, valor-F y matrices de confusión.

5.3 Modelos de clasificación

Se desarrollaron dos modelos diferentes basados en CNNs para resolver el problema de clasificación de los dulces. El primero es un modelo básico de 11 capas con combinaciones de capas convolucionales, de agrupación, de aplanamiento y totalmente conectadas, teniendo en la última etapa un clasificador para dos categorías. La implementación del algoritmo para definir el modelo básico se muestra en el Código 5.1.

Código 5.1: Código fuente en Python/Keras para el modelo de clasificación basado en CNN básico.

```
def create_model_1(input_shape):
    model = Sequential([
        Conv2D(filters=48, kernel_size=3, input_shape=input_shape, activation='relu', padding='
            SAME'),
        MaxPooling2D(pool_size=2),
        Conv2D(filters=32, kernel_size=3, activation='relu', padding='SAME'),
```

```
        MaxPooling2D(pool_size=2),
        Conv2D(filters=32, kernel_size=3, activation='relu', padding='SAME'),
        MaxPooling2D(pool_size=2),
        Conv2D(filters=32, kernel_size=3, activation='relu', padding='SAME'),
        MaxPooling2D(pool_size=2),
        Flatten(),
        Dense(units=512, activation='relu'),
        Dense(units=num_classes, activation='softmax'),
    ])
    return model
```

Para el segundo modelo, basado en la estrategia de TL, se realizaron pruebas preliminares con las redes *''inceptionResNetV2''*, *''VGG16''*, *''ResNet50V2''* y *''ResNet152V2''* [Abubakar et al.. 2020; Szegedy et al.. 2016; Xiao et al.. 2018]. Los mejores resultados observados en las pruebas exploratorias se obtuvieron con la red *''ResNet50V2''*; por tanto, ese modelo fue el seleccionado para resolver el problema, el cual había sido previamente entrenado con imágenes del conjunto de datos *''ImageNet''*. Las últimas capas del modelo fueron sustituidas por un clasificador para las dos categorías del problema. La implementación del algoritmo para la creación del modelo resultante basado en TL se muestra en el Código 5.2.

Código 5.2: Código fuente en Python/Keras para el modelo de clasificación basado en CNN que usa TL.

```
def create_model_2(input_shape):
    base_model = tf.keras.applications.ResNet50V2(include_top=False, weights="imagenet",
        input_tensor=None, input_shape=input_shape, pooling=None, classes=1000,
        classifier_activation="softmax")
    base_model.trainable = False
    model = base_model.output
    x = GlobalAveragePooling2D()(model)
    x = BatchNormalization()(x)
    top_dropout_rate = 0.2
    x = Dropout(top_dropout_rate)(x)
    outputs = Dense(num_classes, activation= 'softmax')(x)
    return Model(inputs = base_model.input, outputs = outputs)
```

La compilación de los dos modelos se realizó utilizando los siguientes parámetros: como optimizador ''rmsprop'', para la pérdida ''categorical_crossentropy'' y finalmente como metrica ''accuracy''. El proceso de entrenamiento se realizó, para todos los experimentos, con un máximo de 30 épocas y con un parámetro de parada temprana (EarlyStopping) de 10 si la pérdida en el proceso de validación no mejora.

5.4 Experimentos

Se realizaron los experimentos de clasificación utilizando los dos modelos de CNN para cada una de las variantes del conjunto de datos. Además, se consideraron 10 permutaciones del conjunto de datos con el fin de promediar los resultados dada la naturaleza estocástica de los modelos CNN, permitiendo así medir la capacidad de solución de cada una de las combinaciones tanto del conjunto de datos como del modelo. El diagrama de las opciones experimentales se muestra a continuación en la Figura **5-5**.

Cada conjunto de datos se dividió en tres partes (entrenamiento, prueba y validación) de la siguiente manera: se utilizó el 65 % del total de las imágenes para el entrenamiento, 15 % para validación, y el 20 % restante para las pruebas. En cifras absolutas, estas particiones de las imágenes corresponden —en cada permutación para el conjunto de datos sin DA— a 260 imágenes para el entrenamiento, 60 para la validación y 80 para las pruebas. Del mismo modo, para el conjunto de datos con DA, se seleccionaron 1300 imágenes para el entrenamiento, 300 para la validación y 400 para las pruebas.

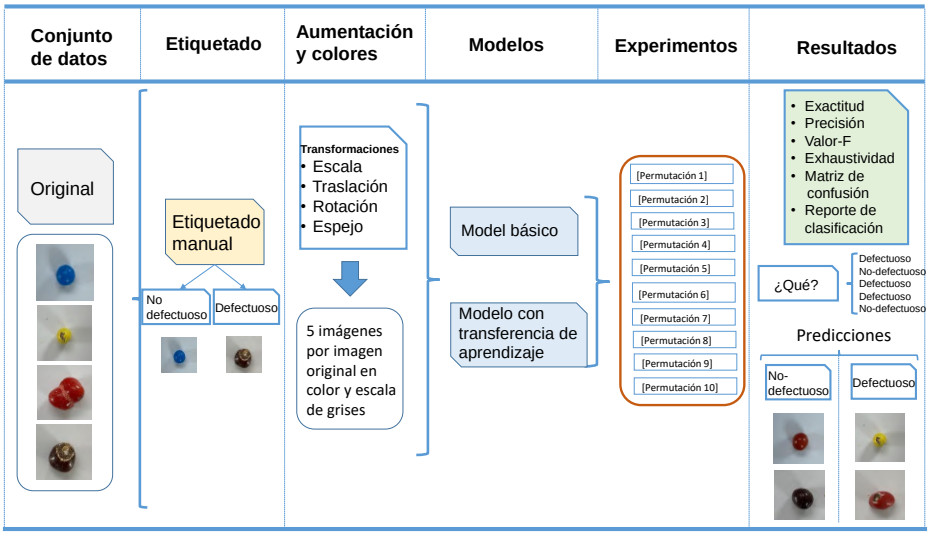


Figura 5-5: Diagrama de experimentos que presenta las diferentes opciones en términos de modelo CNN, modo de color (color verdadero o escala de grises) y uso de DA.

5.5 Resultados y discusiones

Una primera evaluación consistió en explorar la influencia del modo de color de las imágenes en el proceso de clasificación. Para ello, se realizaron experimentos preliminares utilizando imágenes en color y en escala de grises, sin que se observara una diferencia significativa entre utilizar un modo u otro. Dado que los resultados obtenidos utilizando el modo color fueron ligeramente mejores y debido a las limitaciones de espacio y procesamiento, todos los resultados que se presentan a continuación corresponden a los experimentos con imágenes en color.

Los resultados se muestran en función de los dos modelos de CNN y para las dos combinaciones consideradas del conjunto de datos, a saber, para las imágenes originales (es decir, sin DA) y para el conjunto de datos aumentado (siempre utilizando las imágenes en color). Para cada una de las cuatro combinaciones de CNN y el conjunto de datos, se realizaron los respectivos análisis y comparaciones con un amplio conjunto de métricas de clasificación.

5.5.1 Resultados con el modelo CNN básico

El modelo básico pretende determinar la capacidad de solución del problema de clasificación de dulces cuando se utiliza un modelo basado en CNN con pocas capas y sin utilizar TL.

Resultados sin DA - conjunto de datos original Los resultados obtenidos con el modelo básico utilizando el conjunto de datos original conformado por 400 imágenes, se pueden apreciar en la correspondiente matriz de confusión acumulada para 10 ejecuciones (ver Tabla **5-2**). A partir de esta tabla, se pueden estimar las siguientes medidas globales: una exactitud de 0.8475, un TP-rate de 0.8405 y un FP-rate de 0.1456. Esto demuestra el buen rendimiento de una red neuronal con pocas capas en este tipo de problemas. Se puede observar que, dado que el conjunto de datos está equilibrado (200 imágenes *No defectuosas* y 200 imágenes *Defectuosas*), las paridades entre los valores FN y FP (63 y 59 respectivamente) y entre los valores FN-rate y

FP-rate (0.1826 y 0.1456) demuestran que el modelo predice las dos clases de forma similar y razonablemente bien.

Tabla 5-2: Matriz de confusión acumulada de 10 repeticiones con el modelo básico sin DA.

		Predecido		
		Defectuoso	**No-defectuoso**	**Total**
Real	**Defectuoso**	332	63	395
	No-defectuoso	59	346	405

Resultados con DA - conjunto de datos aumentado Los resultados obtenidos con el modelo Básico y utilizando el proceso de DA, en el conjunto de datos original con 2000 imágenes, se muestran en la matriz de confusión acumulada en la Tabla **5-3**. A partir de ahí, se estima una exactitud global de 0.8510, encontrando una ligera mejora de 0.35 % con respecto a los resultados de exactitud del mismo modelo sin utilizar DA. Al igual que en los resultados del experimento anterior, cabe destacar la paridad entre la predicción para las dos clases, con valores similares para el FP-rate y el FN-rate de 0.1456 y 0.1594 respectivamente.

Tabla 5-3: Matriz de confusión acumulada de 10 repeticiones con el modelo básico y utilizando DA.

		Predecido		
		Defectuoso	**No-defectuoso**	**Total**
Real	**Defectuoso**	1642	367	2009
	No-defectuoso	229	1762	1991

Un análisis más profundo del mejor resultado obtenido hasta el momento, con el modelo básico y las imágenes aumentadas, puede verse en el reporte de clasificación respectivo que se muestra en la Tabla **5-4**. Obsérvese que los valores-F de 0.8464 y 0.8553 —para las clases *Defectuoso* y *No defectuoso*, respectivamente— indican la buena y similar capacidad del modelo para reconocer las dos clases.

Tabla 5-4: Reporte de clasificación para una media de 10 permutaciones con el modelo básico e imágenes aumentadas.

	Precisión	**Exhaustividad**	**valor-F**	**Soporte**
Defectuosos	0.8776	0.8173	0.8464	2009
No-defectuosos	0.8276	0.8849	0.8553	1991
Exactitud			0.8510	4000
Macro avg	0.8526	0.8511	0.8508	4000
Weighted avg	0.8527	0.8510	0.8508	4000

Los resultados globales en términos de precisión, TP-rate, FP-rate, FN-rate y TN-rate para el modelo básico en las dos versiones del conjunto de datos se pueden ver en la Tabla **5-5**. Esta tabla, a manera de resumen, nos permite confirmar que al utilizar DA se obtiene un mejor equilibrio entre los valores de reconocimiento de las dos clases.

5.5.2 Resultados con el modelo CNN basado en TL

El uso de TL permite potenciar el entrenamiento de los modelos para mejorar los resultados aprovechando que estos modelos se encuentran previamente entrenados con grandes conjuntos de datos. A continuación se

Tabla 5-5: Exactitud, TP-rate, FP-rate, FN-rate y TN-rate para el modelo básico.

Conjunto	Exactitud	TP-rate	FP-rate	FN-rate	TN-rate
Original	0.8475	0.8405	0.1456	0.1826	0.8849
Aumentado	0.8510	0.8173	0.1150	0.1594	0.8543

muestran y discuten los resultados de utilizar TL para el conjunto de datos original y para el conjunto de datos aumentado.

Resultados sin DA - conjunto de datos original Los resultados de la matriz de confusión acumulada correspondiente, para las 10 permutaciones del conjunto de datos se muestran en la Tabla **5-6**. Se puede observar que para el modelo basado en TL sin hacer uso de DA, la exactitud fue de 0.9212, lo que representa una mejora de 7.02 % sobre el mejor resultado con el modelo básico (0.8510). A partir de la matriz de confusión pueden obtenerse otras medidas de rendimiento, como una FP-rate de 0.0673 y una FN-rate de 0.0893, mostrando que el modelo funciona bien tanto en las categorías *Defectuoso* como *No defectuoso*.

Tabla 5-6: Matriz de confusión acumulada de 10 repeticiones con el modelo basado en TL sin DA.

		Predecido		
		Defectuoso	**No-defectuoso**	**Total**
Real	**Defectuoso**	377	37	414
	No-defectuoso	26	360	386

Resultados con DA - conjunto de datos aumentado La matriz de confusión acumulada obtenida de los experimentos para las 10 permutaciones, cuando se utiliza DA con el modelo basado en TL, se puede ver en la Tabla **5-7**. A partir de ahí, se puede determinar que la exactitud aumenta de 0.9212 cuando no se utiliza DA a 0.9522 cuando se aplica DA. Se trata del mejor resultado global obtenido hasta el momento, que representa una mejora de 3.1 %. Del mismo modo, de la matriz de confusión se derivan valores de 0.0249 para la FP-rate y de 0.0724 para la FN-rate, lo que revela cierto desequilibrio entre la predicción de las dos clases; de hecho, la predicción del modelo para las imágenes *No defectuosas* es mejor en un 4.75 % que para las imágenes *Defectuosas*.

Teniendo en cuenta que el modelo basado en TL y que utiliza DA proporciona el mejor resultado en términos de exactitud (0.9522), en la Tabla **5-8** se presenta el reporte de clasificación detallado del mismo. Puede observarse que dicho sistema detecta bastante bien la clase *Defectuoso* con una precisión de 0.9716 y una exhaustividad de 0.9275 y que, del mismo modo, el rendimiento para la clase *No defectuoso* también es competitivo: una precisión de 0.9359 y una exhaustividad de 0.9750.

Tabla 5-7: Matriz de confusión acumulada de 10 permutaciones para el modelo basado en TL con DA.

		Predecido		
		Defectuoso	**No-defectuoso**	**Total**
Real	**Defectuoso**	1779	139	1918
	No-defectuoso	52	2030	2082

Como resumen final, los resultados globales en términos de precisión, TP-rate, FP-rate, FN-rate y TN-rate resultantes de la utilización del modelo basado en TL se pueden ver en la Tabla **5-9**.

Tabla 5-8: Reporte de clasificación para el resultado promediado de 10 permutaciones con el modelo TL y utilizando DA.

	Precisión	**Exhaustividad**	**valor-F**	**Soporte**
Defectuoso	0.9716	0.9275	0.9490	1918
No-defectuoso	0.9359	0.9750	0.9550	2082
Exactitud			0.9522	4000
Prom. macro	0.9537	0.9512	0.9520	4000
Prom. ponderado	0.9530	0.9522	0.9521	4000

Tabla 5-9: Exactitud, TP-rate, FP-rate, FN-rate y TN-rate para el modelo con TL y utilizando DA.

Conjunto de datos	**Exactitud**	**TP-rate**	**FP-rate**	**FN-rate**	**TN-rate**
Original	0.9212	0.9106	0.0673	0.0893	0.9326
Aumentado	0.9522	0.9275	0.0249	0.0724	0.9750

5.5.3 Predicciones con los modelos

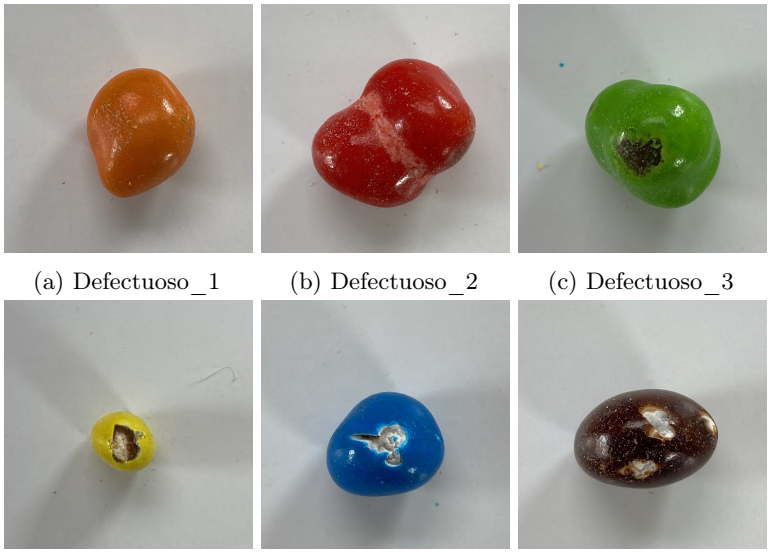
Una vez realizados los diferentes experimentos propuestos anteriormente y entrenados satisfactoriamente los modelos, se verificó la capacidad predictiva de los dos modelos en sus mejores versiones frente a 12 nuevas imágenes, que no pertenecen al conjunto de datos. En concreto, se consideraron 6 imágenes *No defectuosas* y 6 imágenes *Defectuosas*; ver Figs. **5-6** y **5-7** respectivamente.

(a) No_defectuoso_1 (b) No_defectuoso_2 (c) No_defectuoso_3

(d) No_defectuoso_4 (e) No_defectuoso_5 (f) No_defectuoso_6

Figura 5-6: Imágenes no defectuosas para la predicción mediante el modelo básico y el modelo TL con DA.

Las estimaciones resultantes de las probabilidades posteriores para estas 12 nuevas imágenes se presentan en la Tabla **5-10**, como duplas de la forma (posterior *No-defectuoso*, posterior *Defectuoso*). La segunda columna de la tabla corresponde a las predicciones con el modelo básico, mientras que la tercera columna corresponde a las predicciones con el modelo basado en TL. Obsérvese que el modelo básico clasifica correctamente 10 de las 12 imágenes, pero clasifica erróneamente las imágenes *No defectuoso_2* y *No defectuoso_3* como defectuosas con probabilidades posteriores de (0.0107, 0.9892) y ($3.770e{-}06$, 0.9999), respectivamente. Sin embargo, según nuestro etiquetado original, esas imágenes —véanse las Figs. **5-6**b y **5-6**c— corresponden a dulces sin defectos.



(a) Defectuoso_1 (b) Defectuoso_2 (c) Defectuoso_3

(d) Defectuoso_4 (e) Defectuoso_5 (f) Defectuoso_6

Figura 5-7: Imágenes defectuosas para la predicción mediante el modelo básico y el modelo TL con DA.

Tabla 5-10: Probabilidad posterior para 12 imágenes con el modelo básico y el modelo TL, donde el primer valor es la probabilidad para la clase *No_ defectuosa* y el segundo valor es la probabilidad para la clase *Defectuosa*.

Imagen	**Posterior Modelo Básico**	**Posterior Modelo TL**
No-defectuoso_1	(0.9498, 0.0501)	(0.9600, 0.0399)
No-defectuoso_2	**(0.0107, 0.9892)**	(0.7390, 0.2609)
No-defectuoso_3	**(3.770e-06, 0.9999)**	(0.9931, 0.0068)
No-defectuoso_4	(0.9987, 0.0012)	(0.5426, 0.4573)
No-defectuoso_5	(0.6406, 0.3593)	(0.9401, 0.0598)
No-defectuoso_6	(0.9937, 0.0062)	(0.9184, 0.0815)
Defectuoso_1	(0.1036, 0.8963)	(9.477e-08, 0.9999)
Defectuoso_2	(2.202e-05, 0.9999)	(1.916e-05, 0.9999)
Defectuoso_3	(0.0174, 0.9825)	(1.873e-07, 0.9999)
Defectuoso_4	(1.697e-06, 0.9999)	(1.136e-08, 1.0000)
Defectuoso_5	(0.3132, 0.6867)	(2.045e-09, 1.0000)
Defectuoso_6	(4.162-08, 1.0000)	(4.290e-06, 0.9999)

La predicción de clase y las probabilidades posteriores para las 12 imágenes nuevas, pero aplicando el modelo CNN basado en TL y utilizando DA, se muestran en la tercera columna de la Tabla **5-10**. Obsérvese que todas las imágenes se clasifican correctamente; no obstante, es importante destacar el caso de la imagen *No defectuosa_4* (véase la Figura **5-6**d), para la que este modelo arrojó valores posteriores de (0.5426, 0.4573). Aunque la asignación de clase en ese caso es correcta, la decisión no es segura debido a la pequeña diferencia entre los dos valores de las probabilidades posteriores. Para las imágenes consideradas como *Defectuosas* puede verse que, en todas las probabilidades posteriores, la segunda clase es ampliamente dominante, lo que significa que todas las imágenes pertenecen con alta confianza a la segunda categoría correspondiente a imágenes *Defectuosas*.

5.6 Conclusiones

En este capítulo se ha presentado una aplicación de técnicas de aprendizaje profundo para clasificar dulces Defectuosos y No defectuosos, junto con un nuevo conjunto de datos compuesto por 400 imágenes tomadas a 200 dulces: 200 imágenes pertenecen a dulces *No defectuosos* y los otros 200 a dulces considerados *Defectuosos*. El conjunto de datos se ha puesto a disposición del público en un repositorio GitHub asociado [1]. Los resultados de la serie de experimentos que se llevaron a cabo —utilizando diferentes modelos de redes neuronales y técnicas como DA y TL— permitieron extraer las siguientes conclusiones:

Dada la naturaleza y forma elipsoidal de los dulces, resulta difícil abarcar todas las caras del dulce en una sola imagen. Este problema se resolvió parcialmente tomando dos imágenes por dulce; sin embargo, pueden persistir oclusiones, lo que podría dar lugar a resultados de clasificación incorrectos. Además, dado que los dulces son productos alimenticios, los defectos visuales no son necesariamente determinantes a la hora de realizar un control de calidad. No obstante, es muy importante encontrar los defectos que más influyen en el aspecto aceptable del producto, como grietas, deformaciones y tamaños no estándar.

Después de realizar las pruebas preliminares, se comprobó que no hay diferencias significativas entre utilizar imágenes en escala de grises o en color, aunque los modelos entrenados con imágenes en color se comportaron ligeramente mejor. Además, en el modelo TL se probaron varias arquitecturas de red populares. Se observó que ResNet50V2, junto con imágenes en color, proporciona las mejores predicciones para resolver este problema. Además, en el modelo TL, se probaron varias arquitecturas de red populares, a saber: inceptionResNetV2, VGG16, ResNet50V2 y ResNet152V. A partir de esos resultados, se observó que ResNet50V2, junto con las imágenes en color, proporciona las mejores predicciones para resolver este problema.

Los resultados experimentales realizados con dos modelos basados en CNN y dos variantes del conjunto de datos (con y sin DA) sugieren que tanto TL como DA son técnicas clave para mejorar el rendimiento de la clasificación. De hecho, los mejores resultados globales en todas las métricas se obtuvieron para el modelo TL con DA, a saber, una exactitud de 0.9522, una precisión de 0.9716, un FN-rate de 0.0724 y un FP-rate de 0.0249. Además, dado que no se observaron grandes diferencias para las métricas y las probabilidades posteriores por clase, puede concluirse que el sistema seleccionado es una buena solución para el problema de clasificación de dulces.

[1] https://github.com/ejvillegasj/candies

Parte III

Inspección visual con técnicas de aprendizaje inexactamente supervisado

6 Paralelización del cálculo de las disimilitudes en conjuntos de puntos para bolsas con muchas instancias

Este capítulo se basa en los siguiente trabajos:

- Eduardo Villegas-Jaramillo, y Mauricio Orozco-Alzate *''Computational Analysis of Multiple Instance Learning-based Systems for Automatic Visual Inspection: A Doctoral Research Proposal''* [Villegas-Jaramillo and Orozco-Alzate. 2019].
  Presentada en la ''15th International Conference on Distributed Computing and Artificial Intelligence (DCAI 2018)'', llevada a cabo entre el 20 y el 22 de junio de 2018 en la ciudad de Toledo - España.
- Eduardo Villegas-Jaramillo, Ana Lorena Uribe-Hurtado, y Mauricio Orozco-Alzate. *''Multi-core parallelization of point set dissimilarities for accelerating the comparison of bags with many instances''*. In Sigeru Omatu, Rashid Mehmood, Pawel Sitek, Serafino Cicerone, and Sara Rodríguez, editors, Distributed Computing and Artificial Intelligence, 19th International Conference, pages 208–218, Cham, 2023. Springer International Publishing. ISBN 978-3-031-20859-1. [Villegas-Jaramillo et al.. 2023]

6.1 Introducción

La propuesta original de este trabajo de tesis, que se desarrolló para ser presentada en el respectivo examen de calificación, tenía como principal objetivo explorar alternativas para mejorar la complejidad de los algoritmos de clasificación. Una vez realizado dicho examen, atendiendo las observaciones de los jurados, se realizaron diferentes ajustes a la propuesta doctoral, involucrando el uso de redes neuronales y no centrándose en el costo computacional como principal problema a resolver.

La descomposición en bloques es una alternativa para el modelado de instancias que, de forma natural, genera muchas instancias (tantas instancias como bloques en la cuadrícula) y, por tanto, produce bolsas con una gran cardinalidad. Con el fin de atender el problema del costo computacional que se presenta al calcular las disimilitudes entre bolsas con muchas instancias se presentan a continuación los resultados de una propuesta de solución que involucra el uso de estrategias de paralelización utilizando múltiples núcleos de procesamiento.

DL ha surgido como una solución general y potente para los problemas de ML. De hecho, DL se ha convertido en la estrategia dominante tanto en estudios de investigación como en el desarrollo de aplicaciones, incluyendo casos de supervisión débil debido principalmente a la capacidad de las CNN para abordar el reconocimiento de

escenas visuales complejas e integrar —en sí mismas— los pasos de segmentación/regionalización, extracción de características y clasificación como lo presentado en [Smith et al.. 2021]. Sin embargo, los autores de este trabajo, que a su vez hacen referencia a [Bier. 2019], también destacan que *''algunas tareas no son adecuadas para DL''*, en particular porque DL requiere grandes cantidades de datos etiquetados que estén disponibles para entrenar las redes con éxito; no obstante, la escasez de datos etiquetados a menudo se supera utilizando DL, y hasta cierto punto, mediante el uso de estrategias como el TL y el DA. Cuando las regiones de interés en las imágenes son demasiado pequeñas o complejas, una estrategia común es proporcionar máscaras o anotaciones de la imagen para entrenar una red complementaria que realice la segmentación, véanse por ejemplo las dos redes utilizadas en [Božič et al.. 2021]. Sin embargo, dichas anotaciones no siempre están disponibles debido al coste que supone —para el experto humano— proporcionar una delimitación detallada de las regiones de interés para una cantidad suficiente de ejemplos de entrenamiento.

Por lo tanto, a pesar de la popularidad actual y la superioridad general de los métodos basados en DL, todavía hay gran interés en versiones avanzadas de MIL, como por ejemplo —por mencionar algunas iniciativas— MIL con bolsa de disimilitudes [Cheplygina and Tax. 2015], MIL incremental para aprender conceptos de objetivos no estacionarios y recurrentes [Mera et al.. 2019] y un modelo probabilístico discriminativo para etiquetas faltantes en MIL [Nguyen and Raich. 2022]. El primero —MIL con bolsa de disimilitudes— es particularmente atractivo para el desarrollo de este trabajo porque brinda a MIL de los beneficios del llamado enfoque de representación de disimilitudes [Pekalska and Duin. 2005]. A pesar de que MIL utiliza menos parámetros que DL, no está exento de cuestiones críticas como la selección adecuada de las posibles regiones de interés para ser modeladas como instancias. Según [Mera. 2017, Cap. 5], esa tarea puede llevarse a cabo aplicando cualquiera de las siguientes estrategias: i) descomponiendo las imágenes mediante una rejilla de bloques, donde cada bloque se considera una instancia, como se ejemplifica en la Figura **6-2**; ii) confiando en un etiquetado débil, lo que significa que se asume que algunas anotaciones son proporcionadas por los expertos y iii) aplicando un método de segmentación fiable para detectar regiones en la imagen que corresponderán a las instancias. La primera estrategia es sencilla, ya que no requiere ningún conocimiento previo ni la aplicación de algoritmos elaborados y sofisticados. Sin embargo, la descomposición en bloques se ha descartado a menudo como alternativa de modelado de instancias porque genera de forma natural muchas instancias y, por tanto, produce bolsas con una gran cardinalidad. Además, según [Cheplygina and Tax. 2015, p. 274] algunas disimilitudes de bolsas —particularmente las que consideran las bolsas como conjuntos de puntos— *''pueden llegar a ser demasiado costosas de calcular''* para bolsas grandes.

En lugar de preferir las otras estrategias de modelado de bolsas al adoptar MIL con disimilitudes de bolsas, se seleccionó la opción de modelado más simple (es decir, una descomposición en bloques que produce muchas instancias) y explorar las posibilidades de acelerar el cálculo de las disimilitudes de conjuntos de puntos mediante estrategias de paralelización. Como problema ilustrativo para resolver utilizando MIL con disimilitudes entre conjuntos de puntos, se realizaron experimentos para simular el cálculo de las disimilitudes entre pares de bolsas, teniendo éstas tamaños realistas para el modelado por descomposición en bloques de imágenes basadas en un conjunto de datos público de defectos superficiales en tejidos con texturas.

6.2 Métodos

MIL se utiliza normalmente en los casos en que las etiquetas de clase se asignan a toda la bolsa, pero no a sus instancias individuales; sin embargo, no hay ninguna restricción formal para construir incluso bolsas sin etiquetas de clase; por ejemplo, para realizar una agrupación. Dado que el cálculo de las disimilitudes entre conjuntos de puntos —ya sea para la clasificación supervisada o no supervisada— no hace uso de las etiquetas de clase, aquí se introduce la notación MIL necesaria pero, en aras de la claridad, se ignoran intencionadamente las etiquetas. Los detalles más relevantes de MIL fueron previamente analizados en la Sección 3.4.

En MIL, la i-ésima imagen del conjunto de datos se representa como su correspondiente bolsa $\mathcal{X}_{(i)} =$

$\left\{\vec{X}_{(i,1)}, \ldots, \vec{X}_{(i,k)}, \ldots \vec{X}_{(i,N_i)}\right\}$, donde las N_i instancias (también conocidas como vectores de características) son puntos en un espacio vectorial M-dimensional; es decir, se supone que se utiliza el mismo número (M) de características para representar las instancias; Además, las instancias para otra bolsa $\mathcal{X}_{(j)}$ —en el mismo problema de clasificación— también se encuentran en $\mathbb{R}^M$ pero pueden diferir en número [1], es decir, en general $N_i \neq N_j$. Nótese que una bolsa puede entenderse simplemente como un conjunto de puntos en $\mathbb{R}^M$ y, por tanto, las disimilitudes entre conjuntos de puntos son opciones naturales para compararlas cuantitativamente.

6.3 Disimilitudes entre conjuntos de puntos

Las nociones y conceptos necesarios para entender las disimilitudes entre conjuntos de puntos, fueron definidos previamente en la subsección 3.6.3. En esta sección y con el fin de facilitar la lectura se repite la definición de las cuatro opciones para calcular las disimilitudes entre conjuntos de puntos:

$$d_{maxmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \max_k \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{6-1}$$

$$d_{minmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \min_k \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{6-2}$$

$$d_{meanmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \frac{1}{N_i}\sum_{k=1}^{N_i} \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{6-3}$$

$$d_{meanmean}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \frac{1}{N_i N_j}\sum_{k=1}^{N_i}\sum_{l=1}^{N_j} d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{6-4}$$

6.4 Paralelización

Obsérvese que las operaciones iterativas externas en el cálculo de las disimilitudes entre conjuntos de puntos —a saber $\max_k$ en la Ec. (6-1), $\min_k$ en la Ec. (6-2) y $\sum_k$ en ambas Ecs. (6-3) y (6-4)— son operaciones asociativas. Recuerde que una operación $\star$ es asociativa si $\forall a, b, c,\ a\star(b\star c) = (a\star b)\star c$. Por lo tanto, el cálculo de las disimilitudes entre el conjunto de puntos es vergonzosamente paralelizable sobre esas operaciones simplemente considerando tantos subproblemas como el número de unidades de procesamiento disponibles, donde las unidades de procesamiento son los hilos de los núcleos físicos del procesador. Sea P el número de unidades de procesamiento, ya que la cardinalidad de la i-ésima bolsa no es necesariamente múltiplo de P, la operación paralelizada debe dividirse en $P-1$ subproblemas del mismo tamaño y el último se ejecuta sobre las instancias restantes de $\mathcal{X}_{(i)}$.

En las Ecs. (6-5) a (6-9), se presenta una formulación para el cálculo paralelo de cualquiera de las cuatro disimilitudes entre conjuntos de puntos, sobre P unidades de procesamiento. La forma en que la carga de trabajo se distribuye a través de las unidades de procesamiento se muestra en la Ec. 6-9), donde la bolsa $\mathcal{X}_{(i)}$ se divide en P sub bolsas, denotadas $\mathcal{X}_{(i)}^{(p)}$, para posteriormente calcular la distancia con respecto a todas las instancias de la bolsa $\mathcal{X}_{(j)}$, ver Figura **6-1**.

