Maritime Transport Damage Classification with From-Scratch and Pretrained Neural Networks

Hugo Daniel Flores [1], Carlos Neil [2] and Claudio Delrieux [3]

[1,2] Centro de Altos Estudios en Tecnología Informática (CAETI), Facultad de Tecnología Informática, Universidad Abierta Interamericana. Buenos Aires. Argentina.
[3] Universidad Nacional del Sur. Departamento de Ingeniería Eléctrica y Computadoras. Bahía Blanca. Argentina.
hugodaniel.flores@alumnos.uai.edu.ar, Carlos.Neil@uai.edu.ar, cad@uns.edu.ar

Abstract. This article approaches a problem that has arisen as a consequence of the high percentage of vehicles that are transported by car carrier vessel. To control its state and detect damages is did by specialized staff, it causes difficulties to determine imputation of liabilities. The Computer Vision (CV) is a viable alternative for detecting and classificating damages. The target this job is to implement Artificial Intelligence (AI) algorithms to classify damages in the maritime industry. For this, it's described like programming Convolutional Neural Network (CNN) and then as using pretrained networks to improving the results. The results show a deep analysis comparing the accuracy between neural networks implemented from scratch and pretrained. It's concluded that CV with CNN pretrained improving the results to classify damages in this industry. Note: this article extends the previous research carried out in JAIIO54, now focusing on better results by implementing pretrained networks.

Keywords: Vehicle Damages, Convolutional Neural Networks, Computer Vision, Pretrained Networks

Clasificación de daños en transporte marítimo implementando redes neuronales desde cero y preentrenadas

Resumen. Este trabajo aborda un problema surgido como consecuencia del alto porcentaje de vehículos que son transportados vía marítima. Controlar su estado y detectar daños recae en personal especializado, esto genera dificultades en la imputación de responsabilidades. La Visión Artificial (VA) es una alternativa viable para detectar y clasificar daños. El objetivo de este artículo es analizar e implementar la utilización de algoritmos de Inteligencia Artificial (IA) para clasificar daños en la industria marítima. Para ello se describe la programación de modelos de Redes Neuronales Convolucionales (RNC) y luego el uso de redes preentrenadas para mejorar los resultados obtenidos con los modelos programados desde cero. Los resultados muestran un análisis profundo comparando la exactitud de los modelos programados y preentrenados. Se concluye que las técnicas de VA con RNC preentrenadas mejoran las soluciones para clasificar daños en la industria del transporte marítimo de autos. Nota: Este artículo extiende la investigación previa realizada y publicada en JAIIO54, enfocándose ahora en mejores resultados implementando redes preentrenadas.

Palabras clave: Daños Vehículos, Redes Neuronales Convolucionales, Visión Artificial, Redes Preentrenadas

1. Introducción

1.1. Antecedentes

Este artículo extiende la investigación previa realizada en (Flores & Neil, 2025), enfocándose ahora en mejores resultados implementando redes preentrenadas.

La producción de daños en vehículos es un tema recurrente en la industria marítima y en particular en la industria automotriz. La descripción de los antecedentes es especificada en (Flores et al., 2026). Para señalar, la información de los reportes por daños producidos incluye imágenes para la certificación correspondiente. Los problemas se presentan en los cambios de responsabilidad de una empresa a otra. La importancia del tema en cuestión radica en que la evaluación actual de daños depende de la visión humana y del criterio y experiencia del perito.

1.2. Procesos

La utilidad de la investigación propuesta fue desarrollada en (Flores & Neil, 2023). Este artículo propone mejorar los resultados de un clasificador de imágenes creado mediante técnicas de aprendizaje supervisado con RNC usando redes preentrenadas.

La presentación de este artículo tiene como objetivo mostrar como los resultados de la utilización de redes neuronales preentrenadas mejora los resultados obtenidos utilizando redes neuronales desarrolladas desde cero. Cabe destacar que la base de datos utilizada para este artículo es la misma con la que se obtuvieron muy buenos resultados expuestos previamente (Flores & Neil, 2025).