[1] En la práctica, esto significa que MIL es capaz de tratar con imágenes de diferentes tamaños sin necesidad de aplicar ninguna operación previa de redimensionamiento. Por el contrario, las redes neuronales convolucionales sí requieren imágenes del mismo tamaño.

$$d_{nombre} = a \cdot f\left(f\left(g\left(\mathcal{X}_{(i)}^{(0)}, \mathcal{X}_{(j)}\right)\right), ..., f\left(g\left(\mathcal{X}_{(i)}^{(p)}, \mathcal{X}_{(j)}\right)\right), ..., f\left(g\left(\mathcal{X}_{(i)}^{(P-1)}, \mathcal{X}_{(j)}\right)\right)\right) \tag{6-5}$$

donde,

$$a = \begin{cases} 1 & \text{if } nombre = maxmin \text{ or } nombre = minmin \\ \frac{1}{N_i} & \text{if } nombre = meanmin \\ \frac{1}{N_i N_j} & \text{if } nombre = meanmean \end{cases} \tag{6-6}$$

$$f = \begin{cases} max & \text{if } nombre = maxmin \\ min & \text{if } nombre = minmin \\ \sum & \text{if } nombre = meanmin \text{ or } nombre = meanmean \end{cases} \tag{6-7}$$

$$g = \begin{cases} min & \text{if } nombre = maxmin \text{ or } nombre = minmin \text{ or } nombre = meanmin \\ \sum & \text{if } nombre = meanmean \end{cases} \tag{6-8}$$

$$\mathcal{X}_{(i)}^{(p)} = \begin{cases} \mathcal{X}_{(i)}\left[p * m : (p+1) * m + 1\right] & \text{if } 0 \leq p < P - 1 \\ \mathcal{X}_{(i)}\left[(P-1) * m : len(\mathcal{X}_{(i)})\right] & \text{if } p = P - 1 \end{cases} \tag{6-9}$$

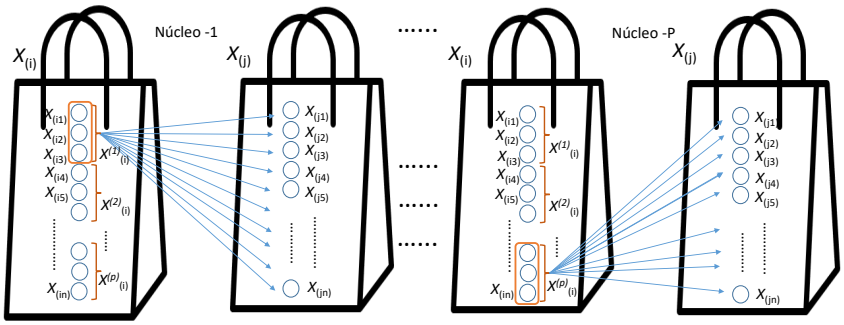


Figura 6-1: Distribución de la carga de trabajo para el cálculo paralelo de la disimilitud de los conjuntos de puntos entre instancias de dos bolsas.

6.5 Configuración experimental

Los resultados experimentales de la paralelización se ilustraron con respecto a un conjunto de imágenes proveniente de la Asociación de Investigación de la Industria Textil (AITEX) [Silvestre-Blanes et al.. 2019], que está conformado por 245 imágenes de 7 tejidos diferentes. Hay 140 imágenes libres de defectos (20 para cada tipo de tejido) y 105 imágenes con diferentes tipos de defectos. Todas las imágenes tienen un tamaño de 4096×256 píxeles. Los experimentos se realizaron utilizando las cuatro medidas de disimilitud entre bolsas en un servidor DELL con 24 núcleos (Intel Xeon$^{(R)}$

CPU E7-4860 v2 @ 2.60GHz) y 48 hilos, bajo Scientific Linux release 7.4 (Nitrogen), con lenguaje de desarrollo Python 3.7.2. Se realizaron cálculos para 10 repeticiones tanto con el algoritmo secuencial como con el algoritmo paralelo para 2, 4, 6, 8, 12, 24 y 48 unidades de procesamiento. Las instancias de las bolsas se generaron dividiendo las imágenes en bloques de 8×8, 16×16, 32×32 y 64×64 (véase la Figura **6-2**). Esos tamaños de cuadrícula para la descomposición en bloques producen bolsas con 16384, 4096, 1024 y 256 instancias respectivamente, que se simularon como vectores aleatorios de dimensión 50 que se asemejan a las 50 características propuestas en [Mera et al.. 2019] para la representación de instancias. Las características propuestas en ese trabajo abarcan descriptores de color, textura y forma.

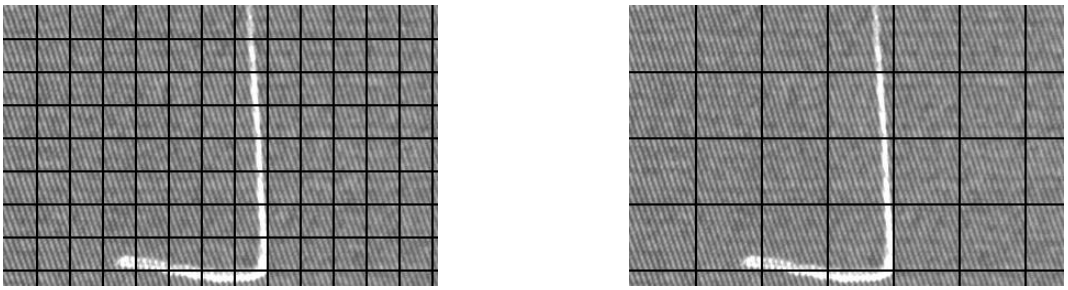


Figura 6-2: Ejemplos de imágenes recortadas del conjunto de datos AITEX [Silvestre-Blanes et al.. 2019], que ilustran la generación de bloques en dos tamaños diferentes de cuadrícula que, a su vez, corresponderán con las instancias.

La función *time()* disponible en la librería *time* de Python, se utilizó para medir los tiempos de procesamiento requeridos para cada una de las funciones de disimilitud definidas anteriormente, de tal forma que el tiempo se mide desde la creación de los procesos, pasando por la generación de índices para la partición de la bolsa $\mathcal{X}_{(i)}$, así como incluyendo el respectivo cómputo de los resultados realizados por cada unidad de proceso y, finalmente, el cómputo del resultado global aplicando la misma operación a los resultados parciales obtenidos por cada unidad de proceso (ver Código 6.2).

En los Códigos 6.1 y 6.2 se presentan los códigos en lenguaje Python para el cálculo secuencial y multinúcleo de la disimilitud d_{maxmin} entre dos bolsas. No se presentan las implementaciones paralelas para las otras tres disimilitudes de conjuntos de puntos debido a que son muy similares y fácilmente reemplazables por el lector.

Código 6.1: Código Python para el cálculo secuencial de la disimilitud d_{maxmin} entre dos bolsas.

```
import random , time, sys
import numpy as np
np.random.seed(0)

def dissMaxMin(Xi, Xj):
    distancesExternal = []
    for k in range(len(Xi)):
        distancesInternal = []
        for l in range(len(Xj)):
            distancesInternal.append(np.linalg.norm(Xi[k]-Xj[l]))
        distancesExternal.append(np.min(distancesInternal))
    return np.max(distancesExternal)

if __name__ == "__main__":
    Ni = 2000; Nj = 1000 # Simulate two bags with 2000 and 1000 instances,
    # each instance having a dimension 50
    Xi = np.random.rand(Ni, 50) # Sample random data
    Xj = np.random.rand(Nj, 50) # Sample random data
    start = time.time() # Start the chronometer
    result = dissMaxMin(Xi, Xj) # Call the function
    elapsedTime = time.time()- start # Take the elapsed time
    print("The computation took " + str(elapsedTime) + " seconds")
    print("The value of the MaxMin dissimilarity is: " + str(result))
```

Código 6.2: Código Python para el cálculo multinúcleo de la disimilitud d_{maxmin} entre dos bolsas.

```
import random , time, sys
from multiprocessing import Pool, cpu_count
import numpy as np; np.random.seed(0)

def dissMaxMin(Xi, Xj):
    distancesExternal = []
    for k in range(len(Xi)):
        distancesInternal = []
        for l in range(len(Xj)):
            distancesInternal.append(np.linalg.norm(Xi[k]-Xj[l]))
        distancesExternal.append(np.min(distancesInternal))
    return np.max(distancesExternal)

def blockTasks(subvectors):
    return dissMaxMin(subvectors[0], subvectors[1])

if __name__ == "__main__":
    Ni = 2000; Nj = 1000 # Simulate two bags with 2000 and 1000 instances,
    # each instance having a dimension of 50
    Xi = np.random.rand(Ni, 50) # Sample random data
    Xj = np.random.rand(Nj, 50) # Sample random data
    P = cpu_count() # Count the number of available processing cores
    start = time.time() # Start the chronometer
    pool = Pool(processes=P) # Create a process Pool with P processes

    # Generation of indexes to split feature vectors into blocks:
    m = len(Xi)//P
    arg = [(Xi[p*m:(p+1)*m+1,:], Xj) for p in range(P-1)]
    arg.append((Xi[(P-1)*m:,:], Xj))
    partialResults = pool.map(blockTasks , arg) # Map blockTasks to the Pool

    finalResult = max(partialResults) # Perform the external max function
    elapsedTime = time.time() - start # Take the elapsed time
    print("The computation took " + str(elapsedTime) + " seconds")
    print("The value of the MaxMin dissimilarity is: " + str(finalResult))
```

6.5.1 Tiempos de ejecución, aceleraciones y complejidad algorítmica

Los incrementos de velocidad mostrados en las Figs. **6-4** y **6-5** se calcularon como $SUP = ET_s/ET_p$, donde ET_s es el tiempo de ejecución del algoritmo secuencial y ET_p es el tiempo de ejecución del algoritmo paralelo. En todos los casos, se consiguen altos incrementos de velocidad para bolsas con pocas instancias cuando se especifica un bloque de 32×32 píxeles. Para ese tamaño de bloque, el incremento de velocidad máximo se consigue con 24 núcleos físicos, mientras que para el tamaño de bloque más grande de 64×64 píxeles, que produce el menor número de instancias, el incremento de aceleración máximo se consigue utilizando 12 núcleos; véase la Tabla **6-1**. Esto se explica por la baja carga de las unidades de procesamiento.

Tabla 6-1: Aceleraciones máximas alcanzadas para cada tamaño de bloque según el número de instancias procesadas. Ins-UP se refiere a la cantidad de instancias para cada unidad de procesamiento y UP a la cantidad de unidades de procesamiento requeridas para alcanzar la aceleración máxima.

Tam. bloque	#Ins.	Ins-UP	MaxMin	MinMin	MeanMin	MeanMean	UP
8×8	16384	341	22.69	22.54	22.89	23.22	48
16×16	4096	85	20.57	21.15	20.90	20.51	48
32×32	1024	43	14.52	14.05	14.48	14.02	24
64×64	256	21	6.20	6.26	5.31	5.83	12

Nótese que, cuando se requiere realizar cálculos con hasta un máximo de 24 unidades de proceso, se utilizan los 24 núcleos o procesadores físicos del computador, cada uno de los cuales emplea un único hilo; sin embargo, cuando

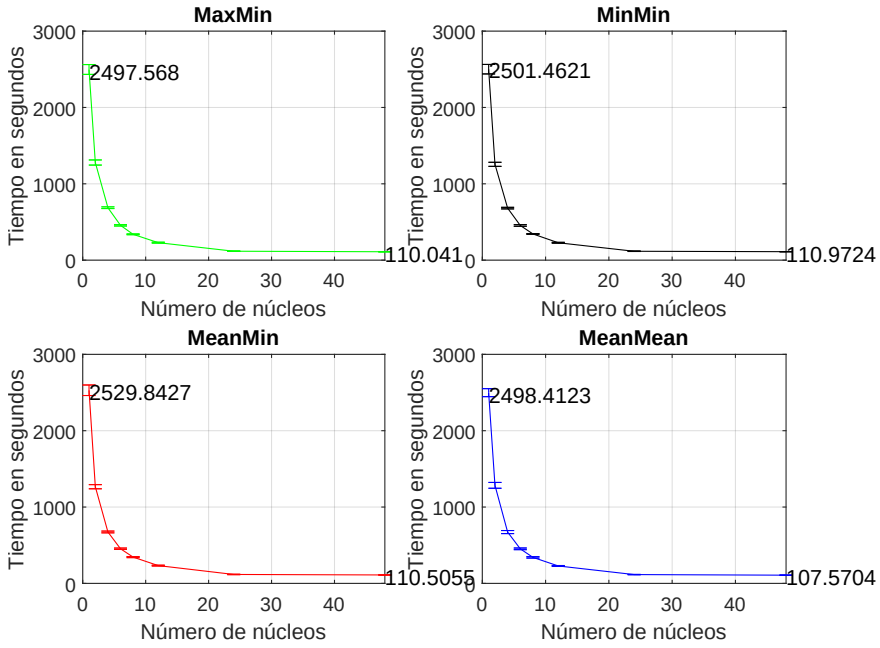


Figura 6-3: Tiempos transcurridos para el cálculo de cada medida de disimilitud del conjunto de puntos en función del número de núcleos (tamaño de bloque 8×8).

las necesidades superan el número de núcleos físicos, el planificador comienza a utilizar dos hilos por núcleo, hasta alcanzar el máximo de 48 hilos totales; no obstante, se observa que la tasa de aceleración, en ese caso, crece lentamente como se ve en la Figura **6-4**. Esta es la razón por la que las tasas de aceleración máximas se alcanzan hasta un máximo de 24 unidades de procesamiento, y a partir de ahí es posible crecer a una tasa muy baja (véase la segunda fila de gráficos de la Figura **6-4**).

La figura **6-3**, muestra que los tiempos de ejecución de las cuatro funciones de disimilitud calculadas para un tamaño de bloque 8×8 tienen un comportamiento estable con resultados similares, indicando que las operaciones *sum*, *max* y *min* tienen casi el mismo coste computacional, lo que se refleja en la aceleración del cálculo de la distancia para cada tamaño de bloque (ver Figura **6-5**).

En cuanto al cálculo de la complejidad algorítmica, la función MaxMin en su versión secuencial está compuesta por dos ciclos anidados que permiten calcular una determinada medida de disimilitud para cada una de las instancias de la primera bolsa respecto a todas las instancias en la segunda bolsa, encontrando el máximo de los mínimos resultantes. La complejidad dependerá de dos elementos, en primer lugar, el número de instancias de las bolsas que en nuestro caso siempre son de la misma dimensión, y el segundo, de la función de distancia utilizada. Si asumimos que la cantidad de instancias por bolsa es n y, como las funciones de distancia utilizadas pertenecen a las bibliotecas de Python cuya complejidad se asume como $\mathcal{O}(1)$, la complejidad resultante de la función secuencial será: $\mathcal{O}(n^2)$.

Para la función MaxMin en su versión paralela utilizando un computador con P procesadores, se mantienen los dos ciclos anidados, pero se reparten las instancias de la primera bolsa sobre los P procesadores, para así compararlas respecto a todas las instancias de la segunda bolsa, lo que se traduce en una complejidad de $\mathcal{O}(n/P * n)$.

6.6 Conclusiones

En este trabajo se propuso una implementación paralela de cuatro medidas de distancia entre conjuntos de puntos que se utilizaron para calcular las disimilitudes entre pares de bolsas de grandes dimensiones que representan imágenes. Las bolsas grandes surgen de forma natural cuando se descomponen imágenes de alta resolución con cuadrículas de

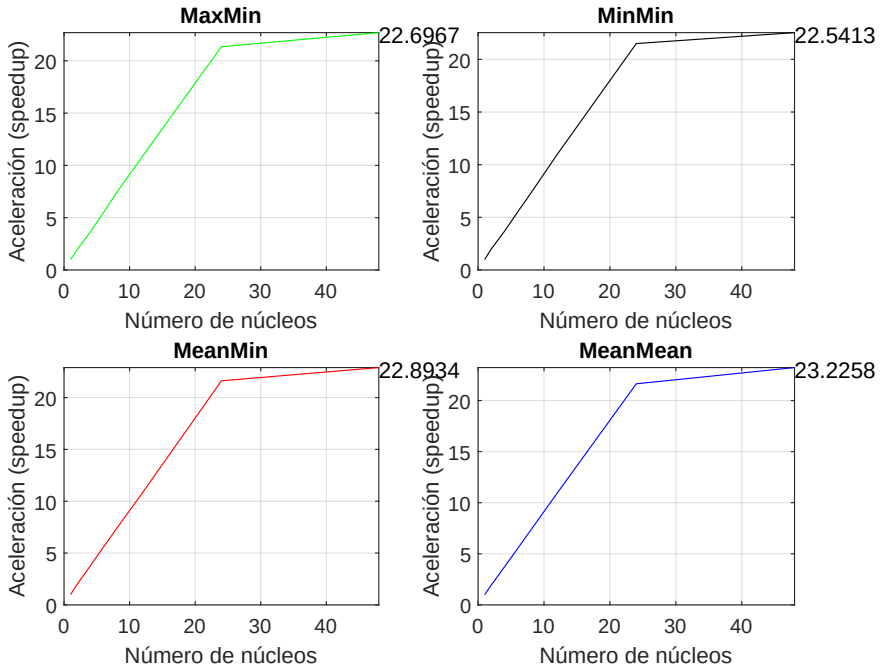


Figura 6-4: Aceleración de las cuatro medidas de disimilitud (tamaño de bloque de 8×8)).

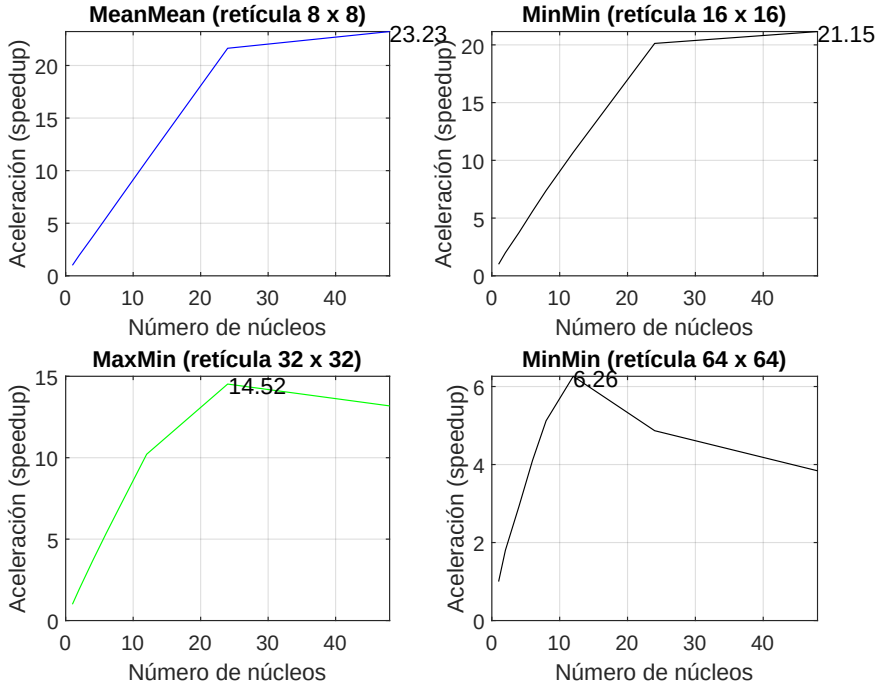


Figura 6-5: Aceleraciones (speedups) máximas alcanzadas en función del número de núcleos y para cuatro tamaños de bloque diferentes.

bloques pequeños para ser consideradas como instancias, así también cuando la descomposición en bloques se hace con traslape entre ellos, por lo que resulta muy costoso calcular la disimilitud entre las bolsas debido a la cantidad de cálculos necesarios y a los elevados tiempos de procesamiento transcurridos. Se realizaron varias pruebas estimando los ‘‘speedups’’ o aceleraciones resultantes con cuatro tamaños de bloques para un problema de inspección visual automática, de forma que se ilustrase el comportamiento de la estrategia paralela propuesta bajo diferentes escenarios con respecto al número de instancias. Las cuatro variantes de tamaños de bloque cubren un rango razonable de tamaños de problema que se pueden encontrar al trabajar con imágenes.

Independientemente de la medida de disimilitud seleccionada entre las cuatro medidas consideradas, tanto el comportamiento como los ‘‘speedups’’ fueron similares, a saber: se observaron aceleraciones de hasta 23 veces al utilizar 48 unidades de procesamiento en una máquina con 24 núcleos físicos, siempre que el número de instancias asignadas a cada unidad de procesamiento sea suficientemente grande; las tasas máximas de aceleración se alcanzan cuando se utilizan todos los núcleos físicos del computador; a partir de ahí, la pendiente de aceleración sigue creciendo pero a un ritmo mucho menor, en cumplimiento de la ley de Amdahl.

En el capítulo siguiente (Cap. 7) se mostrará el uso de DL como extractor de características bajo un enfoque MIL y como trabajo futuro, se deja propuesta la aplicación.

Parte IV

Inspección visual heterogénea combinando aprendizaje profundo y aprendizaje inexactamente supervisado

7 Metodología inexactamente supervisada basada en aprendizaje de múltiples instancias, redes neuronales y disimilitudes entre bolsas

Este capítulo se basa en el siguiente artículo:

- Eduardo Villegas-Jaramillo y Mauricio Orozco-Alzate. *''An Inexactly Supervised Methodology Based on Multiple Instance Learning, Convolutional Neural Networks and Dissimilarities for Interpretable Defect Detection and Localization on Textured Surfaces''*, publicado en IEEE access, vol. 11, pp. 138229-138246, 2023. [Villegas-Jaramillo and Orozco-Alzate. 2023b].

Se presenta el desarrollo de una metodología que mediante la combinación de diferentes tipos de técnicas permita realizar la clasificación, localización e interpretación de defectos en imágenes de superficies texturizadas, la cual parte de una descomposición de las imágenes en bloques sobre diferentes tamaños de cuadrículas, utilizando bolsas como esquema de representación mediante el enfoque MIL, cuyas instancias corresponden a vectores de características generados ya sea mediante redes neuronales con el apoyo de TL, así como técnicas de extracción de características de bajo nivel basadas en colores, texturas, formas y contornos. Posteriormente, se aplica el algoritmo k-NN por medio del cálculo de las disimilitudes entre bolsas, y posteriormente se utiliza dicha información para realizar las tareas de clasificación, localización e interpretación de defectos.

7.1 Introducción

Las técnicas de ML se han aplicado en diferentes contextos industriales, incluyendo la clasificación de objetos y la localización de defectos para el control de calidad. Muchos de los objetos industriales a inspeccionar son superficies como tejidos, láminas de acero, láminas de vidrio, entre otros; su naturaleza plana, por tanto, los hace especialmente adecuados para la clasificación y localización por sistemas de inspección visual [Datta and Davim. 2022].

Este problema se ha resuelto desde varios enfoques, incluida la detección de objetos totalmente supervisados (FSOD) y la detección de objetos supervisados débilmente (WSOD). Con respecto al primer enfoque, que utiliza anotaciones a nivel de instancia para entrenar el sistema, es importante destacar el conjunto de técnicas asociadas al DL como una buena alternativa cuando se aplica en sus diferentes variantes como las CNN, redes profundas y redes basadas en RL [Zhou et al.. 2019], que se pueden complementar con técnicas como el DA [Shorten and Khoshgoftaar. 2019] y el TL [Hafemann et al.. 2015]. Con el método DL se han obtenido buenos resultados sobre todo en la clasificación de objetos y, en menor medida, en la localización de defectos. El problema anterior se ha resuelto recientemente utilizando dos tipos de detectores de objetos de última generación, incluidos los detectores de dos etapas, tales como: R-CNN [Girshick et al.. 2014], Fast R-CNN [Girshick. 2015], Faster R-CNN [Ren et al.. 2017] y Mask R-CNN [He et al.. 2017].

Por otro lado, entre los detectores de una etapa, You Only Look Once (YOLO) [Redmon et al.. 2016] y Single Shot Detector (SSD) [Liu et al.. 2016] son los más populares. Algunas de estas variantes de detectores de objetos requieren inicialmente construir un conjunto de regiones candidatas ya sea mediante búsquedas exhaustivas [Li and Zhao. 2019], creando mapas de características [Marin et al.. 2021] o generando regiones con una CNN adicional [Vemula and Frye. 2020] para cuyo entrenamiento se requiere contar con imágenes ejemplares con las correspondientes etiquetas de clases de defectos, así como con máscaras o anotaciones de sus ubicaciones.

Otro enfoque que se ha destacado más recientemente es WSOD, como una estrategia para la detección de objetos en la que el modelo se entrena utilizando etiquetas incompletas o inexactas [Zhang et al.. 2022]. A diferencia del enfoque FSOD, en el que el modelo se entrena utilizando datos etiquetados con precisión, WSOD requiere solo anotaciones parciales o etiquetas a nivel de imagen, que no indican la ubicación precisa del objeto en ella [Shao et al.. 2022]; esta condición con las etiquetas se conoce como supervisión inexacta en el aprendizaje débilmente supervisado [Zhou et al.. 2018]. MIL, como un caso particular de WSOD [Shao et al.. 2022], asume que la etiqueta de clase de los ejemplos de entrenamiento se asigna a una representación de la imagen completa del objeto (llamada *bolsa de instancias*) pero no va acompañada de información sobre la localización del defecto; por lo tanto, para manejar esa información inexacta, la imagen se descompone en regiones de interés utilizando un algoritmo de propuestas de regiones o una estrategia de segmentación. Posteriormente, cada región resultante se representa mediante un vector de características (llamado *instancia*), y el conjunto de ellas se agrupa en la bolsa etiquetada [Mera et al.. 2016].

Una vez que se resuelve el problema de localización, es importante proporcionar una interpretación de por qué el sistema asignó una etiqueta de clase a una imagen específica; sin embargo, en muchos escenarios, es difícil determinar cómo llegó el sistema a asignar la etiqueta debido a la naturaleza de caja negra de las CNN; a veces ni siquiera es posible determinar la calidad de la solución. Sin embargo, existen iniciativas recientes que buscan explicar analíticamente cómo estas redes alcanzan una respuesta específica [Saadallah et al.. 2022] o mostrar empíricamente los llamados mapas de activación de clase (Class Activation Maps CAM) como indicadores de las regiones discriminatorias en la imagen [Zhou et al.. 2016].

Dada la frecuente ausencia de máscaras o anotaciones y los problemas asociados a la generación de regiones candidatas para guiar este procedimiento, así como la dificultad de interpretar los resultados de clasificación arrojados por las CNNs y otras técnicas, es deseable desarrollar un sistema automático de inspección visual que, preservando el poder de representación de las redes neuronales, no requiera de una costosa etapa de propuesta de regiones y utilice una regla de decisión que sea fácil de interpretar. Por lo tanto, este trabajo propone una metodología débilmente supervisada que no requiere las ubicaciones de los defectos para el entrenamiento, la cual puede ser utilizada para la clasificación de objetos, localización e interpretación de defectos en diferentes tipos de superficies. La metodología propuesta parte de una descomposición de la imagen en bloques utilizando diferentes tamaños de cuadrícula. Luego, las imágenes se representan como bolsas utilizando el enfoque MIL, cuyas instancias corresponden a vectores de características generados por redes neuronales con el apoyo de TL. Como método de extracción de características de referencia y aplicando técnicas de procesamiento de imágenes, se han utilizado características de bajo nivel basadas en colores, texturas, formas y contornos como en [Mera et al.. 2016]. Posteriormente, se aplica el algoritmo k-NN calculando las diferencias entre las bolsas [Cheplygina et al.. 2015] y —después— esa información se utiliza para realizar la clasificación, localización e interpretación, obteniendo buenos resultados que se ilustran con diferentes métricas, gráficos, conteos, ejemplos y figuras correspondientes a experimentos con tres conjuntos de datos públicos: dos de ellos son conjuntos de datos sintéticos provenientes de la Sociedad Alemana para el Reconocimiento de Patrones (DAGM, por sus siglas en alemán) y el tercero es un conjunto con imágenes reales de telas desarrollado por AITEX.

En resumen, las principales contribuciones de este capítulo son las siguientes: i) Se desarrolla una metodología que permite entrenar un sistema de clasificación de objetos y localización de defectos utilizando redes neuronales sin utilizar máscaras o anotaciones que delimiten la ubicación de los defectos. ii) Se utiliza el enfoque MIL con disimilitudes entre bolsas para la representación y una implementación de la distancia de Hausdorff aplicada a las bolsas usando k-NN como clasificador. iii) Se desarrolla una estrategia gráfica para interpretar los resultados de la localización de defectos, ayudando así a validar y comprender los resultados obtenidos.

7.2 Metodología propuesta y sus métodos

La metodología propuesta de supervisión inexacta y la línea de base utilizada para la comparación se presentan en la Figura **7-1**. La parte superior de la figura corresponde a la metodología propuesta, la cual consta de las siguientes etapas: generación de instancias bajo el enfoque MIL [Cheplygina et al.. 2015], extracción de características usando CNN más TL (CNN+TL), reducción de dimensiones, cálculo de disimilitud, clasificación con k-NN y finalmente, la localización e interpretación de los defectos. En la parte inferior de la figura se presenta la línea base utilizada como referencia, la cual se compone de la generación de instancias bajo el enfoque MIL realizando extracción de características de bajo nivel y utilizando el clasificador k-NN con disimilitudes de bolsa, como en [Mera et al.. 2016], sin los pasos de localización e interpretación.

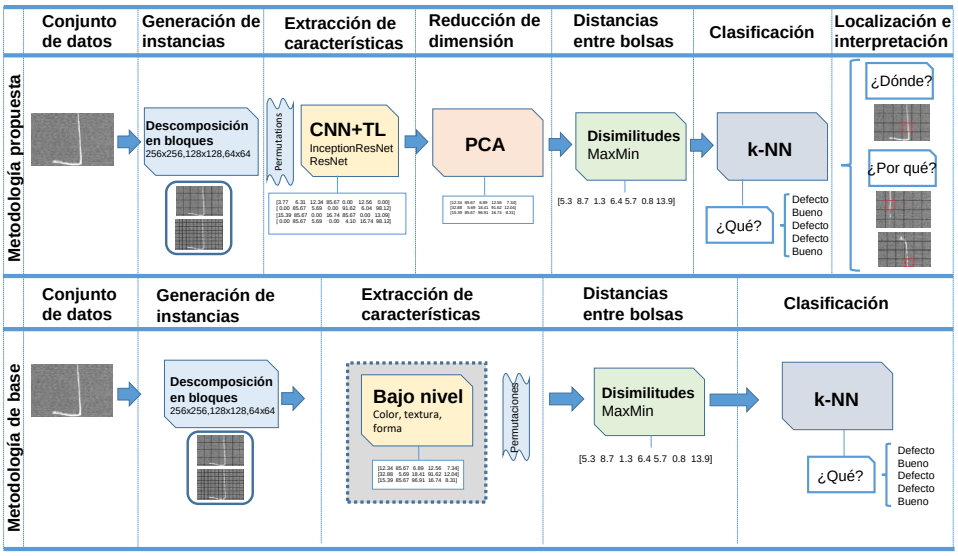


Figura 7-1: Metodologías utilizadas, encontrando en la parte superior la metodología propuesta y en la parte inferior la metodología utilizada como referencia.

7.2.1 Generación de instancias

Teniendo en cuenta que se busca desarrollar una metodología que permita extraer información que sea relevante para representar las imágenes con el fin de realizar su clasificación, detección y posterior interpretación de los defectos, se determinó la necesidad de analizar las imágenes sobre cada una de sus regiones de interés (i.e. las instancias), usando como estrategia para generarlas la descomposición en bloques de cuadrículas [Mera. 2017, p. 48]. Se seleccionó la descomposición en bloques por ser un método simple y de bajo costo, que no requiere información ni algoritmos adicionales, como sí son necesarios al utilizar métodos como el de segmentación utilizado en las Faster R-CNN [Li et al.. 2021a], o el etiquetado débil que requiere el uso de máscaras que previamente debieron ser definidas por un experto como se presentan en [Božič et al.. 2021].

En la descomposición en bloques se utilizan diferentes dimensiones de la cuadrícula de tal manera que, con una estrategia simple y sin necesidad de aplicar costosos algoritmos o procesos de segmentación manual, se pueda a partir de los bloques resultantes realizar la extracción de las características de cada uno mediante CNN con TL o, alternativamente y como referencia de comparación, por medio de la extracción de características de bajo nivel tales como color, textura, contorno, forma, entre otras, posibilitando así ser utilizadas en la clasificación de las imágenes y en la posterior localización de los defectos.

Por tal motivo y teniendo en cuenta las dimensiones de los conjuntos de datos utilizados, se seleccionaron cuadrículas

de 256×256, 128×128 y 64×64, cubriendo así varias posibilidades para la clasificación y localización de los defectos. El tamaño de las cuadrículas se seleccionó como potencias de dos, debido a que las dimensiones de las imágenes de los conjuntos de datos considerados cumplen con dicha condición; no obstante, cualquier tamaño de cuadrícula puede ser utilizado, prefiriendo que sea múltiplo de los tamaños de las imágenes con el fin de evitar el problema de tener regiones de diferentes dimensiones. Si bien la metodología también puede funcionar con bloques de tamaños diferentes, en tal caso se dificultarían los cálculos para la localización e interpretación de los defectos.

En las Figuras **7-2**, **7-3** y **7-4**, se presentan ejemplos de imágenes pertenecientes al conjunto de datos AITEX y a dos versiones del conjunto de datos DAGM respectivamente, en su versión sin cuadrícula, y posteriormente con cuadrículas de 256×256, 128×128 y 64×64 píxeles, que ilustran el proceso de descomposición en bloques.

(a) Imagen original: Bolsa de una única instancia

(b) Cuadrícula de 256×256: Bolsa de 16 instancias

(c) Cuadrícula de 128×128: Bolsa de 64 instancias

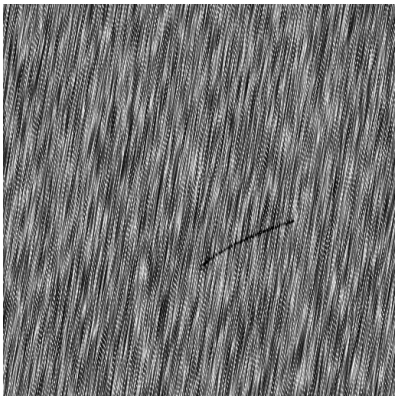
(d) Cuadrícula de 64×64: Bolsa de 256 instancias

Figura 7-2: Imagen con un defecto perteneciente al conjunto de datos AITEX, sin descomposición en la Figura (a), y con diferentes descomposiciones de bloques en las demás figuras.

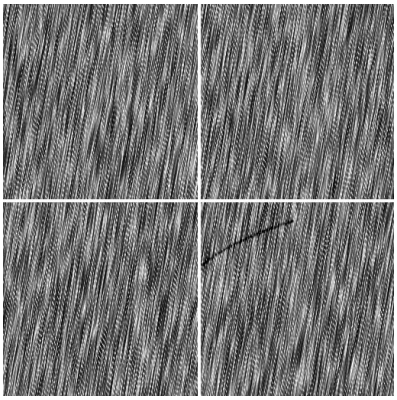
7.2.2 Extracción de características

Una vez que las imágenes se descompusieron en bloques de acuerdo con las cuadrículas y se extrajeron las subimágenes, se utilizaron CNN basadas en TL para realizar la extracción de características. El procedimiento es el siguiente: i) dado que una red neuronal funciona mejor que otra dependiendo del problema, se seleccionaron dos redes basadas en TL con el fin de buscar un buen comportamiento de la caracterización. Las redes seleccionadas fueron ‘‘ResNet50’’ e ‘‘InceptionResNetV2’’, ya que mostraron los mejores resultados en las pruebas preliminares, formando así dos variantes de la metodología propuesta; ii) Se adoptaron las redes seleccionadas, de tal manera que excluyan las últimas capas correspondientes al clasificador pero se incluyan los pesos del aprendizaje previo realizado con las imágenes del conjunto *ImageNet*; iii) Se reemplazaron las etapas del clasificador con un conjunto de capas que convierten el resultado a una matriz unidimensional; iv) A continuación, se entrena la red en sus últimas capas usando algunas épocas (por ejemplo, 5), proporcionándole imágenes del conjunto de entrenamiento. El resultado es una red entrenada que puede generar 2048 características para ResNet50 y 1535 características para InceptionResNetV2; v) Posteriormente, se extraen las características de las instancias. Para ello se utiliza la red entrenada, suministrándole los bloques que fueron divididos en la etapa de generación de instancias de acuerdo con las necesidades y la red seleccionada. Luego, las imágenes de prueba se redimensionaron a 224×224 de manera que puedan ser procesadas por la red neuronal, obteniendo para cada imagen tantos vectores de características como subimágenes se generan según la cuadrícula.

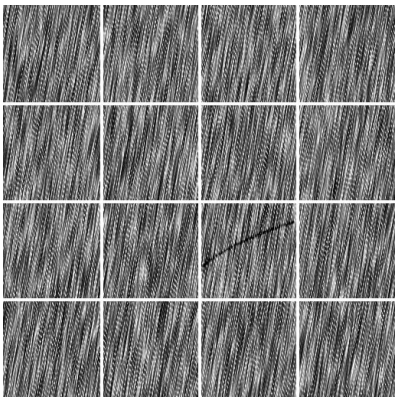
Para la línea de base, con el fin de extraer características de bajo nivel, se emplearon las cincuenta (50) características presentadas por [Mera. 2017] cuyas implementaciones provienen de algoritmos incluidos en la caja de herramientas de Balu para el procesamiento de imágenes [Mery. 2011]. Las 50 características se agrupan de la siguiente manera: 5 características asociadas con el color, 36 características asociadas con patrones binarios locales (LBP) y 9 características basadas en el histograma de gradientes orientados (HoG). El procedimiento es el siguiente: i) Se definen tres conjuntos de características: en el primero, características de intensidad y color; en el segundo, características basadas en LBP; y en el tercero, características basadas en HoG; ii) Luego, cada una de las imágenes se descompone en bloques de



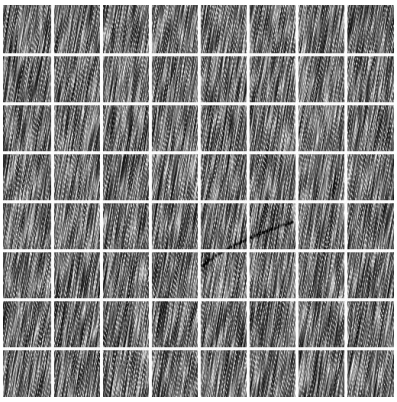
(a) Imagen original: Bolsa de una única instancia



(b) Cuadrícula de 256×256: Bolsa de 4 instancias

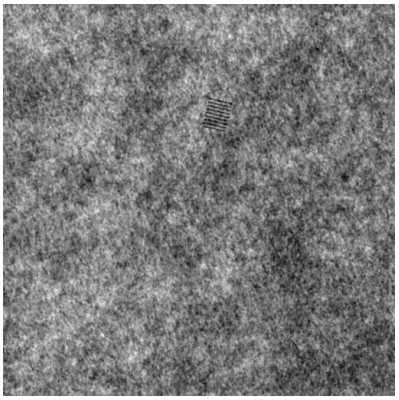


(c) Cuadrícula de 128×128: Bolsa de 16 instancias

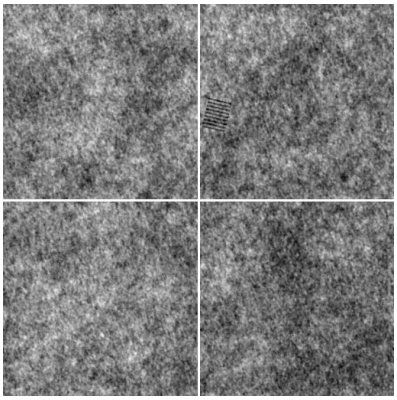


(d) Cuadrícula de 64×64: Bolsa de 64 instancias

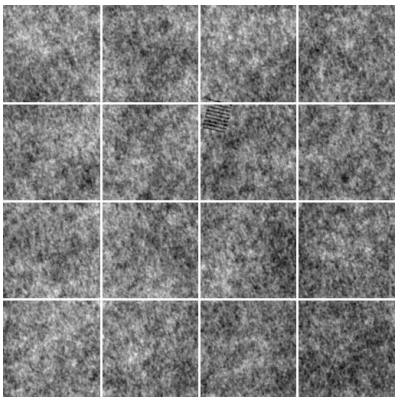
Figura 7-3: Imagen con un defecto perteneciente al conjunto de datos DAGM Clase 2, (a) sin descomposición y (b)—(d) con diferentes descomposiciones de bloques.



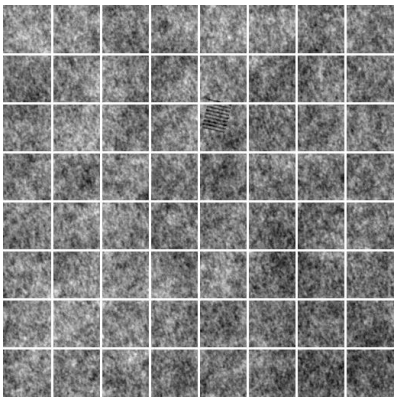
(a) Imagen original: Bolsa de una única instancia



(b) Cuadrícula de 256×256: Bolsa de 4 instancias



(c) Cuadrícula de 128×128: Bolsa de 16 instancias



(d) Cuadrícula de 64×64: Bolsa de 64 instancias

Figura 7-4: Imagen con un defecto perteneciente al conjunto de datos DAGM Clase 5, (a) sin descomposición y (b)—(d) con diferentes descomposiciones de bloques.

acuerdo al tamaño de cuadrícula seleccionado; iii) Para cada uno de los bloques resultantes de la descomposición de las imágenes, se realiza la extracción de características; y iii) Finalmente, el conjunto de características resultante se almacena en una matriz de dimensión 50.

7.2.3 Reducción de dimensión

Dado que queremos hacer una comparación justa de la metodología propuesta con respecto a la metodología de línea de base, hemos tratado de hacer que los dos métodos sean comparables al menos en términos del número de características obtenidas. Esto se debe a que la metodología propuesta con la caracterización basada en CNN+TL genera más de 1500 características (potencialmente muy redundantes) por bloque mientras que, en la metodología de base con la caracterización de bajo nivel propuesta en [Mera. 2017], se obtienen 50 características. Por lo tanto, la dimensión del vector generado por el método CNN+TL se redujo a 50 características mediante la aplicación del Análisis de Componentes Principales (PCA). Así, la cantidad de características proporcionadas por los dos métodos es la misma, permitiendo construir bolsas de imágenes de los mismos tamaños: cada bolsa contiene tantas instancias (vectores) como regiones se obtuvieron de cada imagen en la descomposición en bloques, y cada instancia está compuesta por 50 características.

7.2.4 Distancias entre bolsas

Con el fin de realizar la correspondiente detección, localización e interpretación de los defectos, se seleccionó el algoritmo k-NN por tratarse de un clasificador muy intuitivo y —en general— con desempeños de clasificación competitivos frente a otras técnicas más sofisticadas pero menos interpretables. En la versión de k-NN para MIL con $k = 1$, se busca encontrar la menor disimilitud entre una bolsa, la que se quiere clasificar que pertenece al conjunto de prueba, y las bolsas de ejemplos etiquetadas disponibles en un conjunto de entrenamiento.

Sean $\mathcal{X}_{(i)} = \left\{\vec{X}_{(i,1)}, \ldots, \vec{X}_{(i,k)}, \ldots \vec{X}_{(i,N_i)}\right\}$ y $\mathcal{X}_{(j)} = \left\{\vec{X}_{(j,1)}, \ldots, \vec{X}_{(j,l)}, \ldots \vec{X}_{(i,N_j)}\right\}$ dos bolsas de instancias que se quieren comparar, las siguientes son dos opciones para calcular la disimilitud entre $\mathcal{X}_{(i)}$ y $\mathcal{X}_{(j)}$ de acuerdo con la proposición de [Cheplygina and Tax. 2015] de considerar las bolsas como conjuntos de puntos:

$$d_{maxmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \max_k \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{7-1}$$

$$d_{minmin}(\mathcal{X}_{(i)}, \mathcal{X}_{(j)}) = \min_k \min_l d\left(\vec{X}_{(i,k)}, \vec{X}_{(j,l)}\right) \tag{7-2}$$

La medida de disimilitud presentada en la Ec. (7-1) corresponde a la versión MIL de la distancia de Hausdorff entre conjuntos [van Kreveld et al.. 2022]. La medida de disimilitud de la Ec. (7-2), es decir d_{minmin} es una medida simétrica que también puede ser útil para resolver el problema de encontrar la disimilitud entre dos bolsas.

Según Cheplygina and Tax [2015], las medidas de disimilitud más frecuentemente utilizadas para el cálculo de la distancia entre bolsas son d_{maxmin} y d_{minmin}; sin embargo, esta última es menos sensible a los valores atípicos que la primera, lo que la hace útil cuando los conjuntos de datos contienen valores atípicos. Se descartaron las medidas $d_{meanmin}$ y $d_{meanmean}$, debido a que el cálculo de la media que realizan no permite determinar a qué instancia corresponde la menor distancia obtenida, la cual es requerida para la localización e interpretación de los defectos. Adicionalmente, en las pruebas preliminares que se realizaron, los cálculos con d_{maxmin} produjeron mejores resultados que con d_{minmin}. Por esta razón, en la metodología propuesta se usó únicamente la distancia d_{maxmin}.

En MIL existen dos diferentes enfoques asociados a la comparación de bolsas de acuerdo con el tipo de objetos comparados mediante una medida de disimilitud, a saber: i) entre bolsas y prototipos de bolsas y ii) entre bolsas y prototipos de instancias. El primer enfoque genera como resultado una representación con una relativa baja dimensión, determinada por el número de bolsas de entrenamiento; en contraste, el segundo enfoque da como resultado una representación con una relativa alta dimensión determinada por el número total de instancias en el conjunto de

entrenamiento [Cheplygina et al.. 2016]. En nuestra metodología propuesta, la comparación de dos bolsas $\mathcal{X}_{(i)}$ y $\mathcal{X}_{(j)}$ se realiza entre bolsas y prototipos de instancias, siendo necesario comparar cada instancia de la primer bolsa $\mathcal{X}_{(i)}$, contra la totalidad de las instancias de la segunda bolsa $\mathcal{X}_{(j)}$, tal como se aprecia en la Figura **7-5**.

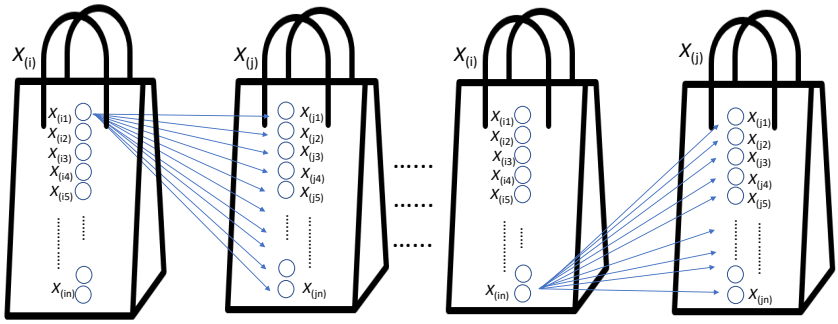


Figura 7-5: Comparación entre dos bolsas a nivel de instancias.

7.2.5 Detección de defectos

Una vez definido el procedimiento para el cálculo de las disimilitudes entre las imágenes que están representadas por bolsas de instancias, se procedió a realizar la detección de defectos en las imágenes de la siguiente manera: Para cada bolsa $\mathcal{X}_{(i)}$ en el conjunto de prueba, se calcula la distancia d_{maxmin} con respecto a todas las bolsas en el conjunto de entrenamiento, cada una de ellas denotada como $\mathcal{X}_{(j)}$. A continuación, se usa el algoritmo k-NN, con $k = 3$ en nuestro caso, para encontrar las tres bolsas de entrenamiento vecinas más cercanas a la bolsa de prueba. Por último, mediante voto mayoritario entre las etiquetas de las bolsas de entrenamiento más cercanas, se establece la etiqueta de clase predicha para la bolsa de prueba.

Como se mencionó anteriormente, se utilizó el algoritmo k-NN por ser simple y fácil de aplicar, el cual puede ser utilizado tanto para resolver problemas de clasificación como de regresión. Adicionalmente, se quiere dar peso y valor al proceso de extracción de características y al cálculo de las disimilitudes entre bolsas para la solución del problema, y no esperar que un buen clasificador como las SVM o las CNN resolvieran el problema escondiendo potenciales falencias presentadas en las etapas previas del sistema de reconocimiento.

El algoritmo correspondiente que compara dos bolsas mediante la función d_{maxmin} incluye dos ciclos anidados sobre el número de instancias; ver Algoritmo 1.

7.2.6 Localización de defectos

El cálculo de las disimilitudes entre bolsas y la aplicación del algoritmo de clasificación k-NN dan como resultado las etiquetas de clase predichas para las imágenes de prueba; por consiguiente, las etapas descritas en las Secciones 7.2.4 y 7.2.5 permiten dar respuesta a la pregunta: *"¿Qué es esto?"*. El siguiente interrogante a resolver por el sistema de reconocimiento automático es: *"¿Donde está el defecto?"*.

Para responder a esta pregunta, se analizó la estructura e información retornada por la función d_{maxmin} al comparar dos bolsas de instancias, ver Listado 1. Nótese que, además de la disimilitud calculada, sería sencillo averiguar qué instancia de la bolsa $\mathcal{X}_{(i)}$ que se está clasificando es la responsable de arrojar la menor distancia entre las dos bolsas

Algoritmo 1 Cálculo de la disimilitud d_{maxmin} regresando la distancia más pequeña entre dos bolsas.

```
function dissMaxMin(Xi, Xj)
    distExter ← []
    for k ← 1 to len(Xi) do
        distInter ← []
        for l ← 1 to len(Xj) do
            distInter ← distInter + norm(Xi[k] − Xj[l])
        end for
        distExter ← distExter + min(distInter)
    end for
    return max(distExter)
end function
```

comparadas, pudiéndose así deducir que si la etiqueta predicha fuera de un defecto, dicha instancia —que corresponde a una determinada región (bloque) de la imagen— tendría altas probabilidades de contener el defecto. Similarmente, puesto que se está utilizado el algoritmo k-NN con k=3 vecinos, las instancias responsables de generar esas k distancias menores, correspondientes a porciones de la imagen, deberán contener el defecto asociado a la imagen en dicha bolsa.

El algoritmo correspondiente que compara dos bolsas mediante la función d_{maxmin} que regresa no solo la distancia calculada sino también la posición de la instancia de la bolsa de prueba que genera dicho valor, se presenta en el Algoritmo 2.

Algoritmo 2 Cálculo de la disimilitud d_{maxmin} regresando la distancia más pequeña entre dos bolsas.

```
function dissMaxMin(Xi, Xj)
    distExter ← []
    for k ← 1 to len(Xi) do
        distInter ← []
        for l ← 1 to len(Xj) do
            distInter ← distInter + norm(Xi[k] − Xj[l])
        end for
        distExter ← distExter + min(distInter)
    end for
    valuMaxi ← max(distExter)
    posiMaxi ← arg max(distExter)
    return valuMaxi, posiMaxi
end function
```

En cuanto a la localización de los defectos, es importante tener en cuenta que al descomponer en bloques las imágenes según las diferentes resoluciones de cuadrícula consideradas, los defectos pueden quedar totalmente incluidos, parcialmente incluidos o excluidos de la cuadrícula debido a que la posición de la cuadrícula es fija al interior de la imagen. Por tanto, se considera que un defecto en una imagen está bien localizado si está totalmente o parcialmente incluido en el recuadro de la cuadrícula. Adicionalmente, con el fin de determinar el grado de certeza en la localización de los defectos y con base en los posibles resultados del algoritmo de k-NN con $k = 3$, se procedió a presentar las siguientes graduaciones de localización del defecto: i) completamente localizado, cuando las tres predicciones están bien localizadas; ii) adecuadamente localizado, cuando dos de las predicciones están bien localizadas; iii) deficientemente localizado, cuando una sola predicción está bien localizada; y por último, iv) no localizado, cuando ninguna de las tres predicciones contienen el defecto o están mal localizadas.

Puesto que uno de los objetivos de la metodología propuesta es brindar un reporte gráfico y directo para el usuario del sistema de reconocimiento, se adoptó la siguiente convención de visualización de los bloques de las imágenes que están asociados a la ubicación de los defectos: pintar las tres regiones con defectos —que corresponden a los tres vecinos calculados con el algoritmo de k-NN— con diferentes colores (verde, rojo y azul), siendo verde el que corresponde a la menor distancia, rojo para la segunda menor distancia y azul para la tercera menor distancia. Adicionalmente, cuando dos o tres regiones coinciden y con el fin de evitar la superposición de las mismas, los recuadros de los bloques

se dibujan con tamaños diferentes, cada vez menores, logrando así una adecuada visualización.

7.2.7 Interpretación de defectos

Una vez resueltos los problemas de detección y localización del defecto y para una metodología como la nuestra que prescinde del uso de máscaras de entrenamiento o anotaciones, resta responder únicamente —en la medida de lo posible— al interrogante *"¿Por qué esa etiqueta se asignó a ese objeto?"*. De nuevo, para responder esta pregunta también resulta útil la estructura y naturaleza de la función d_{maxmin}, la cual podría regresar no solamente la distancia y la posición de la instancia de la bolsa de prueba, sino también una identificación de cuáles bolsas del conjunto de entrenamiento y específicamente con cuáles k instancias de dichas bolsas se obtienen las k distancias mínimas. Así, a partir de dicha información, se puede establecer la porción de cada imagen que es responsable y contribuye, de alguna manera, a la predicción de la etiqueta.

El algoritmo modificado que compara dos bolsas mediante la función d_{maxmin} que regresa no solo la distancia calculada sino también la posición de la instancia de la bolsa de prueba que genera dicho valor, se presenta en el Algoritmo 3.

Algoritmo 3 Cálculo de la disimilitud d_{maxmin} entre dos bolsas devolviendo la menor distancia y las posiciones de las instancias responsables de la menor distancia, en las bolsas de prueba y entrenamiento.

function DISSMAXMINPOS(Xi, Xj)
  $distExter \leftarrow []$
  $posiExter \leftarrow []$
  for $k \leftarrow 1$ to $len(Xi)$ **do**
    $distInter \leftarrow []$
    for $l \leftarrow 1$ to $len(Xj)$ **do**
      $distInter \leftarrow distInter + norm(Xi[k] - Xj[l])$
    end for
    $valuMini \leftarrow min(distInter)$
    $posiMini \leftarrow \arg\min(distInter)$
    $posiExter \leftarrow posiExter + posiMini$
    $distExter \leftarrow distExter + valuMini$
  end for
  $valuMaxi \leftarrow max(distExter)$
  $posiMaxi \leftarrow \arg\max(distExter)$
  $posiMini \leftarrow posiExter[posiMaxi]$
  return $valuMaxi, posiMaxi, posiMini$
end function

La información resultante de las etapas de localización e interpretación se presenta en una secuencia de cuatro imágenes, lo cual permite explicar visualmente el motivo de la asignación de una etiqueta de clase a una determinada bolsa de prueba. La primera imagen es aquélla correspondiente a la bolsa de prueba, en la cual se enmarcan las tres regiones que fueron retornadas como más similares según el algoritmo k-NN y que posiblemente contienen los defectos. Para demarcar estas regiones se utiliza la misma notación de colores definida en la Sección 7.2.6: verde corresponde a la región asociada a la menor distancia; rojo indica la región de la segunda menor distancia y, por último, azul corresponde a la tercer menor distancia. En las tres visualizaciones restantes del reporte gráfico se presentan imágenes pertenecientes al conjunto de entrenamiento, siendo la primera de ellas la imagen con la cual se obtuvo la menor distancia, junto con su región del defecto enmarcada en color verde; la segunda imagen es aquélla que resultó de la segunda menor distancia, con su región del defecto marcada en color rojo; finalmente, la última imagen corresponde a la tercer menor distancia, con el defecto demarcado en color azul.

La interpretación de los resultados proporcionados por nuestra metodología propuesta se plantea entonces en términos de la verificación visual de si, en efecto, cada una de las tres regiones con defectos en la imagen de prueba corresponden a regiones similares en cada una de las tres imágenes del conjunto de entrenamiento. Tras esta serie de visualizaciones,

se puede confirmar que la región de un determinado color (verde, rojo o azul) de la imagen de prueba es similar a la región del mismo color en la imagen de entrenamiento, tal como se puede ver en la Figura **7-6**, explicándose así el porqué se asignó una determinada etiqueta a una imagen de prueba.

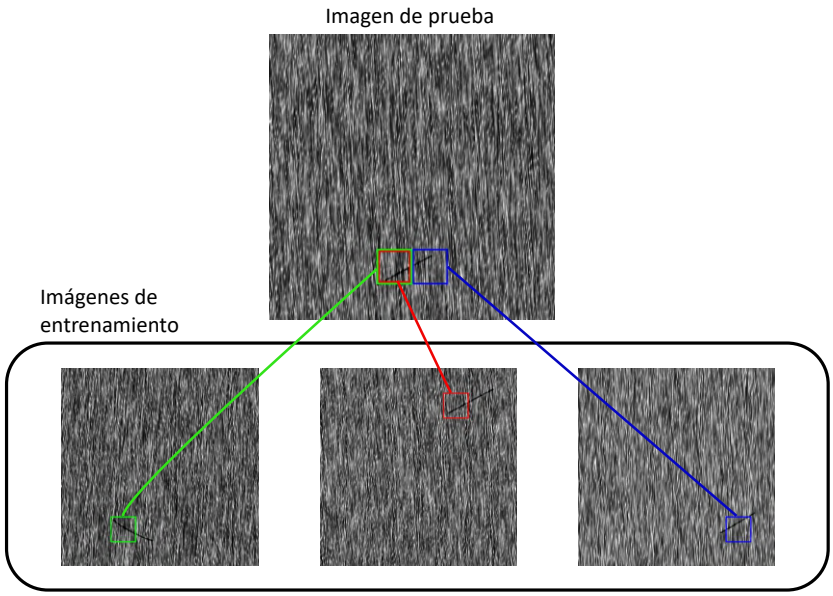


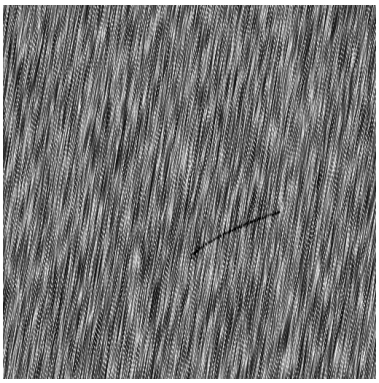
Figura 7-6: Interpretación gráfica de la localización de un defecto en la imagen *19.png* perteneciente al conjunto de datos DAGM C2.

Una vez obtenida la predicción de la etiqueta y las k menores distancias con sus responsables en las bolsas de prueba y entrenamiento, es posible realizar el siguiente análisis: dada una bolsa correspondiente a una imagen de prueba que se quiere predecir y recibe como resultado una etiqueta de imagen defectuosa, proveniente de k etiquetas de k vecinos en el conjunto de entrenamiento en su gran mayoría defectuosos, las posiciones obtenidas por la función d_{maxmin} pertenecientes a la bolsa de prueba etiquetadas como defectuosas se pueden interpretar como las posiciones donde se encuentran las fallas o defectos en dicha imagen. Además, las posiciones retornadas de las bolsas de entrenamiento se pueden interpretar como que el defecto predecido en la bolsa de prueba se explica debido a que existen k instancias pertenecientes a k bolsas de entrenamiento que contienen porciones similares y que, en dichas posiciones, se encuentran las instancias que pueden corresponder a defectos o características similares sobre las cuales se basó la respuesta.

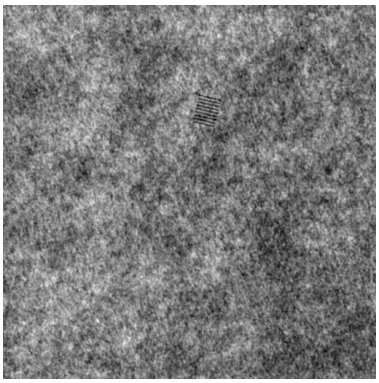
7.3 Configuración experimental

7.3.1 Conjuntos de datos

Con el fin de evaluar experimentalmente la metodología propuesta se utilizaron tres conjuntos de datos de diferente naturaleza, dimensiones y forma, a saber: i) dos conjuntos de datos sintéticos: DAGM Clase 2 y DAGM Clase 5, que se denominarán de aquí en adelante DAGM C2 y DAGM C5, respectivamente, conformados por imágenes cuadradas y con defectos similares, útiles para la detección de defectos en superficies texturizadas, y ii) un conjunto de datos de telas (AITEX), en el cual las imágenes son de forma rectangular y presentan diferentes tipos de defectos. A continuación se brinda una descripción breve de las propiedades de los tres conjuntos.



(a) Imagen del conjunto de datos DAGM-C2



(b) Imagen del conjunto de datos DAGM-C5

(c) Imagen del conjunto de datos AITEX

Figura 7-7: Ejemplos de imágenes con defectos pertenecientes a : (a) conjunto de datos DAGM-C2, (b) conjunto de datos DAGM-C5 y, finalmente, (c) conjunto de datos AITEX.

Tabla 7-1: Descripción de los conjuntos de datos utilizados en los diferentes experimentos.

Conjunto de datos	**Dimensiones**	**Defectuosos**	**No defectuosos**
DAGM C2	512×512	150	1000
DAGM C5	512×512	150	1000
AITEX	4096×256	105	140

Conjuntos de datos DAGM

Se trata de varios conjuntos de datos sintéticos utilizados para la detección de defectos en superficies texturizadas. Se crearon originalmente para un concurso[1] en el Simposio Anual de la DAGM, entre ellos se destacan los conjuntos DAGM C2 y DAGM C5. Estos conjuntos fueron seleccionados para evaluar la metodología propuesta porque contienen imágenes de defectos sobre fondos artificiales de apariencia difusa, particularmente líneas de color negro con ubicación aleatoria —bien sea rectas o curvas— en el conjunto DAGM C2 (ver Figura **7-7**a) y patrones conformados por entre 7 y 10 líneas rectas —paralelas y de similar longitud— en el conjunto DAGM C5 (ver Figura **7-7**b). Cada conjunto de datos consta de 1150 imágenes con una resolución de 512×512 píxeles, 150 de las cuales presentan defectos mientras que las restantes 1000 imágenes están libres de defectos.

Conjunto de datos AITEX

Es un conjunto de datos desarrollado por la AITEX, el cual está compuesto por 245 imágenes provenientes de 7 telas diferentes [Silvestre-Blanes et al.. 2019]. Hay 140 imágenes que no contienen defectos, 20 por cada tipo de tejido, y 105 imágenes con diferentes tipos de defectos; todas las imágenes tienen un tamaño de 4096×256 píxeles. Los defectos pueden ser nudos, arrugas, hilos rotos, contaminación, fallos de trama, entre otros, los cuales se observan en la tela como líneas curvas o rectas —ya sean blancas, negras o del color del textil— o puntos y manchas, principalmente. Sin embargo, en este trabajo sólo consideramos el problema de clasificación de defecto/no defecto.

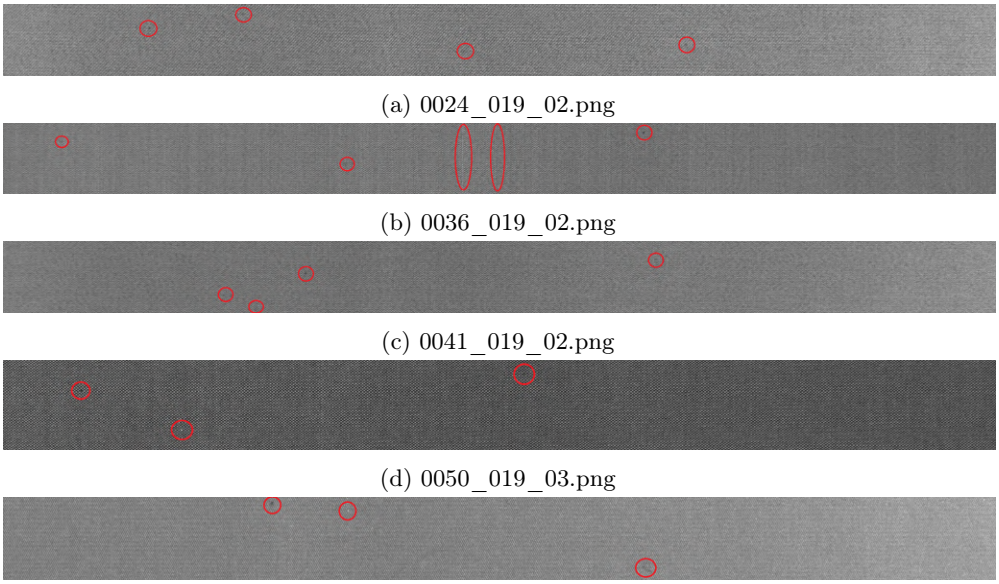
Todas las imágenes presentan una sección en el lado izquierdo de la misma que no pertenece a la tela, la cual no es uniforme en cuanto a su tamaño, composición y textura; ver el ejemplo mostrado en la Figura **7-2**. Por lo tanto, fue necesario recortar todas las imágenes con el objetivo de eliminar dicha sección, puesto que su presencia dificultaba llevar a cabo un adecuado proceso de clasificación. El corte se realizó agrupando las imágenes según las 8 diferentes telas de las que fueron extraídas. Nótese que, en el sitio web[2] de descarga del conjunto de datos, definen que las imágenes provienen de 7 telas; sin embargo, en los archivos correspondientes hay una octava tela compuesta por una sola imagen. Con el fin de facilitar la posterior descomposición en bloques de las imágenes, se realizaron cortes iguales para las imágenes de cada tela en anchuras que fueran múltiplos de 256 píxeles. Los cortes ejecutados por cada tela se presentan en la Tabla **7-2**.

Tabla 7-2: Cortes realizados al conjunto de datos AITEX.

Tela	Cantidad	Recorte	Nueva dimensión
0	27	1280	2816×256
1	32	512	3584×256
2	58	512	3584×256
3	51	1280	2816×256
4	30	768	3328×256
5	20	256	3840×256
6	26	1024	3072×256
7	1	1280	2816×256

Nótese que, en un gran porcentaje de las imágenes que presentan defectos en el conjunto de datos AITEX, éstos son difíciles de identificar visualmente incluso para un experto humano. Esto se debe a que algunos de los defectos abarcan la totalidad de la imagen y otros se confunden con las tramas de los tejidos; en contraste, otros defectos están conformados por unos pocos píxeles. Además, algunas imágenes presentan varias anormalidades que podrían ser consideradas como defectos por un experto humano, tal como se evidencia en las anotaciones que de forma manual se hicieron a algunas imágenes y que se muestran en la Figura **7-8**.

[1] https://www.kaggle.com/datasets/mhskjelvareid/dagm-2007-competition-dataset-optical-inspection
[2] https://www.aitex.es/afid/



(a) 0024_019_02.png

(b) 0036_019_02.png

(c) 0041_019_02.png

(d) 0050_019_03.png

(e) 0055_019_06.png

Figura 7-8: Ejemplos de imágenes del conjunto de datos AITEX con defectos poco visibles o ambiguos, los cuales fueron etiquetados manualmente.

7.3.2 Plataformas de cómputo, parámetros y medidas de desempeño

Los experimentos se realizaron en un servidor DELL con 6 núcleos (Intel Xeon$^{(R)}$ CPU E5-2643 v3 @ 3.40GHz) y 12 hilos, con una tarjeta GPU Tesla K40c, bajo sistema operativo Ubuntu release 18.04 (bionic) con Python 3.8.5 como lenguaje de desarrollo; Keras 2.3.1 y TensorFlow 2.4.1 como librerías de apoyo. La extracción de características de bajo nivel se realizó en un servidor DELL con 24 núcleos (Intel Xeon$^{(R)}$ CPU E7-4860 v2 @ 2.60GHz), bajo Scientific Linux release 7.4 (Nitrogen) y usando MATLAB R2018b (9.5.0.944444) como lenguaje de programación.