2. Materiales y Métodos

2.1. Redes Neuronales Convolucionales

Los fundamentos teóricos sobre los materiales y métodos utilizados en este artículo relacionados con la estructura y la configuración de las RNC fueron descriptos en investigación previa en (Flores & Neil, 2025), que sirve como base conceptual y experimental del presente estudio. En este artículo se entrenan y se muestran los resultados de 4 modelos de RNC para los 3 subproyectos propuestos para comparar y mejorar la clasificación de daños: tipos de daños, partes dañadas y lugares donde se producen.

2.2.1 Programación de red 6 capas

El modelo de red inicial con el que se obtuvieron buenos resultados entrenando desde cero consta de una secuencia de bloques compuestos donde se aplicaron 6 capas convolucionales con kernel (nucleo) de 3 x 3, stride (tamaño de paso) 1 y padding (cantidad de pixeles que se rellenan hacia el exterior para obtener el borde) 1. Por cada capa convolucional se aplicó batch normalization con parámetros eps (1e-05, valor pequeño

para mantener la estabilidad del modelo, y evitar la división por cero), momentum (0.1, valor para evitar oscilaciones, y acelerar y mantener el modelo estable), affine (true, para no perder capacidad de representación), y track running stats (true, para guardar estadísticas globales de media y varianza acumuladas para usarlas en inferencia) por default. Cada 2 capas convolucionales se aplicaron max poooling de 2 x 2 con parámetros padding (0), dilatación (1), y ceil mode (false, no se incluyen los bordes sobrantes en un paso adicional) por default. Y finalmente se alineó el modelo con la función flatten (aplanamiento en un vector unidimensional), la que va conectada a una capa lineal y al final el output de la red con la clasificación de las categorías correspondientes (Flores & Neil, 2025).

Se empleó un learning rate de 1e-4, tamaño de lote de 64 muestras y 64 épocas de entrenamiento. Para la optimización se ha utilizado la función Adam y se incorporó regularización dropout de 0,25 para reducir el sobreajuste.

2.2.2 Adaptación para aprendizaje ResNet101

La red residual es un modelo de aprendizaje profundo utilizado en aplicaciones de visión artificial. La arquitectura ResNet (red neuronal residual) fue presentada en (He et al., 2015). Uno de los avances más significativos en el reconocimiento visual a gran escala fue la introducción de la red residual. ResNet logra resultados sin precedentes al abordar los desafíos asociados con el entrenamiento de redes neuronales muy profundas. A medida que aumenta el número de capas en la red neuronal, los gradientes de la función de pérdida con respecto al peso pueden volverse extremadamente pequeños durante la retropropogación. Esto dificulta que las capas posteriores aprendan patrones significativos, ya que las actualizaciones de los pesos tienen un impacto prácticamente nulo. Por el contrario, en algunos casos, los gradientes pueden volverse muy grandes durante la retropropogación, lo que genera inestabilidad en el proceso de entrenamiento al provocar una gran actualización de los pesos, dificultando así el control del proceso de optimización.

Por consiguiente, el problema de degradación en redes neuronales se refiere al fenómeno en el que, a medida que aumenta la profundidad de una red neuronal, su rendimiento en los datos de entrenamiento se satura y comienza a degradarse. Este efecto es particularmente problemático porque contradice la idea de que las redes más profundas son más capaces de extraer características complejas y abstractas. Se podría considerar como un doble problema. En la medida que aumenta el número de capas en la red neuronal, el error de entrenamiento tiende a saturarse (se crea una meseta) y deja de mejorar. Esto significa que las capas adicionales no tienen ningún beneficio significativo en la reducción del error de entrenamiento. Por otra parte, después de agregar más capas, sorprendentemente el error en el conjunto de validación comienza a aumentar y el rendimiento en los datos no vistos se vuelve deficiente (se degrada la precisión).