La descomposición de las imágenes en bloques se realizó seleccionando tres tamaños de cuadrícula diferentes (256×256, 128×128 y 64×64) y, para la extracción de características con CNN+TL, se seleccionaron dos CNN diferentes basadas en TL: ResNet50 e InceptionResNetV2. El clasificador seleccionado fue el algoritmo k-NN, utilizando la medida de disimilitud entre bolsas d_{maxmin} y $k = 3$ por ser éste el menor número impar mayor que uno, así como por facilidad en la presentación de los resultados de interpretación; sin embargo, el número de vecinos considerados podría ser mayor. Con todas las variantes de configuraciones experimentales se efectuaron cinco permutaciones —para entrenamiento y prueba repetido— diferentes sobre cada uno de los conjuntos de datos, estimando varias medidas de desempeño de clasificación como exactitud, precisión, exhaustividad, valor-F, matrices de confusión, curvas basadas en Características Operativas del Receptor (ROC, por su sigla en inglés), así como reportando gráficas y diferentes conteos sobre la localización e interpretación de los defectos.

7.3.3 Experimentos

A continuación se presentan dos tipos de experimentos, diferenciando entre los que se hacen con la metodología propuesta y los que utilizan la metodología de referencia.

Experimentos con la metodología propuesta

Para aquellos experimentos que hacen uso de la metodología propuesta, se definió la configuración experimental que puede verse en la parte superior de la Figura **7-1**. Nótese que, para cada conjunto de datos, se realiza en primer lugar la descomposición en bloques según la rejilla seleccionada y, posteriormente, se definen 5 permutaciones diferentes distribuyendo las imágenes en entrenamiento y prueba, según la permutación seleccionada, con una proporción de 80 % para entrenar las dos variantes de CNN+TL y el 20 % restante reservado para prueba. La extracción de características mediante CNN+TL se realiza en los conjuntos de entrenamiento y prueba por separado, haciendo uso de la red previamente entrenada. A continuación, procedemos a reducir la dimensión de los vectores de características generados aplicando PCA a cada permutación de los conjuntos de prueba y entrenamiento. La matriz de proyección PCA se estima utilizando únicamente el conjunto de entrenamiento y, a continuación, se aplica tanto al conjunto de entrenamiento como al conjunto de prueba. Por último, las 50 características de cada bloque se reciben con las respectivas etiquetas de bolsa y, a continuación —con el algoritmo k-NN y la disimilitud d_{maxmin}— se lleva a cabo la clasificación de los objetos, la localización de defectos y la interpretación de las imágenes del conjunto de prueba con respecto a las imágenes del conjunto de entrenamiento.

Experimentos con la metodología de referencia

Para los experimentos con la metodología de referencia, se definió una variación de la anterior configuración; véase la parte inferior de la Figura **7-1**. En primer lugar, para cada conjunto de datos, se realiza la descomposición en bloques de acuerdo con la cuadrícula seleccionada y, posteriormente, se realiza la extracción de las 50 características de bajo nivel a cada bloque de la imagen. A continuación, se definen 5 permutaciones diferentes del conjunto de datos, distribuyendo las imágenes en entrenamiento y prueba, según la permutación seleccionada, con una proporción de 80 % para entrenamiento y 20 % para prueba. Finalmente, se reciben los conjuntos de entrenamiento y prueba, donde cada imagen está representada por una bolsa de instancias con sus respectivas etiquetas; a partir de esta información, se utiliza el algoritmo k-NN con la disimilitud d_{maxmin} para realizar la clasificación de las imágenes de prueba con respecto a las imágenes de entrenamiento.

7.4 Resultados y discusiones

Una vez realizadas las ejecuciones de los experimentos sobre cinco (5) permutaciones, utilizando las tres resoluciones de cuadrícula y los dos métodos de extracción de características, se obtuvieron los siguientes resultados para cada uno de los conjuntos de datos:

7.4.1 Resultados con los conjuntos de datos DAGM C2 y DAGM C5

Los resultados para los conjuntos de datos DAGM C2 y DAGM C5 se analizaron conjuntamente, debido a que corresponden a problemas de similar naturaleza, todas sus imágenes son del mismo tamaño (512×512 píxeles) y tienen la misma cantidad de imágenes con idénticas proporciones entre defectuosas y no defectuosas.

Detección de defectos

Los resultados de exactitud de clasificación, presentados en la Tabla **7-3**, demuestran que la extracción de características usando CNN+TL genera una representación más discriminante que la que se obtiene mediante la extracción de características de bajo nivel. Se destaca que, para el conjunto de datos DAGM C2, la mejor exactitud (0.9722) se logra utilizando la red InceptionResNetV2 como extractor de características sobre imágenes divididas en una cuadrícula de 256×256; en contraste, para el conjunto de datos DAGM C5, el mejor resultado (0.9817) se obtiene utilizando

ResNet50 como extractor de características aunque también sobre una cuadrícula de 256×256. Los resultados más bajos fueron los obtenidos con extracción de características de bajo nivel y utilizando una cuadrícula de 128×128: 0.8400 para el conjunto DAGM C2 y 0.8774 para el conjunto DAGM C5.

Tabla 7-3: Resultados de exactitud para los conjuntos de datos DAGM C2 y DAGM C5 según la cuadrícula.

(a) Conjunto DAGM C2

	Exactitudes según la cuadrícula		
Extr. Caract.	256×256	128×128	64×64
InceptionResNetV2	**0.9722**	0.9565	0.9548
ResNet50	0.9661	0.9574	0.9679
Bajo Nivel	0.8487	**0.8400**	0.8417

(b) Conjunto DAGM C5

	Exactitudes según la cuadrícula		
Extr. Caract.	256×256	128×128	64×64
InceptionResNetV2	0.9687	0.9678	0.9617
ResNet50	**0.9817**	0.9617	0.9626
Bajo Nivel	0.8801	**0.8774**	0.9357

La exactitud no es la medida de desempeño más apropiada cuando existe gran desbalance entre clases, como el que se presenta en estos dos conjuntos datos (87 % sin defecto y 13 % con defecto). Para una mejor estimación del desempeño de clasificación en casos de desbalance de clases, Bramer [2020] sugiere calcular la tasa de verdaderos positivos (TP-rate) y la tasa de falsos positivos (FP-rate), donde la clase positiva (P) es aquélla definida como objetivo del sistema de detección —con defecto, en nuestro caso— y la clase negativa (N) es la contraria: sin defecto en nuestro caso. Las medidas TP-rate y FP-rate son mejores estimadores del desempeño porque no dependen de los tamaños relativos de los subconjuntos de P y N. Los resultados de TP-rate y FP-rate para los dos conjuntos DAGM, para las diferentes variantes de extracción de características y resoluciones de descomposición en bloques, se presentan en la Tabla **7-4**.

Tabla 7-4: TP-rate y FP-rate para los conjuntos de datos DAGM C2 y DAGM C5.

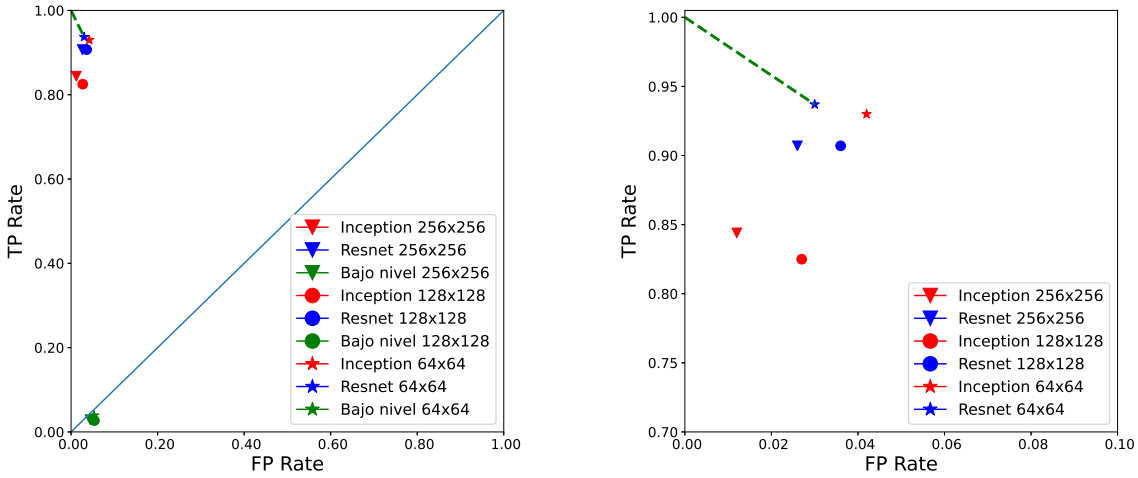
		DAGM C2		**DAGM C5**	
Ext. caract.	**Tamaño cuadr.**	**TP-rate**	**FP-rate**	**TP-rate**	**FP-rate**
Propuesta(Inception)	256×256	0.8440	**0.0118**	0.8186	0.0106
Propuesta(ResNet50)	256×256	0.9066	0.0264	0.8895	**0.0060**
Línea base	256×256	0.0288	**0.0440**	0.2483	0.0364
Propuesta(Inception)	128×128	0.8249	0.0266	0.9886	0.0342
Propuesta(ResNet50)	128×128	0.9066	0.0362	0.9682	0.0391
Línea base	128×128	0.0268	0.0540	0.3186	0.0481
Propuesta(Inception)	64×64	0.9297	0.0422	0.9886	0.0412
Propuesta(ResNet50)	64×64	**0.9374**	**0.0296**	**1.0000**	0.0423
Línea base	64×64	**0.0395**	0.0530	**0.6407**	**0.0245**

Se puede contemplar que, en concordancia con lo observado para la exactitud, los mejores resultados obtenidos para estas medidas de desempeño siguen siendo aquellos obtenidos con extractores de características basados en CNN+TL. En particular, el mayor TP-rate para los dos conjuntos de datos se obtiene utilizando extracción de características con ResNet50 en cuadrículas de 64×64: TP-rates de 0.9374 para el conjunto de datos DAGM C2 y 1.0000 para el conjunto de datos DAGM C5. En cuanto al FP-rate, los mejores resultados en el conjunto de datos DAGM C2 se encuentran al utilizar InceptionResNetV2 sobre una cuadrícula de 256×256, alcanzando un valor de 0.0118; para el conjunto de datos DAGM C5, el mejor resultado es de 0.0060, con ResNet50 como extractor de características sobre una cuadrícula de 256×256.

En contraste, los resultados de TP-rate y FP-rate con la metodología de base difieren significativamente respecto de los obtenidos con la medida de desempeño de exactitud; cf. Tabla **7-3**. Para el caso del conjunto de datos DAGM

C2, el TP-rate más alto es de 0.0395 con una cuadrícula de 64×64 y el FP-rate más bajo es de 0.0440 sobre una cuadrícula de 256×256; esto demuestra la poca efectividad de la metodología de base con extracción de características de bajo nivel para detectar anormalidades o cambios de textura, especialmente cuando los defectos no están bien delimitados y el entorno es bastante difuso. Por otro lado, los resultados para el conjunto de datos DAGM C5 son mejores que los obtenidos para el conjunto DAGM C2, siendo éstos de 0.6407 para el TP-rate y de 0.0245 para el FP-rate, ambos para una cuadrícula de 64×64 y revelando que esta configuración permitiría clasificar bien las imágenes sin defectos y razonablemente bien las que contienen defectos aunque, en todo caso, están alejados de los desempeños logrados cuando se usa la metodología propuesta.

Los resultados combinados de TP-rate y FP-rate se pueden presentar mediante una variación de las curvas ROC –como lo propone Bramer [2020, Chap. 12] y se ha planteado ya en la Sección 3.2.2– donde el mejor clasificador es aquél con la menor distancia al clasificador perfecto, tal como se aprecia para el conjunto de datos DAGM C2 en la Figura **7-9**a, donde se presenta la comparación con las dos variantes de la metodología propuesta y la metodología de línea base. En cambio, en la Figura **7-9**b, sólo se presentan los resultados con la metodología propuesta, debido a los pobres resultados obtenidos con la línea de base.



(a) Metodología propuesta (ResNet50 e InceptionResNetV2), y metodología de referencia (bajo nivel)

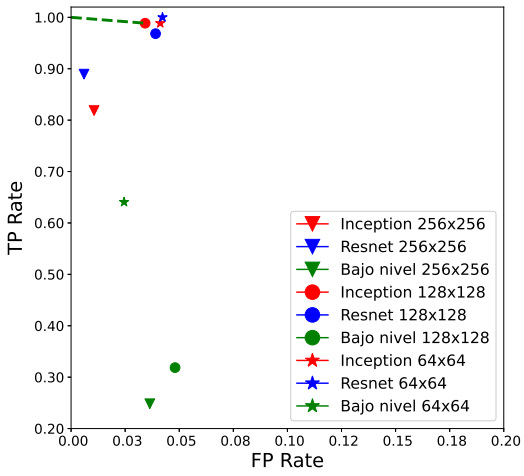
(b) Metodología propuesta (ResNet50 e InceptionResNetV2)

Figura 7-9: Comparación de clasificadores para el conjunto de datos DAGM C2, mostrando el mejor clasificador y su distancia al clasificador perfecto. En la Figura (a), con las dos metodologías; y en la Figura (b), solo con la metodología propuesta.

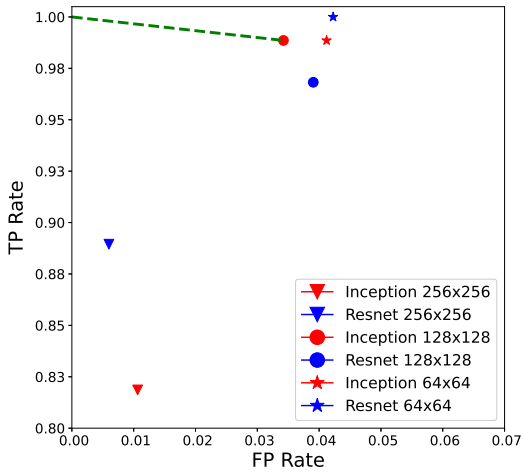
A partir de los gráficos ROC puede observarse que el mejor clasificador para el conjunto de datos DAGM C2 se consigue utilizando la variante Resnet50 de la metodología propuesta utilizando una cuadrícula de 64×64 y una extracción de características con Resnet50, para el cual se obtuvieron valores de TP-rate y FP-rate de 0.9374 y 0.0296, respectivamente, con una exactitud de 0.9679. Adicionalmente, se puede apreciar que el TP-rate es más alto cuando las cuadrículas son más pequeñas (64×64) y la extracción de características se lleva a cabo mediante CNN+TL, permitiendo así clasificar mejor las imágenes con defectos. En contraste, los clasificadores que utilizan Resnet50 como extracción de características son los que mejor balance presentan entre TP-rate y FP-rate, obteniendo mejores resultados en las diferentes medidas de desempeño y, por consiguiente, permitiendo lograr un mejor desempeño promedio de clasificación.

Análogamente, en la Figura **7-10** se presenta una gráfica ROC de la comparación con respecto al clasificador perfecto para el conjunto de datos DAGM C5. Nótese que el mejor clasificador se consigue utilizando la metodología propuesta con InceptionResNetV2 en una cuadrícula de 128×128, el cual obtuvo valores para TP-rate y FP-rate de 0.9886 y 0.0342, respectivamente, con una exactitud de 0.9678. En la Figura **7-10**a se presenta una comparación de las dos metodologías, mientras que en la Figura **7-10**b se han omitido los resultados obtenidos con la metodología de referencia porque corresponden a valores muy alejados del mejor clasificador. De forma similar al comportamiento

observado para el conjunto de datos DAGM C2, en este caso los clasificadores de la metodología de referencia que utilizan la extracción de características de bajo nivel no obtienen buenos resultados, excepto el clasificador que utiliza una cuadrícula de 64×64, para el que se obtuvo un FP-rate de $0,6407$ y un FP-rate de $0,0245$. En cuanto a los demás clasificadores, el uso de la cuadrícula más pequeña (64×64) con CNN+TL como extractor de características da mejores resultados para clasificar las imágenes defectuosas, es decir, TP-rate más altas, mientras que la resolución más grande (256×256) es más útil para clasificar las imágenes sin defectos.



(a) Metodología propuesta (variantes ResNet50 e InceptionResNetV2) y metodología de referencia (bajo nivel)



(b) Metodología propuesta (variantes ResNet50 e InceptionResNetV2)

Figura 7-10: Comparación de clasificadores para el conjunto de datos DAGM C5, mostrando el mejor clasificador y su distancia al clasificador perfecto. En la Figura (a), con las dos metodologías; y en la Figura (b), sólo con la metodología propuesta.

Localización

La capacidad de localización de los defectos de la metodología propuesta se ilustra a continuación para una de las permutaciones de prueba, usando el mejor clasificador que se identificó en la subsección anterior para cada uno de los conjuntos de datos DAGM, a saber: i) extracción de características utilizando Resnet50 sobre una cuadrícula de 64×64 para el conjunto de datos DAGM C2 y ii) extracción de características bajo InceptionResNetV2 con cuadrícula de 128×128 para el caso del conjunto de datos DAGM C5. Los resultados de clasificación para una permutación, en cada conjunto de datos, se presentan en términos de las correspondientes matrices de confusión en la Tabla **7-5**, así como en los reportes de clasificación presentados en la Tabla **7-6**. Nótese, en estas últimas dos tablas, que la clasificación para los dos conjuntos de datos DAGM se realiza sobre 230 imágenes: 35 con defecto y 195 sin defecto.

Tabla 7-5: Matrices de confusión para una permutación del mejor clasificador en los conjuntos de datos DAGM C2 (izquierda) y DAGM C5 (derecha).

		Predicha	
	DAGM C2	**Defecto**	**No defecto**
Actual	Defecto	34	1
	No defecto	8	187

		Predicha	
	DAGM C5	**Defecto**	**No defecto**
Actual	Defecto	33	2
	No defecto	4	191

Para el conjunto de datos DAGM C2 se observó un valor de 0.8095 para la precisión y un valor de 0.9714 para

Tabla 7-6: Reporte de clasificación para una permutación de los conjuntos de datos DAGM C2 y DAGM C5.

DAGM C2				
	Precisión	**exhaustividad**	**Valor-F**	**Soporte**
Defecto	0.8095	0.9714	0.8831	35
No defecto	0.9946	0.9589	0.9765	195
Exactitud			0.9608	230
Promedio macro	0.9021	0.9652	0.9298	230
Promedio ponderado	0.9665	0.9608	0.9623	230
DAGM C5				
	Precisión	**exhaustividad**	**Valor-F**	**Soporte**
Defecto	0.8918	0.9428	0.9166	35
No defecto	0.9896	0.9795	0.9845	195
Exactitud			0.9739	230
Promedio macro	0.9407	0.9612	0.9506	230
Promedio ponderado	0.9747	0.9739	0.9742	230

Tabla 7-7: Conteos de localización para una permutación en los conjuntos de datos DAGM C2 y DAGM C5.

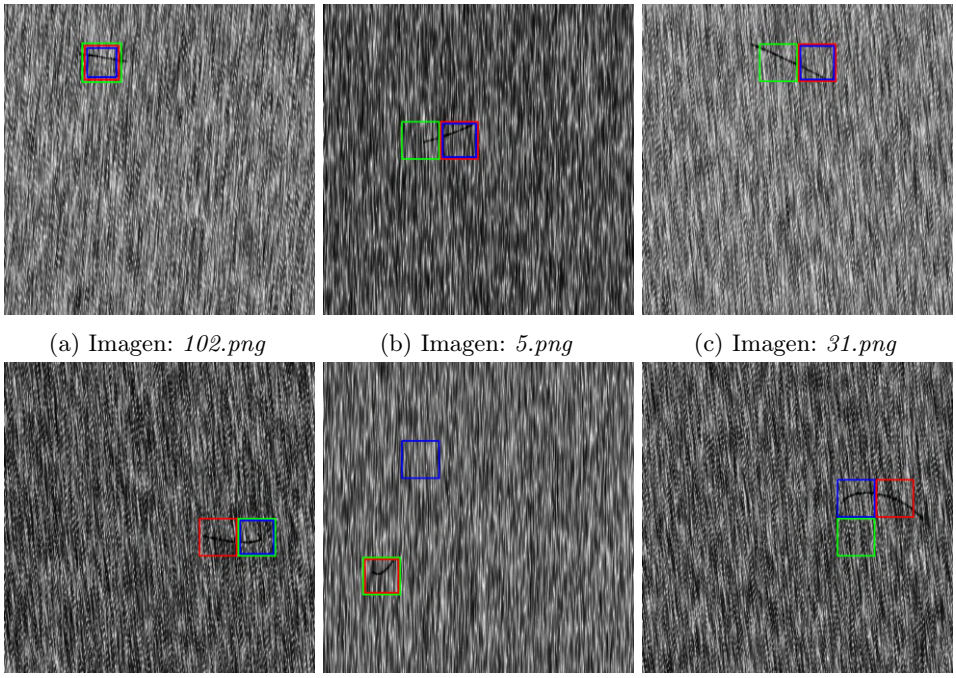
Localización	**DAGM C2**	**DAGM C5**
Completamente localizados	26	32
Adecuadamente localizados	4	0
Deficientemente localizados	3	0
No localizados	1	1
Total	34	33

la exhaustividad en la clase con defecto, lo cual demuestra que el modelo propuesto detecta bien los defectos pero incluye algunas imágenes sin defecto; esto se refleja en un valor-F de 0.8831. De otro lado, para la clase sin defectos en el conjunto de datos DAGM C2, el valor de 0.9946 en precisión y el valor de 0.9589 en exhaustividad reflejan que el modelo detecta bien las imágenes sin defecto e incluye muy pocas con defecto. En contraste, para el conjunto de datos DAGM C5, las métricas de desempeño presentan mejores resultados que los que se observaron para el conjunto de datos DAGM C2, encontrando una precisión con un valor macro promedio de 0.9407 que indica que hay un buen balance entre la clasificación de las imágenes con y sin defecto, lo cual que se corrobora al encontrar valores-F de 0.9166 para las imágenes con defecto y de 0.9845 para las imágenes sin defecto.

Con respecto al conteo para medir los resultados de localización, se utilizó la escala de graduaciones previamente definida en la Sección 7.2.6. Los resultados correspondientes obtenidos para los dos conjuntos de datos DAGM se presentan en la Tabla **7-7**.

Como se puede constatar para el conjunto de datos DAGM C2, de 34 imágenes clasificadas correctamente, 30 están bien localizadas (26 completamente y 4 adecuadamente), lo que corresponde a un 88.23 % de localizaciones correctas. De otro lado, 3 imágenes están clasificadas por sólo una de las 3 predicciones permitiendo aún localizar el defecto aunque de manera deficiente, lo cual corresponde a un 8.82 %; finalmente, sólo en una imagen no pudo ser localizado el defecto con ninguna de las tres predicciones lo cual corresponde al 2.94 % de los casos. Análogamente, para el conjunto de datos DAGM C5, de 33 imágenes clasificadas correctamente, 32 están bien localizadas correspondiendo al 96.97 %, y solo en una imagen no se pudo localizar el defecto con ninguna de las tres predicciones, lo cual equivale al 2.94 %. Nótese que los resultados obtenidos para estos conteos de localización, en ambos conjuntos de datos DAGM, presentan un comportamiento coherente con el observado para los resultados de clasificación reportados arriba.

En la Figura **7-11** se pueden apreciar algunos resultados de la localización para el conjunto de datos DAGM C2, en los cuales se adoptó la convención de visualización explicada en la Sección 7.2.6. Se destaca, en la Figura **7-11**a, que las tres cuadrículas coinciden sobre la misma región de la imagen que contiene el defecto, para un defecto completamente localizado, encontrando que los resultados obtenidos de las tres menores distancias fueron calculados sobre la misma región en la imagen. En las Figuras **7-11**b, **7-11**c y **7-11**d, dos de las cuadrículas coinciden sobre la misma región de la imagen conteniendo el defecto, mientras que la tercera cuadrícula se ubica sobre otra porción de la imagen y que también corresponde al defecto, resultando en defectos completamente localizados. finalmente, en las Figuras **7-11**e y **7-11**f, dos de las cuadrículas se posicionan sobre el defecto, mientras que la tercera se posiciona fuera del defecto, resultando en defectos adecuadamente localizados.



(a) Imagen: *102.png* (b) Imagen: *5.png* (c) Imagen: *31.png*

(d) Imagen: *32.png* (e) Imagen: *48.png* (f) Imagen: *134.png*

Figura 7-11: Resultado gráfico de la localización de defectos en el conjunto de datos DAGM C2 utilizando una cuadrícula de 64×64, donde las Figuras (a)—(d), tienen defectos completamente localizados, mientras que las Figuras (e) y (f) presentan defectos adecuadamente localizados.

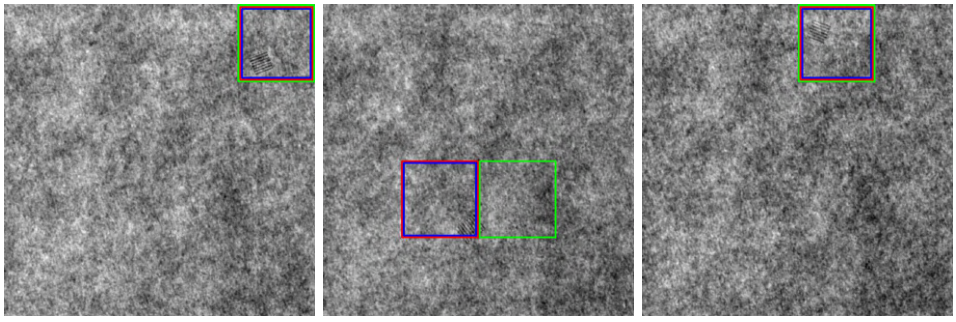
Similarmente, en la Figura **7-12** se ilustra la localización de varias imágenes con defectos pertenecientes al conjunto DAGM C5, todos los cuales están completamente localizados. Se destaca que —en las Figuras **7-12**a, **7-12**c, **7-12**d, **7-12**e y **7-12**f— las tres cuadrículas enmarcan la misma porción del defecto; en cambio, en la Figura **7-12**b, se observa que dos de las cuadrículas enmarcan una misma porción del defecto, mientras que la tercera se posiciona sobre otra parte del defecto.

Otro tipo de resultado que se puede obtener de este método es una localización en multi-resolución; es decir, la visualización del mismo defecto localizado en las tres cuadrículas en que se puede detectar, tal como se aprecia en las Figuras **7-13** y **7-14**. En las Figuras **7-13**a y **7-14**a se presenta la imagen original, mientras que en las demás figuras se presenta la misma imagen localizando el defecto con cuadrículas de 256×256, 128×128 y 64×64, respectivamente. Es interesante ver en ambos casos que, aunque el mayor tamaño de bloque garantiza la detección completa del defecto, se incluye también una porción significativa de área de fondo. Parece entonces preferible un tamaño de bloque intermedio, aunque pueda dejarse por fuera una pequeña parte del defecto o que éste quede repartido en dos bloques distintos.

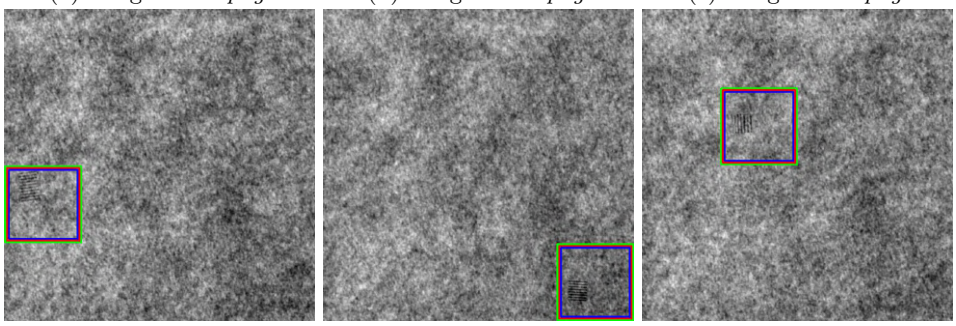
Interpretación

Con el fin de explicar el motivo de la asignación de una etiqueta de clase a una determinada imagen en los conjuntos de datos DAGM C2 y DAGM C5, se utilizan las convenciones definidas previamente en las Secciones 7.2.6 y 7.2.7. En la Figura **7-15** se presenta un ejemplo correspondiente a la interpretación de un defecto perteneciente al conjunto de datos DAGM C2 en el que se puede apreciar, en la Figura **7-15**a, que el defecto está completamente localizado; además, se puede ver que los defectos localizados en las tres imágenes del conjunto de entrenamiento —presentados en las Figuras **7-15**b, **7-15**c y **7-15**d— corresponden o son similares a las secciones del mismo color señaladas en la imagen de prueba, donde todas las cuadrículas enmarcan defectos similares.

De forma análoga, en la Figura **7-16** se pueden ver las interpretaciones de los resultados de detección y localización

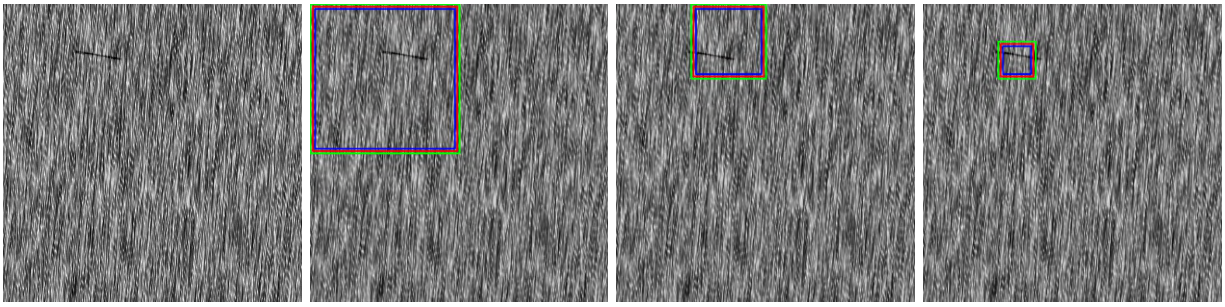


(a) Imagen: *100.png* (b) Imagen: *102.png* (c) Imagen: *132.png*



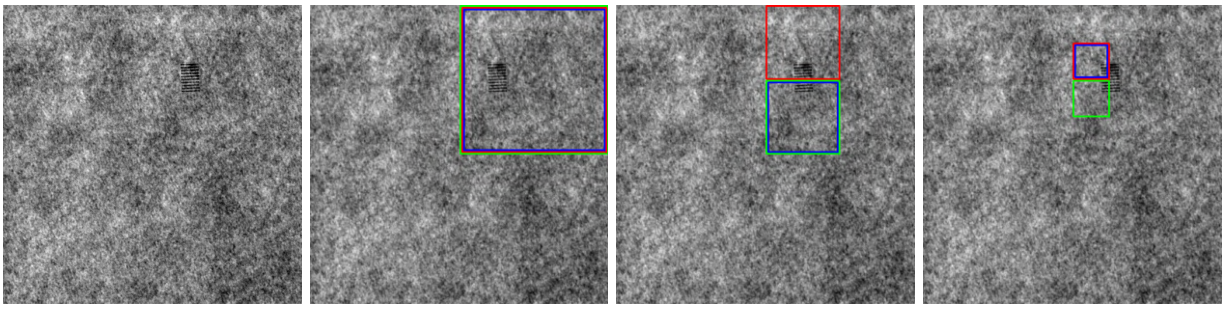
(d) Imagen: *134.png* (e) Imagen: *143.png* (f) Imagen: *91.png*

Figura 7-12: Resultado gráfico de la localización de defectos en el conjunto de datos DAGM C5, utilizando una cuadrícula de 128×128, donde todas las figuras contienen imágenes que presentan defectos completamente localizados.



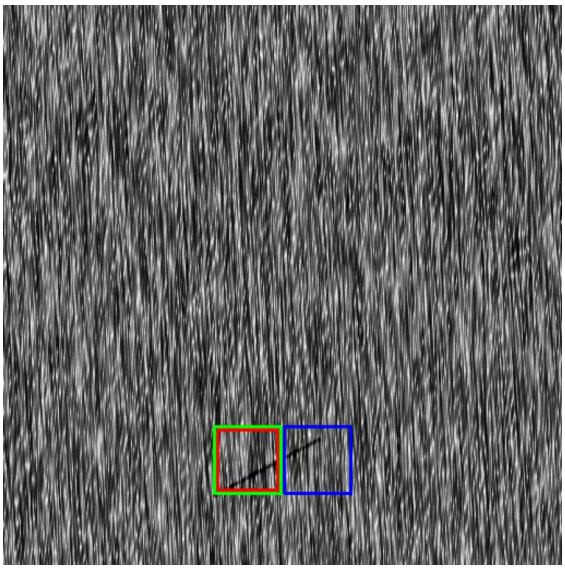
(a) Imagen original (b) 256×256 (c) 128×128 (d) 64×64

Figura 7-13: Resultado gráfico de la localización de un defecto en la imagen *102.png* del conjunto de datos DAGM C2, presentando la imagen original en Figura (a), y su localización en las tres diferentes cuadrículas en las demás figuras.

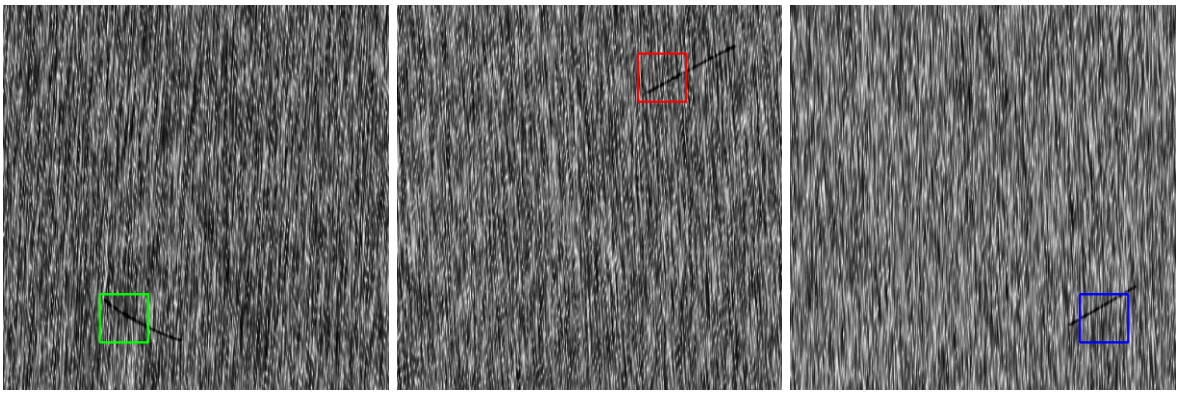


(a) Imagen original (b) 256×256 (c) 128×128 (d) 64×64

Figura 7-14: Resultado gráfico de la localización de un defecto en la imagen *124.png* del conjunto de datos DAGM C5, presentando la imagen original en la Figura (a), y su localización en las tres diferentes cuadrículas en las demás figuras.



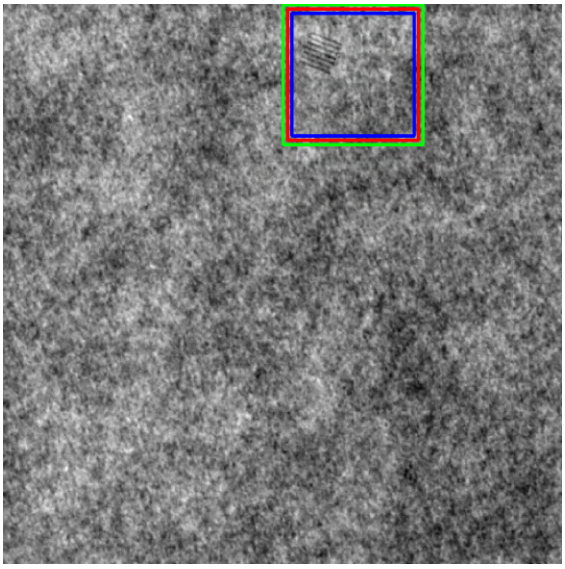
(a) Imagen de prueba



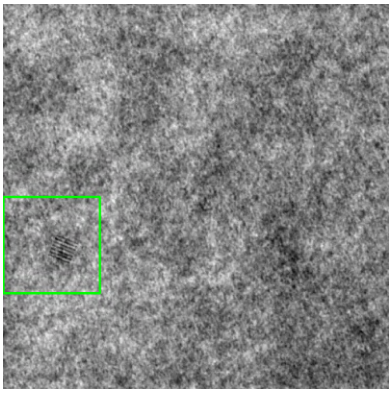
(b) Primer bloque más cercano (c) Segundo bloque más cercano (d) Tercer bloque más cercano

Figura 7-15: Interpretación gráfica de la localización de un defecto en la imagen *19.png* perteneciente al conjunto de datos DAGM C2, donde la Figura (a) es la imagen de prueba en la cual se localiza completamente el defecto que tiene forma de línea; las Figuras (b)—(d) contienen regiones que demarcan segmentos de líneas, similares a las regiones del mismo color en la imagen de prueba.

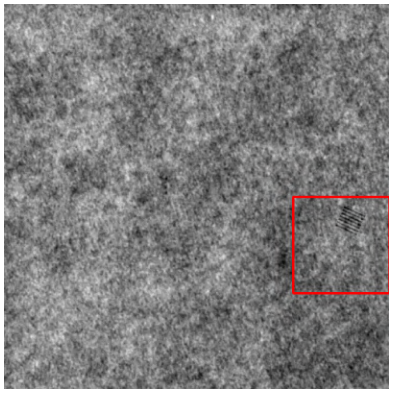
de un defecto en una imagen de prueba con forma de patrón de líneas correspondiente al conjunto de datos DAGM C5. En este ejemplo puede verse, en la Figura **7-16**a, que el defecto está completamente localizado; además, en las Figuras **7-16**b, **7-16**c y **7-16**d, se aprecian imágenes pertenecientes al conjunto de entrenamiento, con defectos que están claramente enmarcados y que presentan características similares a aquéllas del defecto de la imagen de prueba en la Figura **7-16**a.



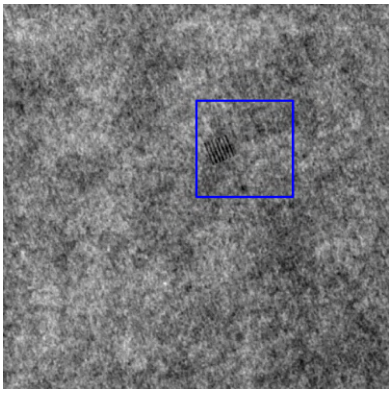
(a) Imagen de prueba



(b) Primer bloque más cercano



(c) Segundo bloque más cercano



(d) Tercer bloque más cercano

Figura 7-16: Interpretación gráfica de la localización de un defecto en la imagen *132.png* perteneciente al conjunto de datos DAGM C5, donde la Figura (a) es la imagen de prueba en la cual se localiza completamente el defecto en forma de patrón de líneas; las Figuras (b)—(d) contienen regiones similares a las regiones del mismo color en la imagen de prueba.

7.4.2 Resultados con el conjunto de datos AITEX

Detección de defectos

Para el conjunto de datos AITEX, cuyos resultados de exactitud en la clasificación se presentan en la Tabla **7-8**, se observa que el uso de la metodología propuesta genera resultados ligeramente mejores que los obtenidos al aplicar la metodología de referencia. Nótese que, con respecto a esta medida de rendimiento, el mejor resultado de 0.8608

se obtiene utilizando la variante ResNet50 en imágenes divididas en una cuadrícula de 256×256. En cuanto a la metodología de referencia, la exactitud de 0.8157 utilizando una cuadrícula de 64×64 es mejor que cuando se utiliza la metodología propuesta (aunque la diferencia con el uso de InceptionResNetV2 no es significativa) y, por otro lado, para una cuadrícula de 128×128, el resultado de 0.8361 para la exactitud es mejor que el alcanzado utilizando ResNet50 como extractor de características —con la cual se logra una exactitud de 0.8159— y ligeramente inferior que cuando se utiliza InceptionResNetV2, con la que se obtiene un desempeño de 0.8523. Estos resultados demuestran que la metodología de base con extracción de características de bajo nivel es bastante efectiva cuando los defectos son más visibles y los fondos son uniformes.

Tabla 7-8: Resultados de exactitud en el conjunto de datos AITEX según la cuadrícula.

Extr. caract.	256×256	128×128	64×64
Propuesta(Inception)	0.8567	0.8523	0.8156
Propuesta(ResNet50)	0.8608	0.8159	0.7702
Línea base	0.8524	0.8361	0.8157

Dado que el conjunto de datos AITEX también presenta un ligero desbalance entre las dos clases (57 % sin defecto y 43 % con defecto) y considerando nuevamente lo sugerido en [Bramer. 2020], se presentan y discuten a continuación las medidas de desempeño TP-rate y FP-rate para este conjunto de datos; ver Tabla **7-9**.

Tabla 7-9: TP-rate and FP-rate para el conjunto de datos AITEX, según el método de extracción de características y la cuadrícula.

	TP-rate, FP-rate según cuadrícula		
Extr. caract.	**Cuadrícula**	**TP-rate**	**FP-rate**
Propuesta(Inception)	256×256	0.7989	0.1001
Propuesta(ResNet50)	256×256	0.7919	0.0881
Línea base	256×256	0.7175	**0.0469**
Propuesta(Inception)	128×128	0.8631	0.1522
Propuesta(ResNet50)	128×128	**0.8835**	0.2283
Línea base	128×128	0.7329	0.0817
Propuesta(Inception)	64×64	0.8427	0.2021
Propuesta(ResNet50)	64×64	0.7280	0.1966
Línea base	64×64	**0.7025**	0.0946

Los resultados obtenidos con la metodología propuesta siguen siendo comparativamente los más altos, destacándose el hecho que el mayor TP-rate (0.8835) se logra usando ResNet50 con una cuadrícula de 128×128; por el contrario, el TP-rate más bajo (0.7025) se obtiene utilizando la metodología de base en una cuadrícula de 64×64. En cuanto al FP-rate, los mejores resultados se obtienen utilizando la metodología base sobre una cuadrícula de 256×256, alcanzando un valor de 0.0469. Estos valores de TP-rate y FP-rate demuestran la efectividad de los diferentes sistemas para detectar imágenes ya sea con defecto o sin defecto.

Con el fin de obtener una valoración conjunta de los resultados de TP-rate y FP-rate, éstos se presentan como puntos en la gráfica ROC [Bramer. 2020] para las diferentes combinaciones de cuadrícula y metodologías; ver Figura **7-17**. Se observa que el mejor clasificador para el conjunto de datos AITEX es alcanzado con la metodología propuesta utilizando InceptionResNetV2 para la extracción de características con una retícula de 128×128; además, se nota que los clasificadores que utilizan la retícula más pequeña (64×64) ofrecen resultados inferiores a los que utilizan retículas más grandes, lo que se puede explicar dado que algunos de los defectos presentan dimensiones mucho más grandes que la misma retícula. Es importante destacar la paridad encontrada entre los clasificadores de la metodología propuesta y retículas bien sean de 256×256 o de 128×128; la similitud de sus valores de Exactitud, TP-rate y FP-rate indican que se requiere un tamaño de descomposición en bloques suficientemente grande pero, una vez garantizado, las diferencias de desempeño observadas no resultan significativas.

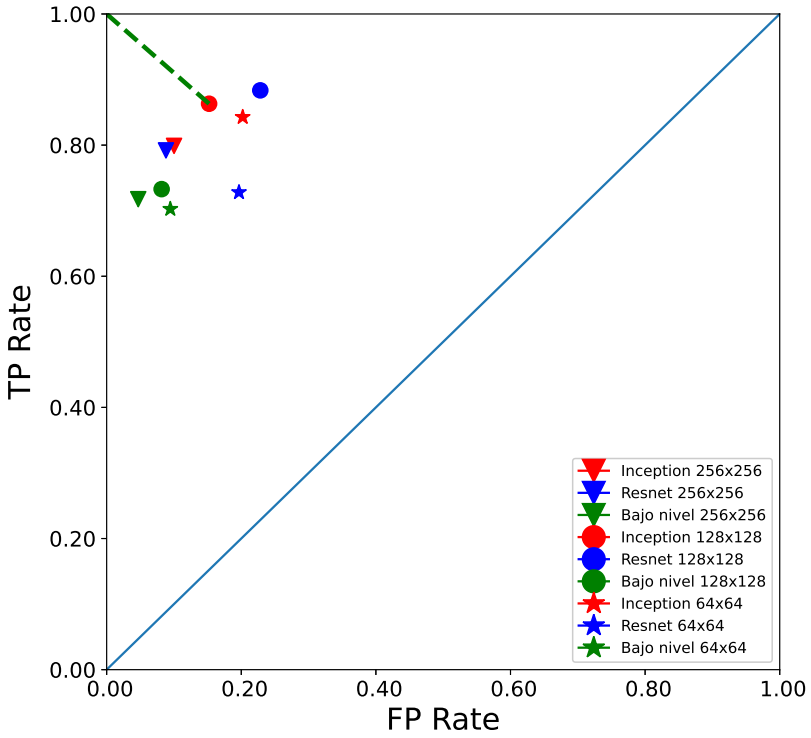


Figura 7-17: Comparación de clasificadores mostrando el mejor clasificador con la distancia desde el clasificador perfecto para el conjunto de datos AITEX.

Localización

Para ilustrar la localización de defectos en el conjunto de datos AITEX, se seleccionaron los resultados obtenidos con la metodología propuesta para una de las permutaciones de prueba, particularmente para el caso de usar la variante de InceptionResNetV2 como extractor de características sobre una cuadrícula de 128×128. La matriz de confusión resultante y un reporte de clasificación de otras medidas de desempeño se muestran en las Tablas **7-10** y **7-11**, respectivamente.

Tabla 7-10: Matriz de confusión para una permutación del conjunto de datos AITEX con extracción de características usando CNN+TL (InceptionResNetV2) sobre una cuadrícula de 128×128.

		Predicha	
	AITEX	**Defecto**	**No defecto**
Actual	Defecto	20	3
	No defecto	1	25

Tabla 7-11: Reporte de clasificación para una permutación del conjunto de datos AITEX.

	Precisión	**Exhaustividad**	**Valor-F**	**Soporte**
Defecto	0.9524	0.8696	0.9091	23
No defecto	0.8928	0.9615	0.9259	26
Exactitud			0.9183	49
Promedio macro	0.9226	0.9155	0.9175	49
Promedio ponderado	0.9208	0.9184	0.9180	49

De la Tabla **7-11** se observa que la métrica con menor valor es la exhaustividad para la clase con defecto, presentando un valor de 0.8696, lo que se refleja en resultados de valor-F diferenciados, a saber: 0.9091 para la clase con defecto y 0.9259 para la clase sin defecto. La consideración conjunta de los resultados de precisión y exhaustividad, para ambas clases, conduce a las siguientes observaciones: para la clase con defecto, una precisión de 0.9524 y una exhaustividad de 0.8696 indican que la metodología propuesta no detecta muy bien la clase positiva pero que, es altamente confiable; por otro lado, para la clase sin defecto, una precisión de 0.8928 y una exhaustividad de 0.9615 sugieren que, aunque la metodología propuesta detecta bien la clase negativa, también puede incluir erróneamente algunas muestras de la clase con defecto.

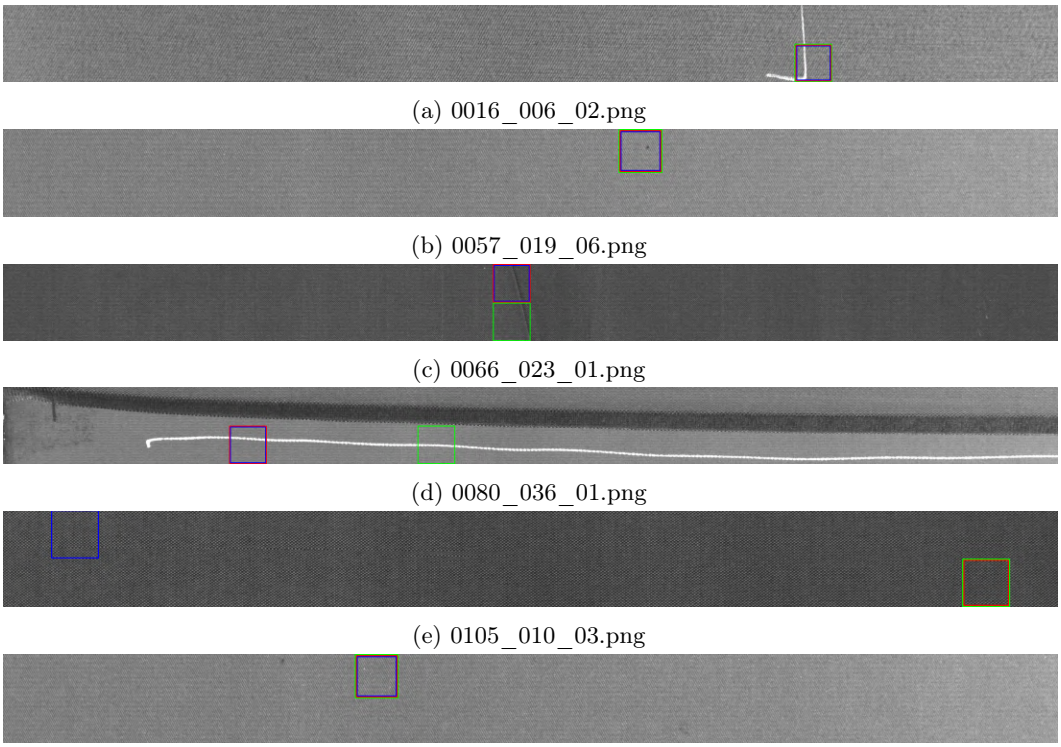
Los resultados de localización, utilizando la escala definida en la Sección 7.2.6 y para una permutación de prueba del conjunto de datos, se presentan en la Tabla **7-12**. Como se puede notar, 15 de 20 imágenes clasificadas correctamente están bien localizadas (completamente y adecuadamente), lo que corresponde al 75.0 % de los casos. Por otra parte, 2 imágenes —es decir, el 10.0 % del total— están clasificadas por sólo una de las 3 predicciones (deficientemente) pero permitiendo aún localizar el defecto; finalmente, en tres imágenes no pudo ser localizado el defecto con ninguna de las tres predicciones, lo que corresponde al 15.0 % de los casos. Los anteriores resultados en parte se pueden atribuir a las complejidades intrínsecas que presenta el conjunto de datos, antes mencionadas en la Sección 7.3.1.

En la Figura **7-18** se pueden apreciar algunos resultados de la localización para el conjunto de datos AITEX. En primer lugar, en la Figura **7-18**a, se observa un ejemplo que contiene un defecto alargado con forma de línea vertical

Tabla 7-12: Conteos de localización para una permutación en el conjunto de datos AITEX.

Localización	**AITEX**
Completamente localizados	14
Adecuadamente localizados	1
Deficientemente localizados	2
No localizados	3
Total	20

de color blanco, en el cual la localización se realiza completamente sobre una parte del defecto; a continuación, en la Figura **7-18**b, se presenta un defecto con forma de punto, el cual también se localiza completamente. En la Figura **7-18**c se presenta un defecto en forma de pliegue vertical y con el mismo color de la tela, el cual se localiza completamente pero sobre dos regiones diferentes del defecto, similar a la Figura **7-18**d, que presenta un defecto longitudinal de color blanco que cubre casi toda la imagen, el cual también se localiza completamente pero sobre dos regiones diferentes del defecto. Finalmente, las Figuras **7-18**e y **7-18**f presentan un defecto deficientemente localizado y otro mal localizado respectivamente.



(a) 0016_006_02.png

(b) 0057_019_06.png

(c) 0066_023_01.png

(d) 0080_036_01.png

(e) 0105_010_03.png

(f) 0055_019_06.png

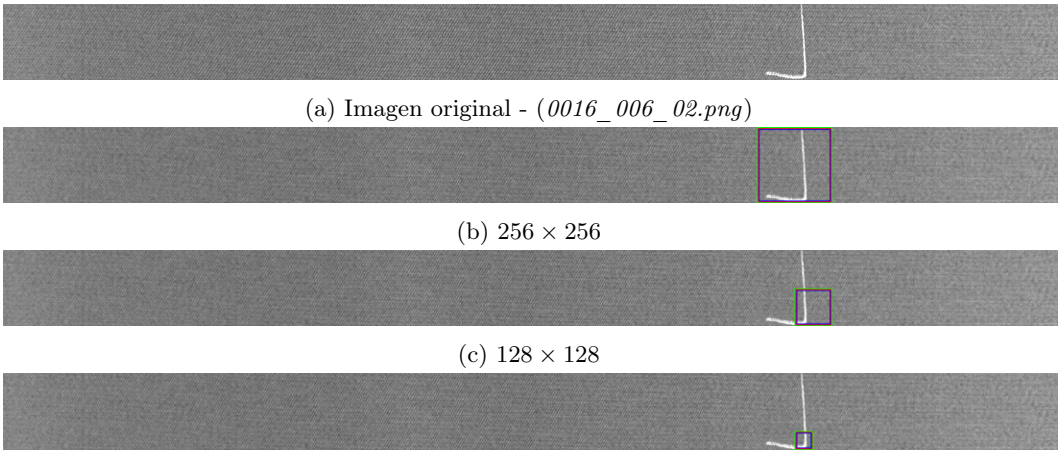
Figura 7-18: Resultado gráfico de la localización de defectos en imágenes del conjunto de datos AITEX, utilizando una retícula de 128×128, donde las imágenes en las Figuras (a)—(d) presentan defectos completamente localizados, la imagen en la Figura (e) presenta un defecto deficientemente localizado, y por último, la imagen en la Figura (f) presenta un defecto mal localizado.

Para este conjunto de datos también se muestra el resultado multi-resolución que se puede obtener del método de localización, el cual permite visualizar el mismo defecto presentando las tres cuadrículas consideradas para la descomposición en bloques, tal como se aprecia en la Figura **7-19**. En la Figura **7-19**a se presenta la imagen original, mientras que en las demás figuras se muestra la misma imagen localizando el defecto con cuadrículas de tamaños 256×256, 128×128 y 64×64 respectivamente. Nótese que, en este caso particular, el bloque de tamaño 256×256 es el que mejor localiza el defecto completo, mientras que los tamaños de bloque más pequeños se enfocan en un área clave donde el defecto lineal cambia de orientación.

Interpretación

Para explicar la asignación de la etiqueta de clase a una determinada imagen del conjunto de datos AITEX con la metodología propuesta, se utilizan las convenciones definidas previamente en las Secciones 7.2.6 y 7.2.7. En particular, en la Figura **7-20**, se presenta un ejemplo correspondiente a la interpretación de un defecto en forma de línea blanca encontrado en la imagen *0013_006_02.png* perteneciente al conjunto de prueba.

En la Figura **7-20**a se logra apreciar que el defecto está completamente localizado, donde las cuadrículas verde y



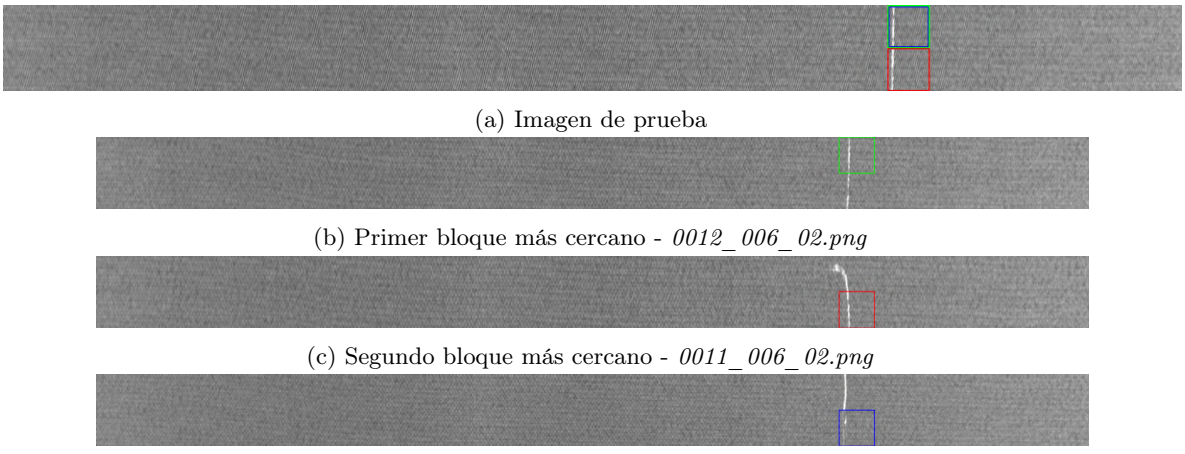
(a) Imagen original - (*0016_ 006_ 02.png*)

(b) 256×256

(c) 128×128

(d) 64×64

Figura 7-19: Resultado gráfico de la localización de un defecto en la imagen *0016_ 006_ 02.png* que pertenece al conjunto de datos AITEX, presentando la imagen original en la Figura (a), y su localización en las tres diferentes cuadrículas en las demás figuras.



(a) Imagen de prueba

(b) Primer bloque más cercano - *0012_ 006_ 02.png*

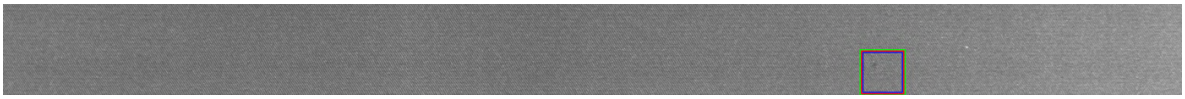
(c) Segundo bloque más cercano - *0011_ 006_ 02.png*

(d) Tercer bloque más cercano - *0015_ 006_ 02.png*

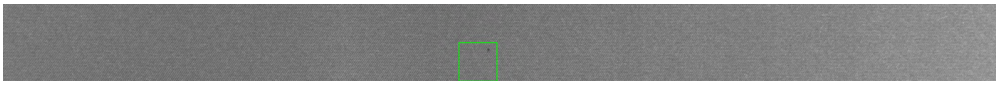
Figura 7-20: Interpretación gráfica de la localización de un defecto en la imagen *0013_ 006_ 02.png* perteneciente al conjunto de datos AITEX , donde la Figura (a) es la imagen de prueba; las Figuras (b)—(d) contienen regiones similares a las regiones del mismo color en la imagen de prueba.

azul se ubican sobre una misma porción del defecto mientras que la cuadrícula roja se posiciona sobre otra porción diferente del mismo defecto lo que permite una mejor localización. De igual manera, en la Figura **7-20**b y con una región de color verde, se presenta una imagen del conjunto de entrenamiento donde se encontró una región que enmarca una línea de color blanco, la cual es similar a la región señalada con color verde en la imagen de prueba. De otra parte, en la Figura **7-20**c y con una región demarcada en color rojo, se presenta otra imagen perteneciente al conjunto de entrenamiento donde se encontró una región que enmarca una sección de línea en color blanco, la cual es similar a la región del mismo color en la imagen de prueba. Finalmente, en la Figura **7-20**d con una región de color azul, se enmarca una porción de defecto en una imagen perteneciente al conjunto de entrenamiento que es similar a la región del mismo color en la imagen de prueba. Adicionalmente, se puede verificar que los defectos localizados en las Figuras **7-20**b, **7-20**c y **7-20**d están bien localizados y que corresponden o son similares a las regiones del mismo color definidas en la imagen de prueba en la Figura **7-20**a.

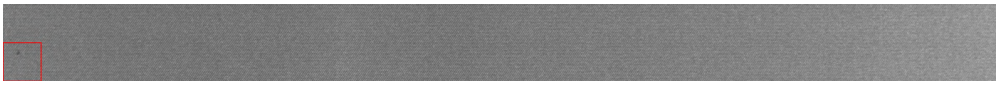
Otro ejemplo de detección e interpretación de defectos, correspondiente a la imagen *0082_030_04.png* del conjunto AITEX, se presenta en la Figura **7-21**. En la Figura **7-21**a se observa que el defecto con forma de pequeña mancha está completamente localizado y que, en este caso, las tres cuadrículas se localizan sobre la misma porción del defecto. Además, al analizar las Figuras **7-21**b, **7-21**c y **7-21**d, se nota que éstas también presentan defectos con forma de pequeñas manchas, los cuales están bien localizados y son similares al defecto de la imagen de prueba mostrada en la Figura **7-21**a.



(a) Imagen de prueba - (*0082_030_04.png*)



(b) Primer bloque más cercano - (*0046_019_04.png*



(c) Segundo bloque más cercano - (*0045_019_04.png*)

(d) Tercer bloque más cercano - (*0043_019_04.png*)

Figura 7-21: Interpretación gráfica de la localización de un defecto en la imagen *0082_030_04.png* perteneciente al conjunto de datos AITEX, donde la Figura (a) es la imagen de prueba; las Figuras (b)—(d) contienen regiones similares a las regiones del mismo color en la imagen de prueba.

7.5 Análisis computacional

Teniendo en cuenta que esta metodología se enfoca en desarrollar soluciones susceptibles de ser empleadas en las líneas de producción, específicamente en los SIVA, en esta sección se hace un análisis computacional incluyendo los tiempos que toman cada una de las diferentes rutinas y algoritmos desarrollados para así determinar su potencial de aplicación. Con el fin de medir los tiempos de respuesta, se procedió a realizar los cálculos sobre el conjunto DAGM C2, el cual contiene 1150 imágenes, de las cuales se utilizaron 930 para el entrenamiento del modelo y las 230 restantes para las pruebas.

En cuanto al desarrollo de la metodología, por su índole investigativo y exploratorio, inicialmente se formuló como cuatro aplicaciones independientes y en concordancia con las etapas que la conforman, las cuales se comunican entre sí mediante el paso de archivos, siendo necesario en cada una de ellas leer datos, cargar modelos, realizar los respectivos procesos y almacenar resultados. El principal objetivo de estas aplicaciones es validar el funcionamiento de la metodología y aplicar las diferentes métricas y estadísticas.

La primera aplicación se encarga de realizar la extracción de características, para lo cual debe leer las imágenes, preprocesarlas, distribuirlas, entrenar la red neuronal basada en CNN+TL, y luego, cada una de las imágenes (las de entrenamiento y las de prueba) debe ser procesada por la red para realizar así la extracción de las características. Posteriormente debe generar y almacenar en archivos los resultados y el modelo entrenado. Como se puede apreciar en la Tabla **7-13**, para la ejecución de esta aplicación se requiere entre 10 minutos para la cuadrícula de 256×256 y 2.7 horas para la cuadrícula de 64×64.

La segunda aplicación utiliza PCA para reducir la cantidad de características extraídas a un valor fijo de 50, según lo propuesto por Mera [2017], siendo necesario leer desde archivos la totalidad de las características almacenadas en la etapa previa, entrenar el algoritmo de PCA con las características de entrenamiento y aplicarlo tanto a las características de entrenamiento como a las de prueba, almacenando en archivos los respectivos resultados y modelos. Aplicar PCA a todo el conjunto de datos toma entre 2.75 y 24 segundos dependiendo de la cuadrícula seleccionada (ver Tabla **7-13**).

La tercera aplicación, se encarga de predecir los resultados para todas las imágenes del conjunto de prueba, siendo necesario leer los archivos con las 50 características resultantes de aplicar PCA, tanto de entrenamiento como de prueba, posteriormente, según la representación MIL, construye bolsas para cada una de las imágenes y a continuación se realiza el proceso de clasificación utilizando el algoritmo k-NN (para $k = 3$), calculando la distancia desde cada una de las bolsas de prueba con respecto a las bolsas de entrenamiento y mediante un proceso de votación se determina la etiqueta de cada bolsa de prueba, así como los elementos responsables de dicha respuesta que se utilizan para la localización e interpretación de los defectos. Finalmente, se calculan las diferentes métricas, se presenta la información y posteriormente se almacena en archivos, tomando entre 33.73 segundos y 1.4 horas (dependiendo de la cuadrícula) realizar el procesamiento y predicción para todas las imágenes del conjunto de prueba como se puede ver en la Tabla **7-13**.

La cuarta y última aplicación, cuyo objetivo es validar y visualizar los resultados, se desarrolló utilizando *Google Colaboratory* y se encarga de leer los archivos resultantes de la predicción y de forma gráfica presentar los resultados de localización e interpretación.

Tabla 7-13: Tiempos transcurridos, en segundos, para las tres primeras aplicaciones de la metodología aplicada sobre una permutación de la variante C2 del conjunto DAGM, con su total y el respectivo promedio para la predicción de una imagen.

	Aplicación			Tiempos	
Tam. cuadrícula	Primera	Segunda	Tercera	Total	Promedio
256×256	641.80	2.75	33.73	678.28	0.13
128×128	2458.72	6.67	351.93	2817.32	1.53
64×64	9723.13	24.24	5044.06	14791.43	21.93

Es importante aclarar que en la forma que se desarrollaron las aplicaciones no sería posible verificar la funcionalidad de la metodología según los requerimientos de un ambiente de producción, no siendo factible determinar para una sola imagen el tiempo real que toma realizar el proceso de predicción, ya que se calcula sobre promedios de ejecutar todas las aplicaciones, incluyendo el entrenamiento del modelo, el procesamiento de todas las imágenes de prueba, así como los tiempos requeridos para leer y almacenar archivos. Por tal motivo, con el fin de determinar la viabilidad y aplicabilidad de la metodología, las anteriores aplicaciones se agruparon y distribuyeron en dos nuevas aplicaciones diferentes e independientes de acuerdo con su funcionalidad y utilidad. La primera se encarga de entrenar el modelo y procesar las imágenes de entrenamiento, mientras que la segunda se encarga de hacer la predicción para una imagen de prueba. Estas dos aplicaciones se explican y analizan con más detalle a continuación.

7.5.1 Entrenamiento del modelo

La primera aplicación se encarga de definir y entrenar el modelo basado en CNN+TL, extraer las características para cada una de las imágenes del conjunto de entrenamiento, aplicar PCA para reducir la dimensión resultante y almacenar en archivos el modelo entrenado, así como el resultado de la extracción de características según la cuadrícula seleccionada. Los pasos correspondientes con esta primera aplicación se pueden ver en el Algoritmo 4.

Algoritmo 4 Entrenamiento de la red neuronal y extracción de las características del conjunto de entrenamiento. Recibe como parámetros: *modelo* es el tipo de modelo utilizado para la predicción, *cuadricula* es el tamaño de la cuadrícula utilizada para dividir las imágenes y *permuta* es el número de la permutación que se va a utilizar para distribuir las imágenes.

```
function ENTRENA_Y_EXTRAE(modelo, cuadricula, permuta)
    modeloCNN ← crearModelo(modelo)
    imagenes, etiquetas, nombres ← leerImagenes()
    imagenes ← preprocesaImagenes(imagenes)
    x_train, y_train, z_train, x_test, y_test, z_test ← dividirConjunto(imagenes, etiquetas, nombres, permuta)
    entrenaModelo(modeloCNN, x_train, y_train)
    datos ← []
    for idx ← 1 to len(x_train) do
        img ← x_train[idx]
        lab ← y_train[idx]
        nam ← z_train[idx]
        listaImagenes ← dividirImagen(img, cuadricula)
        for idy ← 1 to len(listaImagenes) do
            miniImagen ← preprocesaImagen(listaImagenes[idy])
            caracteristicas ← conseguirCaracteristicas(modeloCNN, miniImagen)
            datos ← datos + almacenaCaracteristicas(idx, idy, lab, nam, caracteristicas)
        end for
    end for
    PCA(datos)
    bolsas ← creaBolsas(datos)
    almacenaBolsas(bolsas)
    almacenaModeloCNN(modeloCNN)
    almacenaModeloPCA(PCA)
end function
```

Complejidad computacional

El correspondiente análisis computacional que utiliza la notación O grande (big O) sobre el algoritmo que entrena la red neuronal y extrae las características es el siguiente:

Complejidad: $O(s) + O(n \cdot m \cdot k)$, donde s es el costo de entrenar el modelo que depende de varios parámetros como el número de épocas, la cantidad total de parámetros entrenables, el número de operaciones (convoluciones, agrupación, activación, etc.) y el tamaño del conjunto de entrenamiento; n es el número de imágenes utilizadas para el entrenamiento, m es el número de imágenes resultantes de dividir una imagen según la cuadrícula seleccionada y k es la complejidad de extraer las características para una imagen. La complejidad resultante se interpreta como que el costo computacional depende principalmente de entrenar el modelo más extraer las características para cada una de las sub-imágenes que conforman el conjunto de entrenamiento. Se asume que las complejidades de otras tareas de apoyo como las de creación y compilación del modelo, carga de imágenes, pre-procesamiento de imágenes, aplicación de PCA, creación de bolsas y almacenamiento de modelos y resultados se realizan al mismo nivel y presentan complejidades inferiores a las ya reportadas razón por la cual no se tienen en cuenta en el cálculo global de la complejidad.

Para el caso específico de una permutación del conjunto DAGM C2, donde el conjunto consta de 1150 imágenes de las cuales 930 se destinan para el entrenamiento del modelo y las 230 restantes para las pruebas, y si se utiliza una cuadrícula de 128×128, los valores resultantes son los siguientes:

- $n = 930$
- $m = 16$
- s = Costo de entrenar el modelo que depende de la cantidad de imágenes (n=930), del número de épocas para entrenar (épocas=5), del número de capas de la red (InceptionResNetV2=449 [3] + 4 del nuevo clasificador) y el tipo de las capas de la red.
- k = Costo de extraer las características de una imagen que depende del tamaño de cada sub-imagen que es de 64×64 y del tamaño de la red (InceptionResNetV2=449+4 del nuevo clasificador).
- $O(s) + O(930 \cdot 16 \cdot k)$
- $O(s) + 14880 \cdot O(k)$
- $O(s) + O(k)$

Dado lo anterior, se puede concluir, que la complejidad del algoritmo para entrenar el modelo y extraer las características para las imágenes de entrenamiento, independientemente de la cantidad de imágenes utilizadas para entrenar el modelo, el número de épocas con que se entrena y, la cantidad de imágenes a las que se le extraen características, depende principalmente de la red neuronal seleccionada, incluyendo factores y variables como el número de capas así como la composición y el tipo de las capas de la red.

Análisis de tiempos

Con el fin de determinar y comprender la funcionalidad y comportamiento de esta aplicación, se calcularon los tiempos que requiere el entrenamiento del modelo (InceptionResNetV2) y la extracción de características bajo cinco permutaciones de los datos de entrada, según las tres diferentes cuadrículas donde se calcularon los respectivos tiempos, promedios y SD, tal como se pueden ver en la Tabla **7-14**.

En cuanto al entrenamiento del modelo, es importante tener en cuenta que dicho proceso solo se realiza una vez y, con el servidor de cómputo utilizado, previamente referenciado en la Sección 7.3.2, los tiempos de entrenamiento del modelo varían entre nueve minutos y medio para la cuadrícula de 256×256 y dos horas y media para la cuadrícula de 64×64, tiempos que podrían tener poca relevancia, ya que —como se mencionó anteriormente— el entrenamiento se hace fuera de línea y una única vez.

[3] https://keras.io/api/applications/

Tabla 7-14: Tiempos promedio transcurridos en segundos, con su correspondiente SD, para entrenar el modelo (InceptionResNetV2) sobre cinco permutaciones de la variante C2 del conjunto DAGM diferenciados para las tres posibles cuadrículas.

Cuadrícula	Promedio	SD
256×256	557.9169	2.9792
128×128	2137.9008	3.1057
64×64	8467.5996	43.1642

7.5.2 Predicción de una imagen

Una segunda aplicación, la que potencialmente se utilizaría en un SIVA para realizar la predicción de una imagen, carga el modelo previamente entrenado y las bolsas correspondientes a las imágenes de entrenamiento. Luego, a partir de una nueva imagen (la que se quiere verificar) ejecuta las diferentes fases de la metodología hasta obtener una predicción acompañada de la respectiva información de localización e interpretación. Los pasos correspondientes que permiten realizar la predicción de una imagen se pueden ver en el Algoritmo 5.

Algoritmo 5 Predicción de una imagen. Recibe como parámetros: *modelo* es el tipo de modelo utilizado para la predicción, *nomImagen* es el nombre de la imagen que se va a analizar, *cuadricula* es el tamaño de la cuadrícula utilizada para dividir las imágenes, *permuta* es el número de la permutación que se va a utilizar para distribuir las imágenes y *knn* es el número de vecinos a considerar en el algoritmo k-NN.

function PREDICE($modelo, nomImagen, cuadricula, permuta, knn$)
  $bolsasEntre \leftarrow$ cargaBolsasEntre()
  $modeloCNN \leftarrow$ cargaModeloCNN($modelo, cuadricula$)
  $PCA \leftarrow$ cargaModeloPCA($modelo, cuadricula$)
  $datos \leftarrow []$
  $imagen \leftarrow$ leerImagen($nomImagen$)
  $listaImagenes \leftarrow$ dividirImagen($imagen, cuadricula$)
  for $idy \leftarrow 1$ to $len(listaImagenes)$ **do**
    $miniImagen \leftarrow$ preprocesaImagen($listaImagenes[idy]$)
    $caracteristicas \leftarrow$ conseguirCaracteristicas($modeloCNN, miniImagen$)
    $datos \leftarrow datos +$ almacenaCaracteristicas($1, idy, , nomImagen, caracteristicas$)
  end for
  $datos \leftarrow$ PCA($datos$)
  $bolsaTest \leftarrow$ creaBolsas($datos$)
  $distancias \leftarrow []$
  for $idz \leftarrow 1$ to $len(bolsasEntre)$ **do**
    $dist, posEntre, posTest \leftarrow$ distanciaEntreBolsas($bolsaTest, bolsasEntre[idz]$)
    $distancias \leftarrow distancias + (dist, posEntre, posTest)$
  end for
  $datosLab \leftarrow mayoritario(distancias, knn)$
  mostrarResp($datosLab$)
end function

Complejidad computacional

El correspondiente análisis computacional del algoritmo que realiza la predicción de una nueva imagen, bajo la notación O grande es el siguiente:

Complejidad: $O(m \cdot k) + O(n \cdot m^2) + O(n \cdot \log_2 n)$, la primera parte que corresponde a la división de la imagen en sub-imágenes y la posterior extracción de características para cada sub-imagen, tiene una complejidad de $O(m \cdot k)$,

donde m es el número de instancias o imágenes resultantes de dividir una imagen según la cuadrícula y k es la complejidad de obtener las características para una imagen. En la segunda parte encargada de calcular la distancia de la imagen a verificar con respecto a todas las imágenes del conjunto de entrenamiento representadas mediante bolsas, la complejidad es de $O(n \cdot m^2)$, donde n es el número de bolsas o imágenes de entrenamiento y m es el número de instancias o imágenes resultantes de dividir una imagen según la cuadrícula las cuales se deben verificar todas contra todas al comparar dos bolsas. Finalmente, en la tercera y última parte del algoritmo donde se realiza la predicción, la complejidad será del orden $O(n \cdot \log_2 n)$, resultado de ordenar y seleccionar las 3 bolsas pertenecientes al conjunto de entrenamiento con la menor distancia a la imagen que se está analizando.

Para el caso específico de una permutación del conjunto DAGM C2, donde el conjunto consta de 1150 imágenes de las cuales 930 se destinan para el entrenamiento del modelo y las 230 restantes para las pruebas, y si se utiliza una cuadrícula de 128×128, los valores resultantes son los siguientes:

- $n = 930$
- $m = 16$
- $k =$ Costo de extraer las características de una imagen (depende del tamaño de cada sub-imagen que es de 64×64) y dependiendo del tipo de red (InceptionResNetV2).
- $O(m \cdot k) + O(n \cdot m^2) + O(n \cdot \log_2 n)$
- $O(16 \cdot k) + O(930 \cdot 16^2) + O(930 \cdot \log_2 930)$
- $O(16 \cdot k) + O(238080) + O(9170)$
- $O(k)$

Dado lo anterior, se puede concluir que la complejidad del algoritmo para predecir la etiqueta de una imagen, depende principalmente del costo de la extracción de las características para las imágenes y en gran medida de la constante de calcular la distancia entre la bolsa de prueba y todas las bolsas de entrenamiento (que siempre son de la misma dimensión).

Análisis de tiempos

En cuanto al análisis de los tiempos requeridos para la predicción de una imagen, se realizaron 50 repeticiones de predicciones sobre diferentes imágenes y se calcularon los respectivos tiempos, promedios y SD, tal como se pueden ver en la Tabla **7-15**.

Tabla 7-15: Tiempos promedio transcurridos en segundos con su correspondiente SD para la predicción de 50 imágenes.

Cuadrícula	Promedio	SD
256×256	0.5540	0.0293
128×128	2.3022	0.0816
64×64	10.8033	0.1350

Con respecto a la predicción de las imágenes, que sería el proceso utilizado por los SIVA en una línea de producción, se puede resaltar que para una división de las imágenes con una cuadrícula de 256×256 el sistema requiere medio segundo para realizar una predicción, mientras que para una cuadrícula de 128×128 le toma un poco más de dos segundos, tiempos que podrían funcionar razonablemente bien para algunas líneas de producción, y finalmente, para la cuadrícula de 64×64 el sistema tarda menos de once segundos, tiempo que sería poco útil para su uso en muchos SIVA. Aunque no es uno de los objetivos de este trabajo, la aceleración de algoritmos mediante diferentes estrategias como las empleadas en [Zhu and Jiang. 2018; Jacob et al.. 2019; Villegas-Jaramillo et al.. 2023] pueden ser aplicadas, así como el uso de equipos de cómputo con mayor velocidad de procesamiento permitiendo así obtener mejores tiempos de respuesta y haciendo posible su uso en aquellos SIVA cuyo tiempo de respuesta deba ser menor.

7.6 Conclusiones y trabajos futuros

En este capítulo se presentó una metodología débilmente supervisada para la clasificación de objetos, localización de defectos e interpretación gráfica de defectos, basada principalmente en técnicas de IA y técnicas de procesamiento de imágenes. La metodología propuesta en sus dos diferentes variantes presenta buenos resultados de clasificación, con precisiones promedio de 0.9722, 0.9817 y 0.8608 para los conjuntos de datos DAGM C2, DAGM C5 y AITEX respectivamente, lo que representa mejoras de 12.35 %, 4.6 % y 0.84 %, respecto a la metodología base. En cuanto a TP-rate y FP-rate, valores de 0.9374 y 0.0296 para DAGM C2, 0.9886 y 0.0342 para DAGM C5 y 0.8631 y 0.1522 para AITEX, superan ampliamente los resultados obtenidos con la metodología de línea base, demostrando así la utilidad y capacidad de la metodología propuesta.

La metodología demuestra que la extracción de características con CNN+TL es efectiva en diferentes escenarios y conjuntos de datos, mientras que la metodología de referencia con extracción de características de bajo nivel no es adecuada ni efectiva para realizarse directamente sin aplicar primero algoritmos de segmentación, particularmente en problemas con fondos borrosos o difusos como los presentados en los conjuntos de datos DAGM-C2 y DAGM-C5. En general, la metodología de línea base con extracción de características de bajo nivel solo obtiene resultados competitivos, respecto a los logrados con la metodología propuesta, en imágenes cuyos defectos están mejor definidos como algunos de los defectos en el conjunto de datos de AITEX.

Una de las principales ventajas de la metodología propuesta es que no utiliza algoritmos de segmentación que requieran procedimientos complejos, ni algoritmos de propuesta de regiones tradicionales o basados en DL, ni la supervisión por parte de expertos que crean las anotaciones y máscaras que delimitan los defectos. En cambio, la metodología utiliza una estrategia ingenua que consiste en la creación de cuadrículas regulares, lo que permite —de manera simple, aunque los bloques no enmarquen completamente el defecto— clasificar objetos y localizar defectos con buenos resultados. Otras ventajas de la metodología propuesta son que no requiere grandes volúmenes de imágenes para el entrenamiento, presenta un buen desempeño sobre conjuntos de datos cuyas clases están desbalanceadas, la posibilidad de considerar imágenes de diferentes tamaños, así como poder seleccionar cuadrículas con múltiples resoluciones según los tamaños de las imágenes y de los defectos considerados. Adicionalmente, la metodología propuesta permite la interpretación gráfica de los resultados de la localización, explicando así por qué se asignó la etiqueta a una determinada imagen.

Los resultados de los experimentos sobre los tres conjuntos de datos en el proceso de clasificación de objetos demuestran la efectividad de la metodología propuesta en comparación con la línea de base; además, se analizan y discuten ampliamente las propiedades y el comportamiento de los diferentes métodos utilizados para la propuesta de región, la extracción de características, la creación de bolsas y el cálculo de disimilitud. Finalmente, se presenta, discute y analiza ampliamente una propuesta de localización de defectos con una explicación gráfica logrando buenos resultados. Con respecto al análisis computacional es importante anotar que dados los dos procesos desarrollados, el primero para entrenar la red neuronal y extraer las características y, el segundo, encargado de la predicción de una imagen, la complejidad de los dos procesos depende principalmente de la red utilizada, en este caso InceptionResNetV2. En cuanto a los tiempos de respuesta se puede afirmar que la metodología propuesta, y específicamente los algoritmos de predicción podrían funcionar bien en diferentes escenarios de SIVA.

Si bien en el capítulo 6 se propuso una estrategia de paralelización para el cálculo de las disimilitudes entre bolsas, dicha estrategia no se utilizó aquí debido a que solo es beneficiosa para bolsas con muchas instancias y, en los conjuntos de datos utilizados surgen a lo sumo 64 instancias para las dos variantes del conjunto DAGM y 256 instancias para el conjunto AITEX, en ambos casos con la cuadrícula más pequeña (64×64 píxeles). No obstante, en caso de considerar conjuntos de datos con imágenes de alta resolución y defectos muy pequeños por localizar, podrían surgir bolsas con muchas instancias (por ejemplo del orden de miles de instancias o más), así también cuando la descomposición en bloques se hace con traslape entre ellos mediante múltiples ventanas deslizantes siendo en tales casos útil la paralelización propuesta.

Por último, como trabajos futuros, primero, si se quisiera aumentar los índices de desempeño de clasificación, se podrían utilizar otros clasificadores más potentes y robustos que el de k-NN, por ejemplo utilizando las SVM o las mismas CNN; sin embargo, mejorar el desempeño de clasificación de esta manera se haría a expensas de perder parte de la información necesaria para la localización de los defectos y la interpretación de los resultados.

Como segundo trabajo futuro, sería importante la aplicación de estrategias de aceleración de algoritmos con el fin de

reducir los tiempos de respuesta en la predicción de nuevas imágenes especialmente al utilizar imágenes más grandes o cuadrículas más pequeñas, lo que permitiría la aplicación de la metodología en gran cantidad de escenarios.

Parte V

Conclusiones y trabajo futuro

8 Conclusiones generales

En el desarrollo de esta tesis se presentaron y discutieron algunos de los principales problemas que acarrean los SIVA y que han limitado su adopción y mejor aprovechamiento al interior de la industria manufacturera. Específicamente, se utilizó el enfoque del aprendizaje inexactamente supervisado, buscando solucionar problemas como: reducir el tiempo y el costo requerido por los expertos en el etiquetado manual de las imágenes necesarias para entrenar un sistema de inspección visual, la falta de imágenes suficientes para entrenar adecuadamente los modelos de aprendizaje de máquina, la necesidad de mantener o superar los resultados en cuanto precisión y exactitud de los clasificadores, y la necesidad de disponer de herramientas prácticas para interpretar los resultados obtenidos de un modelo. En este sentido, se propusieron diferentes estrategias que incluyen dos métodos de representación para la información de las imágenes utilizando las características extraídas, el primero con la extracción de características de bajo nivel, y el segundo, realizando la extracción de características con una CNN permitiendo así mediante el uso de disimilitudes, experimentar con diferentes técnicas que incluyen el uso de DL, TL, DA, para posteriormente utilizar diferentes clasificadores.

Cada uno de los anteriores elementos fueron utilizados para conformar una metodología heterogénea para ser aplicada en sistemas de inspección visual, buscando afrontar la problemática de la detección de defectos en superficies texturizadas, donde los principales logros fueron los siguientes:

- Se desarrolló de una metodología heterogénea para la inspección visual automática basada en técnicas de aprendizaje inexactamente supervisado. Esta metodología ha demostrado ser efectiva en la clasificación y detección de patrones y defectos en una variedad de aplicaciones y conjuntos de datos.
- Se pudo corroborar que, métodos y sistemas de clasificación basados en técnicas clásicas diferentes a las mismas CNN, obtienen resultados comparativos y competitivos en cuanto a exactitud y precisión en diferentes problemas de clasificación y detección, donde la posibilidad de interpretación se presenta como un valor agregado, lo que la hace aplicable a diferentes situaciones y problemas de la industria.
- Las diferentes metodologías desarrolladas, incluyendo la heterogénea, se han evaluado en una amplia gama de aplicaciones, incluyendo: clasificación de defectos en láminas de vidrio (Capítulo 4), detección de defectos en telas y en conjuntos sintéticos (Capítulo 7), y detección de defectos en dulces (Capítulo 5), demostrando su capacidad de adaptación a diferentes condiciones y conjuntos de datos.
- Un logro destacado de esta tesis, es la integración de la interpretabilidad a la metodología heterogénea (Capítulo 7), la cual permite comprender de forma gráfica cómo se toman las decisiones al interior de los procesos de inspección visual, lo que es crucial en aplicaciones donde la transparencia es crítica permitiendo así determinar la utilidad y funcionamiento del modelo.
- El uso de la metodología con una ventana deslizante, como complemento para la metodología presentada en el capítulo 7, permite una mejor detección y localización de defectos en ciertos tipos de conjuntos de datos.
- Con respecto al alto costo computacional de los clasificadores MIL, que se detectó en la revisión de la literatura, se presentó en el Capítulo 6, una estrategia para la paralelización del cálculo de las disimilitudes entre conjuntos de puntos, representados bajo bolsas con muchas instancias, con buenos resultados.
- Sobre las limitaciones y alcance de la metodología, se pudo constatar que no funciona bien con objetos donde los defectos son tolerancias o dimensiones y si los defectos se confunden significativamente con la superficie o textura del objeto. Algunos ejemplos de este tipo de problemas se dan en la detección de defectos de fabricación en brocas [Zhang et al.. 2006] y en la inspección de calidad en colectores eléctricos [Tabernik et al.. 2019], con los cuales se hicieron pruebas preliminares con resultados poco satisfactorios. Otra limitación se observa cuando

los defectos presentan tamaños variables que pueden abarcar desde solo una pequeña porción hasta la totalidad de la imagen, como en el caso del conjunto de datos AITEX (ver sección 7.3.1).

9 Trabajos futuros

Habiendo desarrollado la metodología heterogénea propuesta y enfrentado los diferentes problemas que se presentan en la clasificación y localización de defectos, se pueden explorar algunas áreas que permitan un mejor desarrollo del trabajo realizado. Los diferentes trabajos futuros se listan a continuación, haciendo referencia a la etapa de la metodología a la cual pertenecen de acuerdo con lo mostrado en el diagrama de la Figura **9-1**:

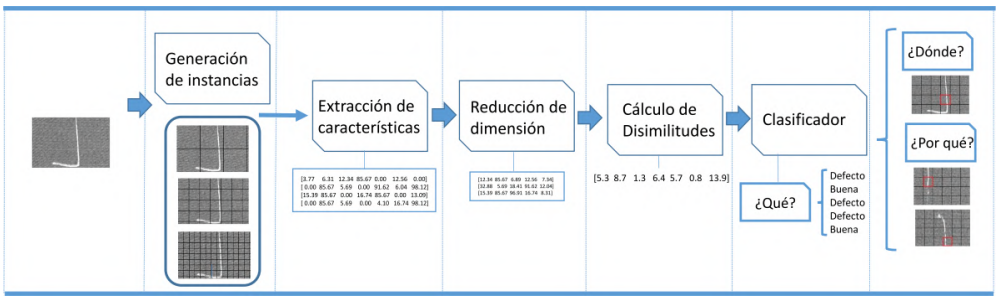


Figura 9-1: Diagrama de las etapas de la metodología con las que se asocian los trabajos futuros.

- En cuanto a la **generación de instancias**, como una primera etapa, se pueden utilizar métodos alternativos como la ventana deslizante (ver Anexo A), el uso de R-CNN, la búsqueda selectiva entre otros, que permitan contar con diferentes opciones para generar el conjunto de instancias que conforman una determinada imagen.
- Una vez generadas las instancias se procede a la **extracción de características**, la cual se puede alternar entre métodos tradicionales con diferentes opciones y nuevas arquitecturas de redes neuronales que permitan mejores desempeños en otros conjuntos de datos.
- Teniendo en cuenta que la generación de instancias puede producir vectores o bolsas de altas dimensiones, podría ser necesario utilizar un proceso de **reducción de dimensión**, que para nuestro trabajo se centró en PCA con 50 características, siendo posible probar diferentes combinaciones que van desde cambiar el número de características con PCA, utilizar otros métodos de reducción como el análisis de discriminantes lineales (LDA), el análisis de componentes independientes (ICA), los autoencoders entre otros, o incluso no utilizar reducción alguna.
- Para el posterior **cálculo de las disimilitudes**, que para este trabajo se desarrolló bajo funciones basadas en distancias, se considera importante explorar el uso de otras medidas de disimilitud entre conjuntos de puntos, que podrían ser aplicadas dependiendo de los conjuntos de datos, así como la posibilidad de explorar otras alternativas como la comparación directa entre imágenes o la construcción de un espacio de disimilitudes.
- En cuanto a la etapa de **clasificación**, se exploró el uso de k-NN, el cual podría ser reemplazado por otros clasificadores más robustos como diversas variantes de las SVM, así como las diferentes versiones de clasificadores MIL o variantes de CNN que podrían ofrecer mejores resultados en cuanto a exactitud y precisión, pero sacrificando información importante para la localización e interpretación de los defectos.
- Sobre la **interpretación**, se considera viable explorar combinaciones de técnicas que incluyan los mapas de activación de clase e información proveniente de clasificadores como el del vecino más cercano ($k - NN$).

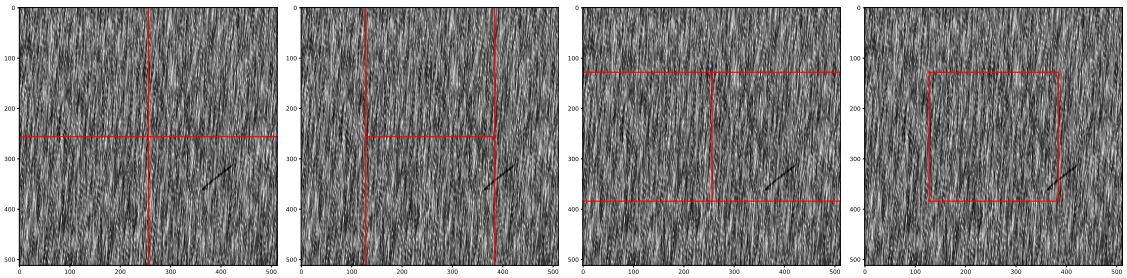
- Otras posibles vías de investigación incluyen la proposición de metodologías heterogéneas basadas en ensambles de MIL y DL, fusionándolos en diferentes etapas del proceso de clasificación; es decir, considerando rutas que se combinen en etapas tempranas o tardías como por ejemplo en la extracción de características, en las probabilidades a posteriori de los clasificadores o en las decisiones finales de éstos.
- Finalmente, de manera global y transversal a diferentes etapas de la metodología, se puede explorar la aplicación de estrategias de aceleración de algoritmos que podría beneficiar el uso de la metodología propuesta directamente en los SIVA, especialmente al momento de realizar la predicción de nuevas imágenes.

A Metodología inexactamente supervisada para la detección y localización de defectos utilizando una ventana deslizante

A.1 Problemática

Este anexo se desarrolló con el fin de mejorar la exactitud y calidad de la detección de defectos de la *Metodología inexactamente supervisada basada en aprendizaje de múltiples instancias, redes neuronales y disimilitudes entre bolsas* presentada en el capítulo 7. Una limitación de la metodología allí propuesta radica en que la posición de las cuadrículas utilizadas es fija, por lo cual la localización de los defectos puede no ser muy precisa y muchas veces puede quedar desplazada con respecto al defecto. Para tratar de solucionar dicha problemática se determinó utilizar un sistema de ventana deslizante que permitiera aumentar la cantidad y ubicación de las diferentes cuadrículas y así localizar los defectos de una manera más precisa, y más centrada con respecto a la ventana. Para cubrir completamente las imágenes con la ventana deslizante, se determinó desplazar la ventana en una cantidad de píxeles igual a la mitad del ancho de la cuadrícula, primero sobre el eje X, luego sobre el eje Y y finalmente sobre los dos ejes de forma simultánea.

Para la división de una imagen de 512×512, utilizando una cuadrícula de 256×256, se generan las 4 particiones originales como se aprecia en la Figura **A-1**a y, posteriormente se generan 2 nuevas particiones desplazadas 128 píxeles en el eje X (ver Figura **A-1**b), más otras 2 nuevas particiones desplazadas 128 píxeles en el eje Y (ver Figura **A-1**c). Finalmente, se genera otra partición que resulta de desplazamientos de 128 píxeles en el eje X y 128 píxeles en el eje Y (ver Figura **A-1**d). El total de regiones resultantes es de 9.

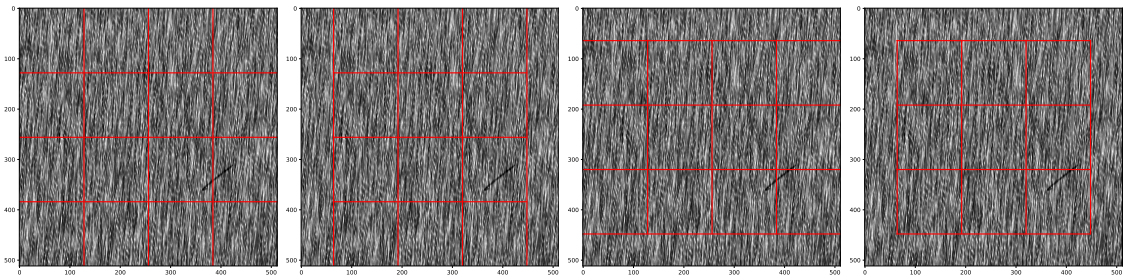


(a) Originales (b) Desplazadas en X (c) Desplazadas en Y (d) Desplazada en X y Y

Figura A-1: Regiones resultantes para una división de 256×256 píxeles.

Para dividir la misma imagen pero con una cuadrícula de 128×128, se generan las 16 particiones originales como se

aprecia en la Figura **A-2**a, así como 12 nuevas particiones desplazadas 64 píxeles en el eje X (ver Figura **A-2**b), más 12 nuevas particiones desplazadas 64 píxeles en el eje Y (ver Figura **A-2**c) y, finalmente 9 nuevas particiones desplazadas 64 píxeles en el eje X y 64 píxeles en el eje Y (ver Figura **A-2**d), para un total de 49 posibles regiones.



(a) Originales (b) Desplazadas en X (c) Desplazadas en Y (d) Desplazadas en X y Y

Figura A-2: Regiones resultantes para una división de 128×128 píxeles.

Es importante tener en cuenta que, al aumentar la cantidad de particiones, el costo computacional y por lo tanto el tiempo necesario para realizar la clasificación, detección y localización aumenta de forma considerable.

A.2 Configuración experimental

Teniendo en cuenta la gran cantidad de particiones que se generan al utilizar la ventana deslizante con el costo computacional que conlleva, se decidió probar esta variante de la metodología heterogénea con los conjuntos de datos DAGM C2 y DAGM C5 debido a que contienen imágenes cuadradas con una relativa baja resolución de 512×512 píxeles, que se ajusta bien a las diferentes ubicaciones de la ventana. Dado el tamaño de las imágenes, se seleccionó el uso de retículas de 256×256 y 128×128, ya que se genera un conjunto de particiones bajo y manejable por la metodología, y finalmente, se seleccionó la red InceptionResNetV2 para la extracción de características, PCA para reducir la cantidad de características a 50 y como clasificador k-NN (con $k = 3$) debido a los buenos resultados que produjeron al ser utilizadas con la versión original de la metodología.

Los demás pasos de la configuración experimental son iguales a los expuestos en la Sección 7.2 y, para el cálculo de las diferentes métricas, se ejecutó la metodología sobre cinco permutaciones de cada uno de los dos conjuntos de datos y sobre los dos tamaños de cuadrícula disponibles 256×256 y 128×128. En cuanto a plataformas de cómputo y medidas de desempeño, se utilizaron las mismas previamente referenciadas en la Sección 7.3.2.

A.3 Detección de defectos

Los resultados en cuanto a exactitud de clasificación obtenidos con la variante de la metodología, presentados en la Tabla **A-1**, y en comparación con los resultados originales (es decir, en los que no usa una ventana deslizante y que fueron presentados en la Tabla **7-3**) demuestran que la extracción de características usando CNN+TL (InceptionResNetV2) con ventana deslizante produce mejores resultados. Se destaca que, para el conjunto de datos DAGM C2, se logra una exactitud de 1.0 utilizando imágenes divididas con una cuadrícula de 256×256 en contraste con el valor de 0.9679 con la metodología original; por otro lado, para el conjunto de datos DAGM C5, el mejor resultado de 0.9826 se obtiene utilizando una cuadrícula de 128×128 que supera por poco el valor de 0.9817 obtenido con la cuadrícula fija de 256×256.

La exactitud no es la medida de desempeño más apropiada cuando existe gran desbalance entre clases, como el que se presenta en estos dos conjuntos datos (87 % sin defecto y 13 % con defecto). Para una mejor estimación del desempeño de clasificación en casos de desbalance de clases, Bramer [2020] sugiere calcular la tasa de verdaderos

Tabla A-1: Resultados de exactitud para los conjuntos de datos DAGM C2 y DAGM C5 utilizando Inception-ResNetV2 para la extracción de características y una cuadrícula deslizante, primero de tamaño 256×256 y a continuación de 128×128 píxeles.

	Exactitud	
Tamaño	DAGM C2	DAGM C5
256×256	1.0	0.9782
128×128	0.9956	0.9826

positivos (TP-rate) y la tasa de falsos positivos (FP-rate). Los resultados de TP-rate y FP-rate para los dos conjuntos DAGM, utilizando InceptionResNetV2 para la extracción de características y resoluciones para la descomposición en bloques de 256×256 y 128×128, se presentan en la Tabla **A-2** y que se pueden comparar con los obtenidos en la metodología sin ventana deslizante que fueron presentados previamente en la Tabla **7-4**.

Tabla A-2: TP-rate y FP-rate para los conjuntos de datos DAGM C2 y DAGM C5.

	DAGM C2		**DAGM C5**	
Tamaño	**TP-rate**	**FP-rate**	**TP-rate**	**FP-rate**
256×256	1.0	0.0	0.8695	0.0096
128×128	1.0	0.0051	1.0	0.02

Se puede apreciar que, en concordancia con lo observado para la exactitud, los valores de TP-rate para los dos conjuntos de datos utilizando la variante de la metodología son superiores y se alcanzan utilizando cuadrículas de 256×256 y 128×128 con valores de 1.0. En cuanto al FP-rate, los mejores resultados en el conjunto de datos DAGM C2 se consiguen al utilizar una cuadrícula de 256×256, alcanzando un valor de 0.0; para el conjunto de datos DAGM C5, el mejor resultado es de 0.0096, también con una cuadrícula de 256×256.

Los resultados de clasificación, para una permutación del conjunto de datos, se presentan en términos de las correspondientes matrices de confusión en la Tabla **A-3**, así como en los reportes de clasificación en la Tabla **A-4**. Nótese, en estas últimas dos tablas, que la clasificación para el conjunto de datos DAGM C2 se realiza sobre 35 imágenes con defecto y 195 sin defecto, mientras que para el conjunto de datos DAGM C5 se realiza sobre 30 imágenes con defecto y 200 sin defecto.

Tabla A-3: Matrices de confusión para una permutación en los conjuntos de datos DAGM C2 (cuadrícula de 256×256) en el lado izquierdo y DAGM C5 (cuadrícula de 128×128) en el lado derecho.

		Predecida	
	DAGM C2	**Defecto**	**No defecto**
Actual	Defecto	35	0
	No defecto	0	195

		Predecida	
	DAGM C5	**Defecto**	**No defecto**
Actual	Defecto	30	0
	No defecto	4	196

Para el conjunto de datos DAGM C2 dado que la exactitud fue de 1.0 se observan valores de 1.0 en todas las métricas lo que refleja que el modelo detecta muy bien las imágenes sin defecto y con defecto. En contraste, para el conjunto de datos DAGM C5, las métricas de desempeño presentan resultados inferiores a los observados para el conjunto de datos DAGM C2, encontrando valores de precisión de 0.8823 y 1.0 para las clases con defecto y sin defecto respectivamente, lo que indica que la metodología clasifica bien las imágenes sin defecto, pero no siempre logra hacerlo para las imágenes con defecto, lo cual se corrobora al encontrar valores-F de 0.9375 para las imágenes con defecto y de 0.9899 para las imágenes sin defecto.

Tabla A-4: Reporte de clasificación para una permutación de los conjuntos de datos DAGM C2 y DAGM C5.

	DAGM C2			
	Precisión	**exhaustividad**	**Valor-F**	**Soporte**
Defecto	1.0	1.0	1.0	35
No defecto	1.0	1.0	1.0	195
Promedio macro	1.0	1.0	1.0	230
Promedio ponderado	1.0	1.0	1.0	230
	DAGM C5			
	Precisión	**exhaustividad**	**Valor-F**	**Soporte**
Defecto	0.8823	1.0	0.9375	30
No defecto	1.0	0.98	0.9899	200
Promedio macro	0.9411	0.99	0.9637	230
Promedio ponderado	0.9846	0.9826	0.9831	230

A.4 Localización de defectos

La capacidad de localización de defectos con la variante de la metodología propuesta se ilustra a continuación para una de las permutaciones del conjunto de prueba, realizada sobre los dos conjuntos de datos y con las cuadrículas de 256×256 y de 128×128.

Con respecto a los conteos para medir los resultados de localización, se utilizó la escala de graduaciones previamente definida en la Sección 7.2.6. Los resultados obtenidos con la variante de la metodología se presentan en la Tabla **A-5**, los cuales se pueden comparar con los resultados sin utilizar la ventana deslizante que se presentaron en la Tabla **7-7**.

Tabla A-5: Conteos de localización para una permutación en los conjuntos de datos DAGM C2 y DAGM C5

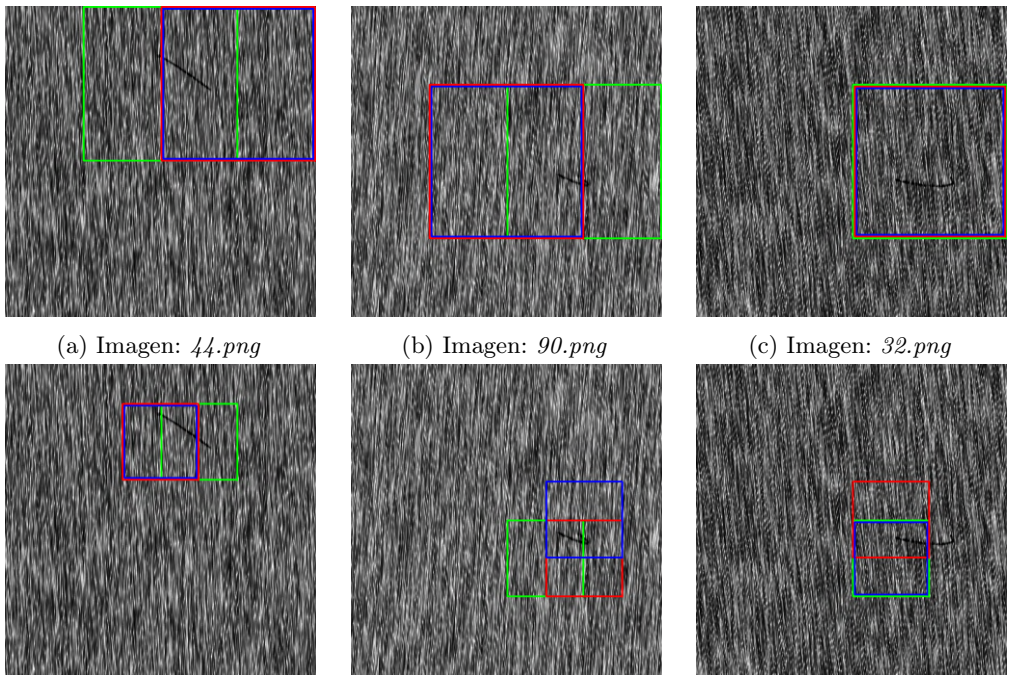
	DAGM C2		DAGM C5	
Localización	256×256	128×128	256×256	128×128
Completamente localizados	33	34	30	35
Adecuadamente localizados	1	1	3	0
Deficientemente localizados	1	0	1	1
No localizados	0	0	2	0
Total	35	35	36	36

Como se puede identificar para el conjunto de datos DAGM C2, los mejores resultados se alcanzaron utilizando una cuadrícula de 128×128. De 35 imágenes clasificadas, las 35 están bien localizadas (34 completamente y 1 adecuadamente), lo que corresponde al 100.0 % de localizaciones correctas frente a un valor de 88.23 % de localizaciones correctas sin utilizar la ventana deslizante. No resultaron imágenes en las cuales la localización fuera deficiente o no pudieran ser localizados los defectos con ninguna de las tres predicciones.

Análogamente, para el conjunto de datos DAGM C5, los mejores resultados se alcanzaron utilizando una cuadrícula de 128×128. De 36 imágenes clasificadas, 35 están bien localizadas (35 completamente y 0 adecuadamente), correspondiendo a un desempeño de localizaciones correctas del 97.22 % que mejora levemente el 96.97 % logrado sin utilizar la ventana deslizante. Una imagen está clasificada por sólo una de las 3 predicciones lo que corresponde al 2.77 % y en ninguna imagen no se pudo localizar el defecto con ninguna de las tres predicciones. Nótese que los resultados obtenidos para estos conteos de localización, en los dos conjuntos de datos DAGM, presentan un comportamiento coherente con el observado para los resultados de clasificación reportados anteriormente.

En la Figura **A-3** se pueden apreciar algunos resultados gráficos de la localización de defectos para el conjunto de datos DAGM C2, en los cuales se adoptó la convención de visualización explicada en la Sección 7.2.6. Es importante aclarar que en la fila superior con las Figuras **A-3**a, **A-3**b y **A-3**c se presenta la localización de defectos para tres

imágenes utilizando una cuadrícula de 256×256 y que, en la fila inferior con las Figuras **A-3**d, **A-3**e y **A-3**f, se presenta la localización de defectos en las mismas tres imágenes pero utilizando una cuadrícula de 128×128. Para las tres imágenes en que se realizó la localización del defecto haciendo uso de los dos tamaños de cuadrículas, es evidente que las tres cuadrículas resultantes coinciden sobre el defecto desde diferentes posiciones de las cuadrículas para defectos completamente localizados.



(a) Imagen: *44.png* (b) Imagen: *90.png* (c) Imagen: *32.png*

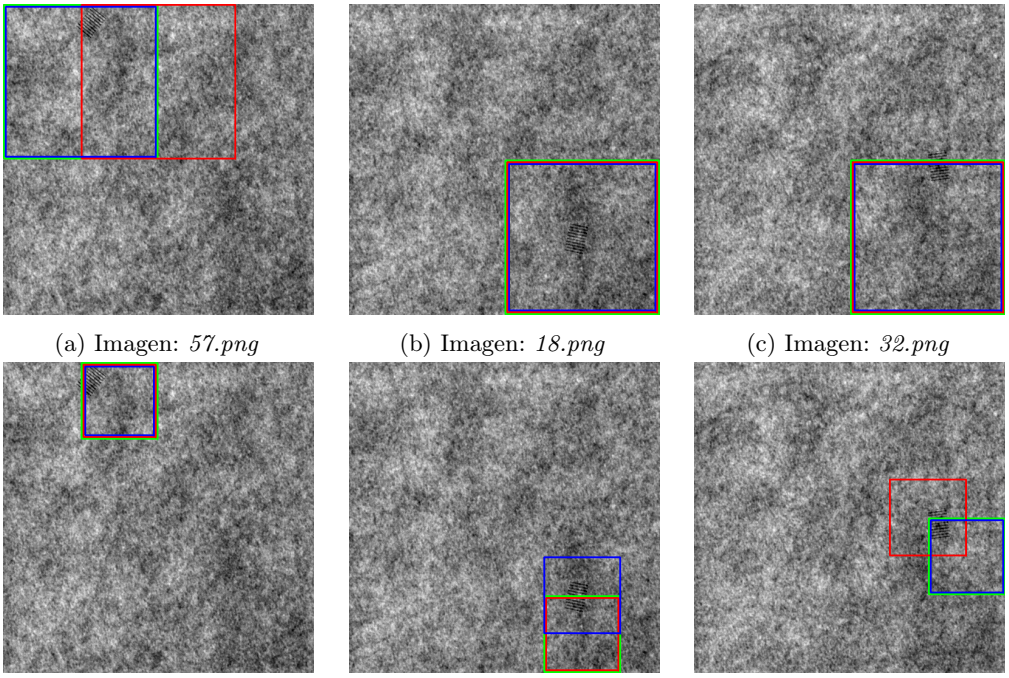
(d) Imagen: *44.png* (e) Imagen: *90.png* (f) Imagen: *32.png*

Figura A-3: Resultado gráfico de la localización de defectos en el conjunto de datos DAGM C2 utilizando cuadrículas de 256×256 en la fila superior y 128×128 en la fila inferior, presentando defectos completamente localizados.

De forma similar, en la Figura **A-4** se ilustra la localización de defectos para tres imágenes pertenecientes al conjunto DAGM C5, utilizando los dos tamaños de cuadrículas, donde todos los defectos están completamente localizados. Se destaca en la primera fila donde se utiliza una división de 256×256, para la imagen en la Figura **A-4**a que el defecto es localizado por dos posiciones de cuadrículas diferentes, mientras que en las Figuras **A-4**b y **A-4**c, los defectos son localizados desde la misma posición de la cuadrícula. Para la segunda fila que contiene las localizaciones de las mismas tres imágenes sobre una cuadrícula de 128×128, la Figura **A-4**d presenta un defecto localizado por tres cuadrículas ubicadas en la misma parte del defecto, así como en las Figuras **A-4**e y **A-4**f dos de las cuadrículas enmarcan una misma porción del defecto mientras que la tercer cuadrícula enmarca una porción diferente del defecto.

A.5 Interpretación de defectos

Con el fin de explicar el motivo de la asignación de una etiqueta de clase a una determinada imagen en los conjuntos de datos DAGM C2 y DAGM C5, se utilizan las convenciones definidas previamente en las Secciones 7.2.6 y 7.2.7. En las Figuras **A-5** y **A-6** se presenta la interpretación de la localización de un mismo defecto perteneciente al conjunto de datos DAGM C2 (imagen *102.png*) donde en la Figura **A-5**a, utilizando una cuadrícula de 256×256, se puede observar que el defecto está completamente localizado; además, se puede ver que los defectos localizados en las tres imágenes del conjunto de entrenamiento (presentados en las Figuras **A-5**b, **A-5**c y **A-5**d) corresponden o son similares a las secciones del mismo color señaladas en la imagen de prueba, donde todas las cuadrículas enmarcan defectos similares. En la Figura **A-6** se localiza e interpreta en la imagen *102.png* el mismo defecto utilizando una cuadrícula de 128×128, donde en la Figura **A-6**a se aprecia que el defecto está completamente localizado, con dos de



(a) Imagen: *57.png* (b) Imagen: *18.png* (c) Imagen: *32.png*

(d) Imagen: *57.png* (e) Imagen: *18.png* (f) Imagen: *32.png*

Figura A-4: Resultado gráfico de la localización de defectos en el conjunto de datos DAGM C5, utilizando cuadrículas de 256×256 en la fila superior y de 128×128 en la fila inferior, donde todas las figuras contienen imágenes que presentan defectos completamente localizados.

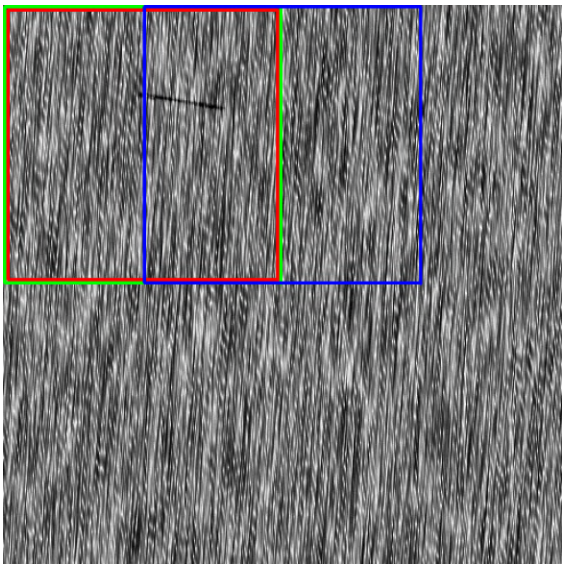
las cuadrículas enmarcando la misma porción del defecto y la tercera cuadrícula enmarcando otra porción diferente del mismo. En las tres figuras restantes **A-6**b, **A-6**c y **A-6**d se puede apreciar que se localizan defectos, pertenecientes al conjunto de entrenamiento, que son similares a las porciones del mismo color resaltadas en la imagen de prueba.

De forma análoga, en las Figuras **A-7** y **A-8** se puede visualizar la interpretación de los resultados de detección y localización de un defecto en una imagen de prueba con forma de patrón de líneas correspondiente al conjunto de datos DAGM C5 (imagen *99.png*) utilizando las dos resoluciones de cuadrícula. En la Figura **A-7**a, se localiza el defecto utilizando una cuadrícula de 256×256, el cual está completamente localizado desde dos ubicaciones diferentes; además, en las Figuras **A-7**b, **A-7**c y **A-7**d, se aprecian imágenes pertenecientes al conjunto de entrenamiento, con defectos que están claramente enmarcados y que presentan características similares a aquéllas del defecto de la imagen de prueba en la Figura **A-7**a. En la Figura **A-8** se muestra la localización del mismo defecto pero haciendo uso de una cuadrícula de 128×128, donde en la Figura **A-8**a se presenta un defecto completamente localizado desde dos posiciones diferentes y las demás Figuras **A-8**b, **A-8**c y **A-8**d contienen defectos localizados que pertenecen al conjunto de entrenamiento y que son similares al defecto presentado en la imagen de prueba.

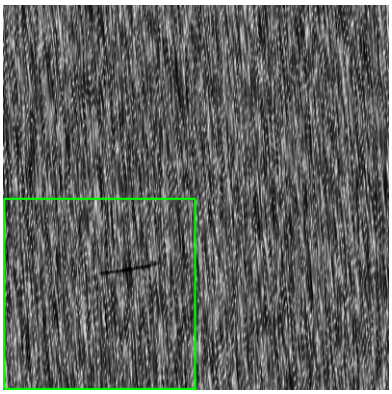
A.6 Conclusiones

Los resultados alcanzados por la variante de la metodología que utiliza ventana deslizante, comparados con los obtenidos sin utilizarla permiten afirmar lo siguiente:

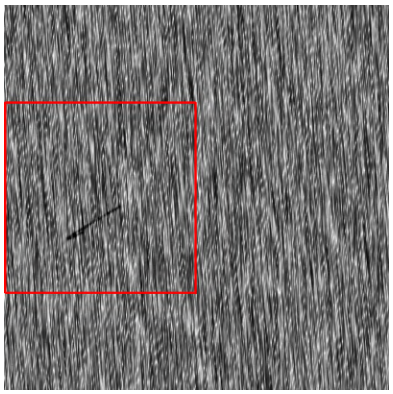
- Para la exactitud en los conjuntos DAGM C2 y DAGM C5 los nuevos valores obtenidos de 1.0 y 0.9826 mejoran los resultados de 0.9722 y 0.9817 obtenidos sin utilizar la ventana deslizante.
- En cuanto a TP-rate y FP-rate los valores obtenidos por la variante de la metodología de (1.0 y 0.0) para DAGM C2 y (1.0 y 0.02) para DAGM C5 son mejores con respecto a los valores de (0.9374 y 0.0296) para



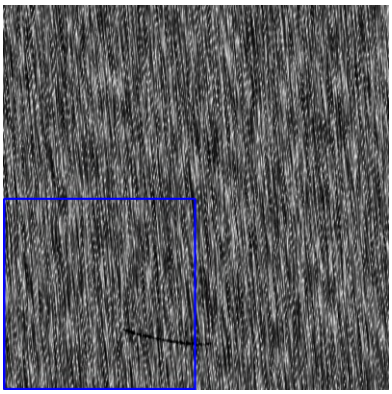
(a) Imagen de prueba



(b) Primer bloque más cercano

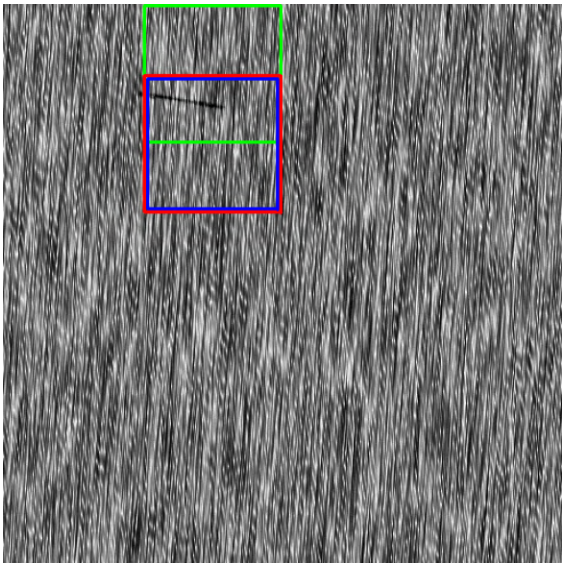


(c) Segundo bloque más cercano

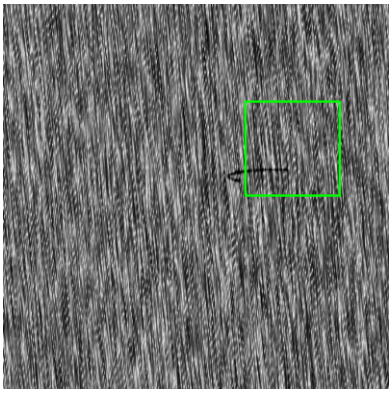


(d) Tercer bloque más cercano

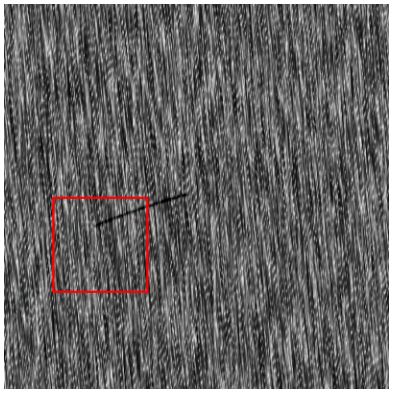
Figura A-5: Interpretación gráfica de la localización de un defecto en una imagen *102.png* perteneciente al conjunto de datos DAGM C2, utilizando una retícula de 256×256, donde la Figura (a) es la imagen de prueba en la cual se localiza completamente el defecto que tiene forma de línea; las Figuras (b)—(d) contienen regiones que demarcan segmentos de líneas, similares a las regiones del mismo color en la imagen de prueba.



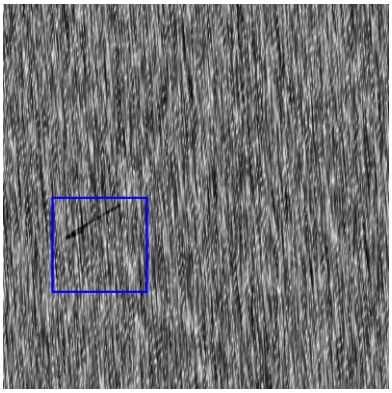
(a) Imagen de prueba



(b) Primer bloque más cercano

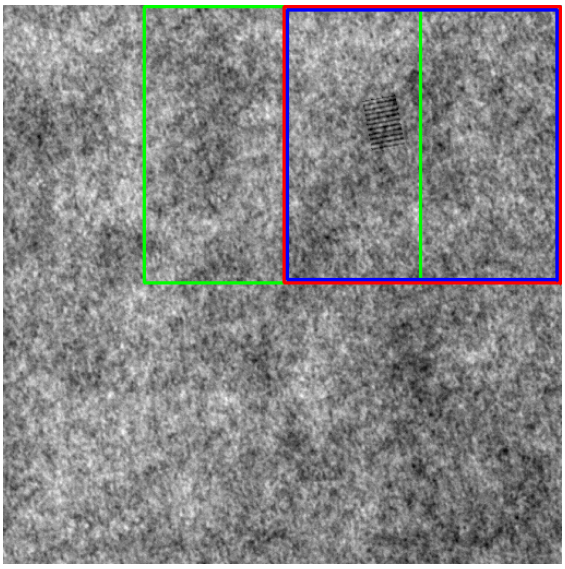


(c) Segundo bloque más cercano

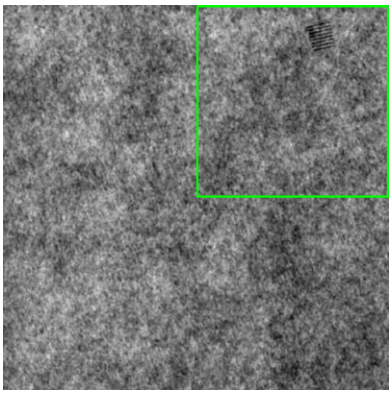


(d) Tercer bloque más cercano

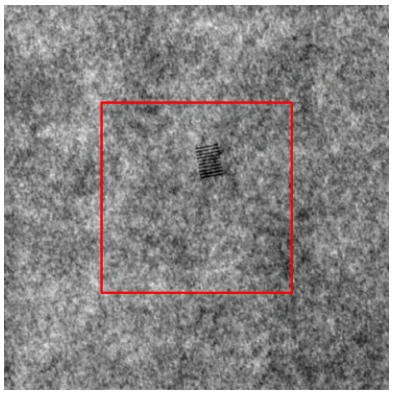
Figura A-6: Interpretación gráfica de la localización de un defecto en la imagen *102.png* perteneciente al conjunto de datos DAGM C2, utilizando una cuadrícula de 128×128, donde la Figura (a) es la imagen de prueba en la cual se localiza completamente el defecto que tiene forma de línea; las Figuras (b)—(d) contienen regiones que demarcan segmentos de líneas, similares a las regiones del mismo color en la imagen de prueba.



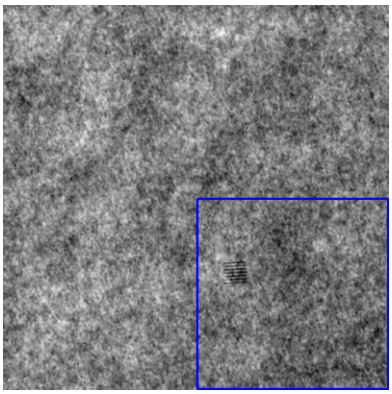
(a) Imagen de prueba



(b) Primer bloque más cercano

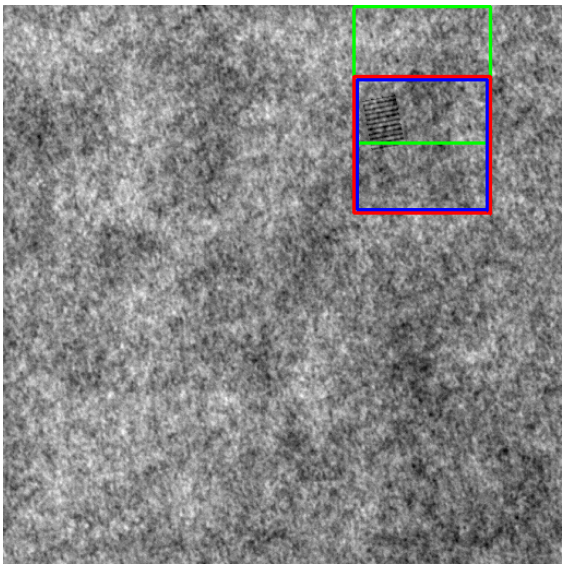


(c) Segundo bloque más cercano

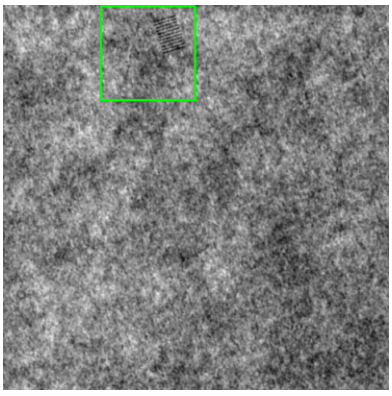


(d) Tercer bloque más cercano

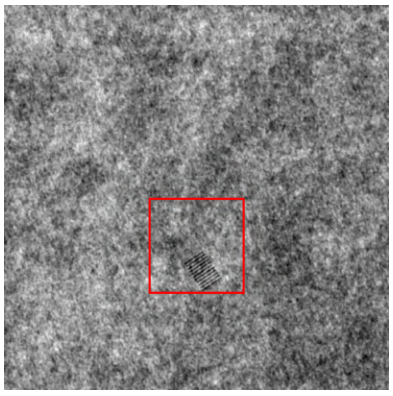
Figura A-7: Interpretación gráfica de la localización de un defecto en la imagen *99.png* perteneciente al conjunto de datos DAGM C5, utilizando una cuadrícula de 256×256, donde la Figura (a) es la imagen de prueba en la cual se localiza completamente el defecto en forma de patrón de líneas; las Figuras (b)—(d) contienen regiones similares a las regiones del mismo color en la imagen de prueba.



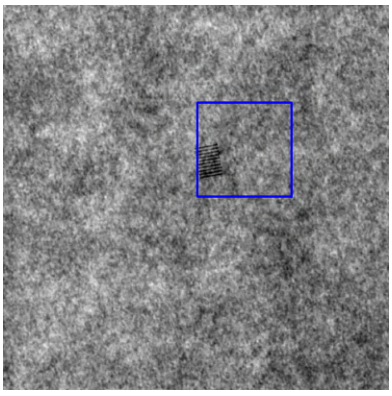
(a) Imagen de prueba



(b) Primer bloque más cercano



(c) Segundo bloque más cercano



(d) Tercer bloque más cercano

Figura A-8: Interpretación gráfica de la localización de un defecto en la imagen *99.png* perteneciente al conjunto de datos DAGM C5, utilizando una cuadrícula de 128×128, donde la Figura (a) es la imagen de prueba en la cual se localiza completamente el defecto en forma de patrón de líneas; las Figuras (b)—(d) contienen regiones similares a las regiones del mismo color en la imagen de prueba.

DAGM C2 y (1.0 y 0.0423) para DAGM C5.

- Con respecto a los conteos de localización también se evidencia una mejora, con porcentajes de localizaciones correctas de 100.0 % y 97.22 % para DAGM C2 y DAGM C5, utilizando la ventana deslizante, contra 88.23 % para DAGM C2 y 96.97 % sin utilizarla.
- Visualmente es posible apreciar que con la ventana deslizante algunos defectos quedan mejor enmarcados dentro de la ventana, y que muchas veces el sobrelapamiento de dos ventanas enmarca mejor los defectos.

Dado lo anterior se puede concluir que el uso de una ventana deslizante, como complemento para la metodología presentada en el capítulo 7, mejora los resultados en las diferentes métricas permitiendo así una mejor detección y localización de defectos en ciertos tipos de conjuntos de datos.

Es importante mencionar que, aunque no se calculan, los tiempos y complejidad de la variante de la metodología propuesta deben aumentar considerablemente pues, para la cuadrícula de 256×256 se pasa de 4 posibles ubicaciones de la ventana a 9, lo que corresponde a un incremento de 125 % y en la cuadrícula de 128×128 se pasa de 16 posibles ubicaciones de la ventana a 49 correspondiendo a un incremento del 206.25 %. La medición de tiempos de cómputo y la proposición de una solución (por ejemplo mediante la paralelización de las operaciones) se espera desarrollar como trabajo futuro.

Referencias bibliográficas

Aliyu Abubakar, Mohammed Ajuji, and Ibrahim Usman Yahya. Comparison of Deep Transfer Learning Techniques in Human Skin Burns Discrimination. *Applied System Innovation*, 3(2), 2020. ISSN 2571-5577. doi: 10.3390/asi3020020.

Ethem Alpaydin, Veronika Cheplygina, Marco Loog, and David M.J. Tax. Single- vs. multiple-instance classification. *Pattern Recognition*, 48(9):2831--2838, 2015. ISSN 00313203. doi: 10.1016/j.patcog.2015.04.006.

Jaume Amores. Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, 201:81--105, 2013. ISSN 0004-3702. doi: 10.1016/j.artint.2013.06.003.

Katharina Anding, Lilli Haar, Galina Polte, Jurij Walz, and Gunther Notni. Comparison of the performance of innovative deep learning and classical methods of machine learning to solve industrial recognition tasks. In *Proceedings Volume 11144, Photonics and Education in Measurement Science 2019*, page 26, 09 2019. doi: 10.1117/12.2530899.

Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support Vector Machines for Multiple-Instance Learning. *Advances in Neural Information Processing Systems*, 15:561--568, 01 2002.

Massimo Aria and Corrado Cuccurullo. bibliometrix: An R-tool for comprehensive science mapping analysis. *Journal of Informetrics*, 11(4):959--975, 2017. doi: 10.1016/j.joi.2017.08.007.

Boris Babenko. Multiple Instance Learning: Algorithms and Applications. In *Computer Science, Mathematics*, 01 2008. URL `https://api.semanticscholar.org/CorpusID:2153770`.

Saber Mirzaee Bafti, Chee Siang Ang, Md Moinul Hossain, Gianluca Marcelli, Marc Alemany-Fornes, and Anastasios D. Tsaousis. A crowdsourcing semi-automatic image segmentation platform for cell biology. *Computers in Biology and Medicine*, 130:104204, 2021. ISSN 0010-4825. doi: 10.1016/j.compbiomed.2020.104204.

Ganbayar Batchuluun, Jiho Choi, and Kang Ryoung Park. CAM-CAN: Class activation map-based categorical adversarial network. *Expert Systems with Applications*, 222:119809, 2023. ISSN 0957-4174. doi: 10.1016/j.eswa.2023.119809.

Laura Lucía Becerra Elejalde. Supercoco se renueva tras 70 años e incursiona con agua de coco en su portafolio. *La República*, June 16th, 2020. `https://tinyurl.com/43dsd9ad`.

Samia Benyahia, Boudjelal Meftah, and Olivier Lézoray. Multi-features extraction based on deep learning for skin lesion classification. *Tissue and Cell*, 74:101701, 2022. ISSN 0040-8166. doi: 10.1016/j.tice.2021.101701.

Jeff Bier. Is deep learning the solution to all computer vision problems? Blog post at the Vision Systems Design website, 2019. URL `https://tinyurl.com/bdf8yfw2`.

Tiago Botari, Rafael Izbicki, and Andre C. P. L. F. de Carvalho. Local Interpretation Methods to Machine Learning Using the Domain of the Feature Space. In Peggy Cellier and Kurt Driessens, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 241--252, Cham, 2020. Springer International Publishing. ISBN 978-3-030-43823-4.

Jakob Božič, Domen Tabernik, and Danijel Skočaj. End-to-end training of a two-stage neural network for defect detection. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 5619--5626, 2021. doi: 10.1109/icpr48806.2021.9412092.

Jakob Božič, Domen Tabernik, and Danijel Skočaj. Mixed supervision for surface-defect detection: From weakly to fully supervised learning. *Computers in Industry*, 129:103459, 08 2021. doi: 10.1016/j.compind.2021.103459.

Max Bramer. *Principles of Data Mining*. Springer, fourth edition edition, 2020. ISBN 978-1-4471-7493-6. doi: 10.1007/978-1-4471-7493-6.

Gilles Brassard and Paul Bratley. *Fundamentos de Algoritmia*. Prentice Hall, Montreal, primera edition, 1997. ISBN 84-89660-00-X.

Marc André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 77:329--353, 2017. ISSN 00313203. doi: 10.1016/j.patcog.2017.10.009.

Ssu-Han Chen and Der-Baau Perng. Directional textures auto-inspection using principal component analysis. *International Journal of Advanced Manufacturing Technology*, 55:1099--1110, 08 2011. doi: 10.1007/s00170-010-3141-1.

Wen-Chin Chen and Shou-Wen Hsu. A neural-network approach for an automatic LED inspection system. *Expert Systems with Applications*, 33:531--537, 08 2007. doi: 10.1016/j.eswa.2006.06.011.

Min-Yuan Cheng, Riqi Radian Khasani, and Kent Setiono. Image quality enhancement using HybridGAN for automated railway track defect recognition. *Automation in Construction*, 146:104669, 2023. ISSN 0926-5805. doi: 10.1016/j.autcon.2022.104669.

Veronika Cheplygina and David M. J. Tax. Characterizing Multiple Instance Datasets. In *Similarity-Based Pattern Recognition*, pages 15--27, Cham, 2015. Springer International Publishing. ISBN 978-3-319-24261-3. doi: 10.1007/978-3-319-24261-3_2.

Veronika Cheplygina, David M.J. Tax, and Marco Loog. Multiple instance learning with bag dissimilarities. *Pattern Recognition*, 48 (1):264--275, 2015. ISSN 0031-3203. doi: 10.1016/j.patcog.2014.07.022.

Veronika Cheplygina, David M. J. Tax, and Marco Loog. Dissimilarity-Based Ensembles for Multiple Instance Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 27(6):1379--1391, June 2016. ISSN 2162-237X. doi: 10.1109/TNNLS.2015.2424254.

François Chollet. *Deep Learning with Python*. Manning Publications, 2 edition, October 2021. ISBN 9781617296864. URL `https://www.manning.com/books/deep-learning-with-python-second-edition`.

Roger Chugh, Vardaan Bhatia, Karan Khanna, and Vandana Bhatia. A Comparative Analysis of Classifiers for Image Classification. In *2020 - 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, pages 248--253, 01 2020. doi: 10.1109/Confluence47617.2020.9058042.

Yandre Costa, Diego Bertolini, Alceu Jr, George Cavalcanti, and Luiz Soares de Oliveira. The dissimilarity approach: a review. *Artificial Intelligence Review*, 53:2783--2808, 04 2020. doi: 10.1007/s10462-019-09746-z.

Esteban Cumbajin, Nuno Rodrigues, Paulo Costa, Rolando Miragaia, Luís Frazão, Nuno Costa, Antonio Fernández-Caballero, Jorge Carneiro, Leire H. Buruberri, and António Pereira. A Systematic Review on Deep Learning with CNNs Applied to Surface Defect Detection. *Journal of Imaging*, 9(10), 2023. ISSN 2313-433X. doi: 10.3390/jimaging9100193.

Suresh Dara and Priyanka Tumma. Feature Extraction By Using Deep Learning: A Survey. In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pages 1795--1801, 2018. doi: 10.1109/ICECA.2018.8474912.

Shubhabrata Datta and J. Paulo Davim. *Machine Learning in Industry*. Management and Industrial Engineering. Springer, Cham, Switzerland, 2022. ISBN 978-3-030-75846-2. doi: 10.1007/978-3-030-75847-9.

Soham De, Anirbit Mukherjee, and Enayat Ullah. Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration. *arXiv*, 2018. doi: 10.48550/ARXIV.1807.06766.

Deltamax Automazione . Electronic References, 2016. URL `http://www.deltamaxautomazione.it/risolvi/`.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248--255, 2009. doi: 10.1109/CVPR.2009.5206848.

Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, 89:31--71, 1997. ISSN 0004-3702/97.

Peilei Dong and Wenmin Wang. Better region proposals for pedestrian detection with R-CNN. In *2016 Visual Communications and Image Processing (VCIP)*, pages 1--4, 2016. doi: 10.1109/VCIP.2016.7805452.

Robert P W Duin. Dissimilarity measures , Pattern Recognition Tools, 2012. URL `http://37steps.com/1264/dissimilarity-measures/`.

Robert P. W. Duin, Manuele Bicego, Mauricio Orozco-Alzate, Sang-Woon Kim, and Marco Loog. Metric Learning in Dissimilarity Space for Improved Nearest Neighbor Performance. In Pasi Fränti, Gavin Brown, Marco Loog, Francisco Escolano, and Marcello Pelillo, editors, *Structural, Syntactic, and Statistical Pattern Recognition*, pages 183--192, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. ISBN 978-3-662-44415-3. doi: 10.1007/978-3-662-44415-3_19.

Robert P.W. Duin and Elzbieta Pekalska. The dissimilarity representation for pattern recognition, a tutorial. `http://rduin.nl/presentations/DisRep_Tutorial.pdf`, 2011.

Ross Girshick. Fast R-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440--1448, 2015. doi: 10.1109/ICCV.2015.169.

Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580--587, 2014. doi: 10.1109/CVPR.2014.81.

Facundo E. Godoy. Métodos clásicos de clasificación: comparación y aplicación. Trabajo especial licenciatura en matemática, Facultad de Matemática, Astronomía, Física y Computación. Universidad Nacional de Córdoba, July 2021.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Luiz G. Hafemann, Luiz S. Oliveira, Paulo R. Cavalin, and Robert Sabourin. Transfer learning between texture classification tasks using Convolutional Neural Networks. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1--7, 2015. doi: 10.1109/IJCNN.2015.7280558.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision -- ECCV 2014*, pages 346--361, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10578-9.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980--2988, 2017. doi: 10.1109/ICCV.2017.322.

Francisco Herrera, Sebastián Ventura, Rafael Bello, Chris Cornelis, Amelia Zafra, Dánel Sánchez-Tarragó, and Sarah Vluymans. *Multiple instance learning: Foundations and algorithms*. Springer International Publishing, 2016. ISBN 9783319477596. doi: 10.1007/978-3-319-47759-6.

Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. CNN Architectures for Large-Scale Audio Classification, 2017.

O. Hmidani and E. M. Ismaili Alaoui. A comprehensive survey of the R-CNN family for object detection. In *2022 5th International Conference on Advanced Communication Technologies and Networking (CommNet)*, pages 1--6, 2022. doi: 10.1109/CommNet56067.2022.9993862.

Rudolf Hoffmann and Christoph Reich. A Systematic Literature Review on Artificial Intelligence and Explainable Artificial Intelligence for Visual Quality Assurance in Manufacturing. *Electronics*, 12(22), 2023. ISSN 2079-9292. doi: 10.3390/electronics12224572.

Shih-Chung Hsu, Chung-Lin Huang, and Cheng-Hung Chuang. Vehicle detection using simplified fast R-CNN. In *2018 International Workshop on Advanced Image Technology (IWAIT)*, pages 1--3, 2018. doi: 10.1109/IWAIT.2018.8369767.

Shiluo Huang, Zheng Liu, Wei Jin, and Ying Mu. Bag dissimilarity regularized multi-instance learning. *Pattern Recognition*, 126:108583, 2022. ISSN 0031-3203. doi: 10.1016/j.patcog.2022.108583.

Yibin Huang, Congying Qiu, Xiaonan Wang, Shijun Wang, and Kui Yuan. A Compact Convolutional Neural Network for Surface Defect Inspection. *Sensors*, 20(7), 2020. ISSN 1424-8220. doi: 10.3390/s20071974. URL `https://www.mdpi.com/1424-8220/20/7/1974`.

Dejice Jacob, Phil Trinder, and Jeremy Singer. Python programmers have GPUs too: automatic Python loop parallelization with staged dependence analysis. In Stefan Marr and Juan Fumero, editors, *Proceedings of the 15th ACM SIGPLAN International Symposium on Dynamic Languages, DLS 2019, Athens, Greece, October 20, 2019*, pages 42--54. ACM, 2019. ISBN 978-1-4503-6996-1. doi: 10.1145/3359619.3359743.

Minqi Jiang, Chaochuan Hou, Ao Zheng, Xiyang Hu, Songqiao Han, Hailiang Huang, Xiangnan He, Philip Yu, and Yue Zhao. Weakly Supervised Anomaly Detection: A Survey, 02 2023.

Manjunath Jogin, Mohana, M S Madhulika, G D Divya, R K Meghana, and S Apoorva. Feature Extraction using Convolution Neural Networks (CNN) and Deep Learning. In *2018 3rd IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology (RTEICT)*, pages 2319--2323, 2018. doi: 10.1109/RTEICT42901.2018.9012507.

M. Keshavarzi, Mohammad Ali Dehghan, and Mashaallah Mashinchi. Classification based on similarity and dissimilarity through equivalence classes. *Applied and Computational Mathematics*, 8:203--215, 01 2009.

Salman Khan, Hossein Rahmani, and Syed Afaq Ali Shah. *A Guide to Convolutional Neural Networks for Computer Vision*. Morgan & Claypool Publishers, 2018. ISBN 1681730219.

S. Kornblith, J. Shlens, and Q. V. Le. Do Better ImageNet Models Transfer Better? In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2656--2666, Los Alamitos, CA, USA, jun 2019. IEEE Computer Society. doi: 10.1109/CVPR.2019.00277.

Bangjun Lei, Guangzhu Xu, Ming Feng, Yaobin Zou, Ferdinand Van der Heijden, Dick de Ridder, and David M. J. Tax. *Classification, Parameter Estimation and State Estimation: An Engineering Approach Using MATLAB*. Wiley, 2017. doi: 10.1002/9781119152484.

Christian Leistner, Amir Saffari, and Horst Bischof. MIForests: Multiple-Instance Learning with Randomized Trees. In Kostas Daniilidis, Petros Maragos, and Nikos Paragios, editors, *Computer Vision -- ECCV 2010*, pages 29--42, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. ISBN 978-3-642-15567-3. doi: 10.1007/978-3-642-15567-3_3.

Li Li, Zuopai Zhou, Na Bai, Tao Wang, Kan-Hao Xue, Huajun Sun, Qiang He, Weiming Cheng, and Xiangshui Miao. Naive Bayes classifier based on memristor nonlinear conductance. *Microelectronics Journal*, 129:105574, 2022. ISSN 0026-2692. doi: 10.1016/j.mejo.2022.105574.

Pengcheng Li, Zihao Dong, Jianjie Shi, Zengzhi Pang, and Jinping Li. Detection of Small Size Defects in Belt Layer of Radial Tire Based on Improved Faster R-CNN. In *2021 11th International Conference on Information Science and Technology (ICIST)*, pages 531--538, 2021a. doi: 10.1109/ICIST52614.2021.9440580.

Shengyuan Li and Xuefeng Zhao. Image-Based Concrete Crack Detection Using Convolutional Neural Network and Exhaustive Search Technique. *Advances in Civil Engineering*, 2019:1--12, 04 2019. doi: 10.1155/2019/6520620.

Xuhong Li, Haoyi Xiong, Xingjian Li, Xuanyu Wu, Xiao Zhang, Ji Liu, Jiang Bian, and Dejing Dou. Interpretable Deep Learning: Interpretation, Interpretability, Trustworthiness, and Beyond, 2021b.

Yu Li and ZiWei Wang. Research on Textile Defect Detection Based on Improved Cascade R-CNN. In *2021 International Conference on Artificial Intelligence and Electromechanical Automation (AIEA)*, pages 43--46, 2021. doi: 10.1109/AIEA53260.2021.00017.

Yu Liang, Siguang Li, Chungang Yan, Maozhen Li, and Changjun Jiang. Explaining the black-box model: A survey of local interpretation methods for deep neural networks. *Neurocomputing*, 419:168--182, 2021. ISSN 0925-2312. doi: 10.1016/j.neucom.2020.08.011.

Hong-Dar Lin and Duan-Cheng Ho. Detection of pinhole defects on chips and wafers using DCT enhancement in computer vision systems. *The International Journal of Advanced Manufacturing Technology*, 34:567--583, 09 2007. doi: 10.1007/s00170-006-0614-3.

Bo Liu, Zhi-Feng Hao, and Xiao-Wei Yang. Nesting support vector machinte for muti-classification [machinte read machine]. In *2005 International Conference on Machine Learning and Cybernetics*, volume 7, pages 4220--4225 Vol. 7, 2005. doi: 10.1109/ICMLC.2005.1527678.

Caihui Liu, Bowen Lin, Jianying Lai, and Duoqian Miao. An improved decision tree algorithm based on variable precision neighborhood similarity. *Information Sciences*, 615:152--166, 2022a. ISSN 0020-0255. doi: 10.1016/j.ins.2022.10.043.

Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single Shot MultiBox Detector. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision — ECCV 2016*, pages 21--37, Cham, 2016. Springer International Publishing. doi: 10.1007/978-3-319-46448-0_2.

Wei Liu, Jiarui Zhang, and Yijun Zhao. A Comparison of Deep Learning and Traditional Machine Learning Approaches in Detecting Cognitive Impairment Using MRI Scans. In *2022 IEEE 46th Annual Computers, Software, and Applications Conference (COMPSAC)*, pages 998--1001, 2022b. doi: 10.1109/COMPSAC54236.2022.00154.

Tanya Makkar, Yogesh Kumar, Ashwani Kr Dubey, Álvaro Rocha, and Ayush Goyal. Analogizing time complexity of KNN and CNN in recognizing handwritten digits. In *2017 Fourth International Conference on Image Information Processing (ICIIP)*, pages 1--6, 2017. doi: 10.1109/ICIIP.2017.8313707.

Elias N. Malamas, Euripides G M Petrakis, Michalis Zervakis, Laurent Petit, and Jean Didier Legat. A survey on industrial vision systems, applications and tools. *Image and Vision Computing*, 21(2):171--188, 2003. ISSN 0262-8856/03.

Borja Marin, Keith Brown, and Mustafa S. Erden. Automated Masonry crack detection with Faster R-CNN. In *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, pages 333--340, 2021. doi: 10.1109/CASE49439.2021.9551683.

Oded Maron and Aparna Lakshmi Ratan. Multiple-Instance Learning for Natural Scene Classification. In *Proceedings of the Fifteenth International Conference on Machine Learning*, ICML '98, page 341–349, San Francisco, CA, USA, 1998. Morgan Kaufmann Publishers Inc. ISBN 1558605568. URL `https://dl.acm.org/doi/10.5555/645527.657299`.

Carlos Mera. *Detección de defectos en sistemas de inspección visual automática a través del aprendizaje de múltiples instancias*. Tesis doctoral, Universidad Nacional de Colombia - Sede Medellín, Medellín, Colombia, may 2017. URL `https://repositorio.unal.edu.co/handle/unal/59346`. Doctorate in Engineering - Systems and Informatics.

Carlos Mera, Mauricio Orozco-Alzate, John Branch, and Domingo Mery. Automatic visual inspection: An approach with multi-instance learning. *Computers in Industry*, 83:46--54, 2016. doi: 10.1016/j.compind.2016.09.002.

Carlos Mera, Mauricio Orozco-Alzate, and John Branch. Incremental learning of concept drift in Multiple Instance Learning for industrial visual inspection. *Computers in Industry*, 109:153--164, aug 2019. doi: 10.1016/j.compind.2019.04.006.

Domingo Mery. BALU: A Matlab toolbox for computer vision, pattern recognition and image processing, 2011. URL `http://dmery.ing.puc.cl/index.php/balu/`.

Tim Miller. Explanation in artificial intelligence: Insights from the social sciences. *Artificial Intelligence*, 267:1--38, 2019. ISSN 0004-3702. doi: 10.1016/j.artint.2018.07.007.

H. B. Mitchell. *Image Similarity Measures*, pages 167--185. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010. ISBN 978-3-642-11216-4. doi: 10.1007/978-3-642-11216-4_14.

Elhassan Mohamed, Konstantinos Sirlantzis, and Gareth Howells. A review of visualisation-as-explanation techniques for convolutional neural networks and their evaluation. *Displays*, 73:102239, 2022. ISSN 0141-9382. doi: 10.1016/j.displa.2022.102239.

Divya Pramasani Mohandoss, Yong Shi, and Kun Suo. Outlier Prediction Using Random Forest Classifier. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0027--0033, 2021. doi: 10.1109/CCWC51732.2021.9376077.

Christoph Molnar. *Interpretable Machine Learning*. Independently published, 2019. URL `https://christophm.github.io/interpretable-ml-book/`.

Romain Mormont, Pierre Geurts, and Raphaël Marée. Comparison of Deep Transfer Learning Strategies for Digital Pathology. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2343--2352, 2018. doi: 10.1109/CVPRW.2018.00303.

Saad Naeem, Noreen Jamil, Habib Ullah Khan, and Shah Nazir. Complexity of Deep Convolutional Neural Networks in Mobile Computing. *Complexity*, 2020:3853780, Sep 2020. ISSN 1076-2787. doi: 10.1155/2020/3853780.

Tam Nguyen and Raviv Raich. Incomplete Label Multiple Instance Multiple Label Learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1320--1337, 2022. doi: 10.1109/tpami.2020.3017456.

Mark Nixon and Alberto Aguado. *Low-level feature extraction (including edge detection)*, pages 137--216. Photonics and Education in Measurement Science, 12 2012. ISBN 9780123965493. doi: 10.1016/B978-0-12-396549-3.00004-5.

C. Guadalupe Origel-Rivas, Eréndira Rendón Lara, Itzel Abundez Barrera, and Roberto Alejo Eleuterio. Redes neuronales artificiales y árboles de decisión para la clasificación con datos categóricos. *Res. Comput. Sci.*, 149(8):541--554, 2020.

Taiwo R. Oyedare, Vijay Shah, Daniel Jakubisin, and Jeffrey Reed. Keep It Simple: CNN Model Complexity Studies for Interference Classification Tasks. *arXiv - EE - Signal Processing*, 03 2023. doi: 10.48550/arXiv.2303.03326.

Christian Payer, Darko Štern, Horst Bischof, and Martin Urschler. Integrating spatial configuration into heatmap regression based CNNs for landmark localization. *Medical Image Analysis*, 54:207--219, 2019. ISSN 1361-8415. doi: 10.1016/j.media.2019.03.007.

Elzbieta Pekalska and Robert P. W. Duin. *The dissimilarity representation for pattern recognition , Foundations and Applications*. World Scientific, Delft, 2005. ISBN 9812565302. doi: 10.1142/9789812703170.

Jacek Piotrowski. Top-Down Approach to Image Similarity Measures. In Leonard Bolc, Juliusz L. Kulikowski, and Konrad Wojciechowski, editors, *Computer Vision and Graphics*, pages 66--69, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg. doi: 10.1007/978-3-642-02345-3_7.

Leonardo Plotegher, Chiara Corridori, Fabio Dolci, Claudio Andreatta, Silvio Benini, and Matteo Devilli. The GI dataset for glass inspection 1st release (August 2016), 2016. URL `http://www.deltamaxautomazione.it/risolvi`.

Diana Porro-Munoz, Robert P W Duin, Isneri Talavera, and Mauricio Orozco-Alzate. Classification of three-way data by the dissimilarity representation. *Signal Processing*, 91(11):2520--2529, 2011. ISSN 01651684. doi: 10.1016/j.sigpro.2011.05.004.

Konpat Preechakul, Sira Sriswasdi, Boonserm Kijsirikul, and Ekapol Chuangsuwanich. Improved image classification explainability with high-accuracy heatmaps. *iScience*, 25(3):103933, 2022. ISSN 2589-0042. doi: 10.1016/j.isci.2022.103933.

Q.R. Razlighi, N. Kehtarnavaz, and S. Yousefi. Evaluating similarity measures for brain image registration. *Journal of Visual Communication and Image Representation*, 24(7):977--987, 2013. ISSN 1047-3203. doi: 10.1016/j.jvcir.2013.06.010.

Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779--788, 2016. doi: 10.1109/CVPR.2016.91.

Payam Refaeilzadeh, Lei Tang, and Huan Liu. *Cross-Validation*, pages 532--538. Springer US, Boston, MA, 2009. ISBN 978-0-387-39940-9. doi: 10.1007/978-0-387-39940-9_565.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39(06):1137--1149, jun 2017. ISSN 1939-3539. doi: 10.1109/TPAMI.2016.2577031.

Amal Saadallah, Jan Buescher, Omar Abdulaaty, Thorben Panusch, Jochen Deuse, and Katharina Morik. Explainable Predictive Quality Inspection using Deep Learning in Electronics Manufacturing. *Procedia CIRP*, 107:594--599, 01 2022. doi: 10.1016/j.procir.2022.05.031.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*, 128(2): 336--359, Feb 2020. ISSN 1573-1405. doi: 10.1007/s11263-019-01228-7.

Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, and Yann LeCun. OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. *International Conference on Learning Representations (ICLR) Banff*, 2013. doi: 10.48550/ARXIV.1312.6229.

Bhoomi Shah and Hetal Bhavsar. Time Complexity in Deep Learning Models. *Procedia Computer Science*, 215:202--210, 2022. ISSN 1877-0509. doi: 10.1016/j.procs.2022.12.023. 4th International Conference on Innovative Data Communication Technology and Application.

Feifei Shao, Long Chen, Jian Shao, Wei Ji, Shaoning Xiao, Lu Ye, Yueting Zhuang, and Jun Xiao. Deep Learning for Weakly-Supervised Object Detection and Localization: A Survey. *Neurocomputing*, 496:192--207, 2022. ISSN 0925-2312. doi: 10.1016/j.neucom.2022.01.095.

Wei Shen, Zelin Peng, Xuehui Wang, Huayu Wang, Jiazhong Cen, Dongsheng Jiang, Lingxi Xie, Xiaokang Yang, and Q. Tian. A Survey on Label-Efficient Deep Image Segmentation: Bridging the Gap Between Weak Supervision and Dense Prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1--20, 2023. doi: 10.1109/TPAMI.2023.3246102.

Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*, 6(60): 1--48, 07 2019. ISSN 2196-1115. doi: 10.1186/s40537-019-0197-0.

Javier Silvestre-Blanes, Teresa Albero-Albero, Ignacio Miralles, Rubén Pérez-Llorens, and Jorge Moreno. A Public Fabric Database for Defect Detection Methods and Results. *Autex Research Journal*, 19(4):363--374, 06 2019. doi: 10.2478/aut-2019-0035.

C. Smith. *Decision Trees and Random Forests: A Visual Introduction for Beginners*. Blue Windmill Media, 2017. ISBN 9781549893759. URL `https://books.google.com.co/books?id=Hi_CtAEACAAJ`.

Melvyn L. Smith, Lyndon N. Smith, and Mark F. Hansen. The quiet revolution in machine vision - a state-of-the-art survey paper, including historical review, perspectives, and future directions. *Computers in Industry*, 130:103472, 2021. doi: 10.1016/j.compind.2021.103472.

Huazhu Song, Zichun Ding, Cuicui Guo, Zhe Li, and Hongxia Xia. Research on Combination Kernel Function of Support Vector Machine. In *2008 International Conference on Computer Science and Software Engineering*, volume 1, pages 838--841, 2008. doi: 10.1109/CSSE.2008.1231.

Mohsen Soori, Behrooz Arezoo, and Roza Dastres. Machine learning and artificial intelligence in CNC machine tools, A review. *Sustainable Manufacturing and Service Economics*, page 100009, 2023. ISSN 2667-3444. doi: 10.1016/j.smse.2023.100009.

Miao Sun, Tony X. Han, Ming-Chang Liu, and Ahmad Khodayari-Rostamabad. Multiple instance learning convolutional neural networks for object recognition. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 3270--3275, 2016. doi: 10.1109/ICPR.2016.7900139.

Zhiyuan Sun, Yunhao Yuan, Xiaoxiao Dong, Zhimei Liu, Kelong Cai, Wei Cheng, Jingjing Wu, Zhiyuan Qiao, and Aiguo Chen. Supervised machine learning: A new method to predict the outcomes following exercise intervention in children with autism spectrum disorder. *International Journal of Clinical and Health Psychology*, 23(4):100409, 2023. ISSN 1697-2600. doi: 10.1016/j.ijchp.2023.100409.

Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander Alemi. Inception-v4, Inception-ResNet and the impact of residual connections on learning. *AAAI Conference on Artificial Intelligence*, 31, 02 2016. doi: 10.1609/aaai.v31i1.11231.

Domen Tabernik, Samo Šela, Jure Skvarč, and Danijel Skočaj. Segmentation-Based Deep-Learning Approach for Surface-Defect Detection. *Journal of Intelligent Manufacturing*, May 2019. ISSN 1572-8145. doi: 10.1007/s10845-019-01476-x.

Tomoumi Takase, Ryo Karakida, and Hideki Asoh. Self-paced Data Augmentation for Training Neural Networks, 2020.

Michael W. Trosset, Carey E. Priebe, Youngser Park, and Michael I. Miller. Semisupervised learning from dissimilarity data. *Computational Statistics & Data Analysis*, 52(10):4643--4657, 2008. ISSN 0167-9473. doi: 10.1016/j.csda.2008.02.030.

Marc van Kreveld, Tillmann Miltzow, Tim Ophelders, Willem Sonke, and Jordi L. Vermeulen. Between shapes, using the Hausdorff distance. *Computational Geometry*, 100, 2022. ISSN 0925-7721. doi: 10.1016/j.comgeo.2021.101817.

Srikanth Vemula and Michael Frye. Mask R-CNN powerline detector: A deep learning approach with applications to a UAV. In *2020 AIAA/IEEE 39th Digital Avionics Systems Conference (DASC)*, pages 1--6, 2020. doi: 10.1109/DASC50938.2020.9256456.

R. Venkatesan and Li. Baoxin. *Convolutional neural networks in visual computing : a concise guide*. CRC press, first edition edition, 2017. ISBN 9781315154282 (ebook). URL `https://lccn.loc.gov/2017029154`.

Eduardo Villegas-Jaramillo. Github-candies, 2023. URL `https://github.com/ejvillegasj/candies`.

Eduardo Villegas-Jaramillo and Mauricio Orozco-Alzate. Computational Analysis of Multiple Instance Learning-Based Systems for Automatic Visual Inspection: A Doctoral Research Proposal. In Sara Rodríguez, Javier Prieto, Pedro Faria, Sławomir Kłos, Alberto Fernández, Santiago Mazuelas, M. Dolores Jiménez-López, María N. Moreno, and Elena M. Navarro, editors, *Distributed Computing and Artificial Intelligence, Special Sessions, 15th International Conference*, pages 374--377, Cham, 2019. Springer International Publishing. ISBN 978-3-319-99608-0. doi: 10.1007/978-3-319-99608-0_49.

Eduardo Villegas-Jaramillo and Mauricio Orozco-Alzate. *Convolutional Neural Networks and Deep Learning Techniques for Glass Surface Defect Inspection*, pages 67--99. IGI Global, 701 E. Chocolate Avenue Hershey PA, USA 17033, 2022. doi: 10.4018/978-1-6684-4991-2.ch004.

Eduardo Villegas-Jaramillo and Mauricio Orozco-Alzate. Candy classification using convolutional neural networks, data augmentation and transfer learning: Application and a new public dataset. In *2023 IEEE 13th International Conference on Pattern Recognition Systems (ICPRS)*, pages 1--7, 2023a. doi: 10.1109/ICPRS58416.2023.10179072.

Eduardo Villegas-Jaramillo and Mauricio Orozco-Alzate. An Inexactly Supervised Methodology Based on Multiple Instance Learning, Convolutional Neural Networks and Dissimilarities for Interpretable Defect Detection and Localization on Textured Surfaces. *IEEE Access*, 11:138229--138246, 2023b. doi: 10.1109/ACCESS.2023.3340047.

Eduardo Villegas-Jaramillo, Ana Lorena Uribe-Hurtado, and Mauricio Orozco-Alzate. Multi-core Parallelization of Point Set Dissimilarities for Accelerating the Comparison of Bags with Many Instances. In Sigeru Omatu, Rashid Mehmood, Pawel Sitek, Serafino Cicerone, and Sara Rodríguez, editors, *Distributed Computing and Artificial Intelligence, 19th International Conference*, pages 208--218, Cham, 2023. Springer International Publishing. ISBN 978-3-031-20859-1.

A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis. Deep Learning for Computer Vision: A Brief Review. *Computational Intelligence and Neuroscience*, 2018:13, 2018. ISSN 1687-5265. doi: 10.1155/2018/7068349.

Hua Wang, Cuiqin Ma, and Lijuan Zhou. A Brief Review of Machine Learning and Its Application. In *2009 International Conference on Information Engineering and Computer Science*, pages 1--4, 2009. doi: 10.1109/ICIECS.2009.5362936.

Juan Wang and Bin Xia. Weakly supervised image segmentation beyond tight bounding box annotations. *Computers in Biology and Medicine*, 169:107913, 2024. ISSN 0010-4825. doi: 10.1016/j.compbiomed.2023.107913. URL `https://www.sciencedirect.com/science/article/pii/S0010482523013781`.

Shenlong Wang, Kaixin Han, and Jiafeng Jin. Review of image low-level feature extraction methods for content-based image retrieval. *Sensor Review*, ahead-of-print, 08 2019a. doi: 10.1108/SR-04-2019-0092.

Wei Wang, Linyang He, Guohua Cheng, Ting Wen, and Yan Tian. Learning from ambiguous labels for X-Ray security inspection via weakly supervised correction. *Multimedia Tools and Applications*, 83:1--16, 05 2023. doi: 10.1007/s11042-023-15299-9.

Xinggang Wang, Yongluan Yan, Peng Tang, Wenyu Liu, and Xiaojie Guo. Bag similarity network for deep multi-instance learning. *Information Sciences*, 504:578--588, 2019b. ISSN 0020-0255. doi: 10.1016/j.ins.2019.07.071.

Xin Wen, Jvran Shan, Yu He, and Kechen Song. Steel Surface Defect Recognition: A Survey. *Coatings*, 13(1), 2023. ISSN 2079-6412. doi: 10.3390/coatings13010017.

Xinquan Wu and Xuefeng Yan. A spatial pyramid pooling-based deep reinforcement learning model for dynamic job-shop scheduling problem. *Computers & Operations Research*, 160:106401, 2023. ISSN 0305-0548. doi: 10.1016/j.cor.2023.106401.

Ting Xiao, Lei Liu, Kai Li, Wenjian Qin, Nicolas Yu, and Zhi-Cheng Li. Comparison of Transferred Deep Neural Networks in Ultrasonic Breast Masses Discrimination. *BioMed Research International*, 2018:1--9, 06 2018. doi: 10.1155/2018/4605191.

Xin Xu and Eibe Frank. Logistic regression and boosting for labeled bags of instances. In Honghua Dai, Ramakrishnan Srikant, and Chengqi Zhang, editors, *Advances in Knowledge Discovery and Data Mining*, pages 272--281, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-24775-3. doi: 10.1007/978-3-540-24775-3_35.

Yan Xu, Yeshu Li, Zhengyang Shen, Ziwei Wu, Teng Gao, Yubo Fan, Maode Lai, and Eric Chang. Parallel multiple instance learning for extremely large histopathology image analysis. *BMC Bioinformatics*, 18, 08 2017. doi: 10.1186/s12859-017-1768-8.

Mingqiang Yang, Kidiyo Kpalma, and Joseph Ronsin. A Survey of Shape Feature Extraction Techniques. *IN-TECH*, 15, 11 2007. doi: 10.5772/6237.

Chia-Yu Yen and Krzysztof J. Cios. Image recognition system based on novel measures of image similarity and cluster validity. *Neurocomputing*, 72(1):401--412, 2008. ISSN 0925-2312. doi: 10.1016/j.neucom.2007.12.018. Machine Learning for Signal Processing (MLSP 2006) / Life System Modelling, Simulation, and Bio-inspired Computing (LSMS 2007).

Jun Yue, Leyuan Fang, Pedram Ghamisi, Weiying Xie, Jun Li, Jocelyn Chanussot, and Antonio Plaza. Optical Remote Sensing Image Understanding With Weak Supervision: Concepts, methods, and perspectives. *IEEE Geoscience and Remote Sensing Magazine*, 10 (2):250--269, 2022. doi: 10.1109/MGRS.2022.3161377.

Amelia Zafra and Eva Gibaja. Nearest neighbor-based approaches for multi-instance multi-label classification. *Expert Systems with Applications*, 232:120876, 2023. ISSN 0957-4174. doi: 10.1016/j.eswa.2023.120876.

Dingwen Zhang, Junwei Han, Gong Cheng, and Ming-Hsuan Yang. Weakly Supervised Object Localization and Detection: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5866--5885, 2022. doi: 10.1109/TPAMI.2021.3074313.

W. J. Zhang, D. Li, F. Ye, and H. Sun. Automatic optical defect inspection and dimension measurement of drill bit. In *2006 International Conference on Mechatronics and Automation*, pages 95--100, 2006. doi: 10.1109/ICMA.2006.257459.

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning Deep Features for Discriminative Localization. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921--2929, 2016. doi: 10.1109/CVPR.2016.319.

Lei Zhou, Yu Zhao, Jie Yang, Qi Yu, and Xun Xu. Deep multiple instance learning for automatic detection of diabetic retinopathy in retinal images. *IET Image Processing*, 12(4):563--571, 2018. ISSN 1751-9659. doi: 10.1049/iet-ipr.2017.0636.

Tongxue Zhou, Su Ruan, and Stéphane Canu. A review: Deep learning for medical image segmentation using multi-modality fusion. *Array*, 3-4:100004, sep 2019. ISSN 2590-0056. doi: 10.1016/j.array.2019.100004.

Zhi-Hua Zhou. A Brief Introduction to Weakly Supervised Learning. *National Science Review*, 1(August 2017):44--53, 2017. ISSN 2095-5138. doi: 10.1093/nsr/nwx106.

Keqian Zhu and Jingfei Jiang. Research on Parallel Acceleration for Deep Learning Inference Based on Many-Core ARM Platform. In Chao Li and Junjie Wu, editors, *Advanced Computer Architecture*, pages 30--41, Singapore, 2018. Springer Singapore. ISBN 978-981-13-2423-9. doi: 10.1007/978-981-13-2423-9_3.

Justus Zipfel, Felix Verworner, Marco Fischer, Uwe Wieland, Mathias Kraus, and Patrick Zschech. Anomaly detection for industrial quality assurance: A comparative evaluation of unsupervised deep learning models. *Computers & Industrial Engineering*, 177: 109045, 2023. ISSN 0360-8352. doi: 10.1016/j.cie.2023.109045.

Łukasz Struski, Szymon Janusz, Jacek Tabor, Michał Markiewicz, and Arkadiusz Lewicki. Multiple instance learning for medical image classification based on instance importance. *Biomedical Signal Processing and Control*, 91:105874, 2024. ISSN 1746-8094. doi: 10.1016/j.bspc.2023.105874. URL `https://www.sciencedirect.com/science/article/pii/S1746809423013071`.