El preprocesamiento de imágenes de la base de datos para implementar ResNet101 utiliza una transformación en formato ImageNet de 224 x 224 píxeles y se utilizan los valores estándares de normalización (*ImageNet*, 2025). Para poder usar los datos de la red preentrenada se cambiaron los head (el encabezado) de la red ResNet101 por la última capa correspondientes a la clasificación de daños, partes dañadas o lugares

donde se producen los daños. Toda la abstracción que aprendió el modelo a través de ImageNet se transfiere al modelo de este artículo. Luego se aplana la red transformándola en un vector lineal y la última capa se configura con la clasificación correspondiente a cada uno de los subproyectos (5 o 4 categorías). Finalmente se implementa un optimizador como en los modelos programados desde cero.

Para implementar este modelo preentrenado se utiliza la tecnología Torchvision models (PyTorch, 2025b). Una vez cargado el modelo que es útil para este artículo, se procede al congelamiento del cuerpo de este, y solo se entrenan las capas agregadas. Puesto que entrenar el sistema completo es complejo y no tiene sentido hacerlo porque se puede adaptar la red y solo entrenar la parte interesante para este proyecto en sus diferentes versiones, esto es, un modelo para detección de daños, otro para detección de partes con daños y otro para clasificación de lugares donde se producen los daños. El sistema funciona como un extractor de features, o sea que va a extraer las características más importantes y las implementará en la última capa que será la que estará aprendiendo y la cual ajustará los parámetros conforme al requerimiento de cada proyecto. El congelamiento del cuerpo del modelo se realiza configurando los parámetros que indican si el modelo es preentrenable o no (pretrained=true).

Se ha configurado un learning rate de 5e-4, tamaño de lote de 64 muestras y 100 épocas de entrenamiento. La optimización se desarrolló mediante Adam. Se utiliza batch normalization que es integrada para regularizar su entrenamiento.

2.2.3. Adaptación para aprendizaje EfficientNet-B0

Cuando se desarrollan redes neuronales convolucionales, se hace con un costo fijo de recursos. Estas redes se escalan posteriormente para lograr mejores precisiones cuando hay más recursos disponibles. Un modelo ResNet18 por ejemplo puede escalarse a un modelo ResNet200 añadiendo más capas al modelo original. En la mayoría de los casos, esta técnica de escalado ha ayudado a proporcionar mejores precisiones en la mayoría de los conjuntos de datos de referencia. Sin embargo, las técnicas convencionales de escalado de modelos son muy aleatorias. Algunos modelos se escalan en profundidad y otros en anchura. Algunos modelos simplemente toman imágenes de mayor resolución para obtener mejores resultados. Esta técnica de escalado aleatorio de modelos requiere ajuste manual y muchas horas operativas de recursos de personal, lo que a menudo resulta en poca o ninguna mejora en el rendimiento. Los autores de EfficientNet propusieron escalar los modelos de RNC para obtener mejor precisión y eficiencia de una manera mucho más íntegra (Tan & Le, 2019).

EfficientNet utiliza una técnica llamada coeficiente compuesto para escalar los modelos de forma sencilla pero eficaz. En lugar de aumentar aleatoriamente el ancho, la profundidad o la resolución, el escalado compuesto escala uniformemente cada dimensión con un conjunto fijo de coeficientes de escalado. Utilizando este método de escalado y AutoML (automatización de múltiples funciones de aprendizaje automático), los autores de EfficientNet desarrollaron siete modelos de diversas dimensiones que superaron la precisión de vanguardia de la mayoría de las redes neuronales convolucionales y con una eficiencia mucho mayor.

El preprocesamiento de imágenes de la base de datos para implementar EfficientNet-B0 es conceptualmente el mismo que el del modelo ResNet101 e inclusive se utiliza la transformación con los valores estándares del formato de ImageNet 224 x 224 pixeles. También se cambian los encabezados por la última capa de los tres subproyectos de este trabajo, y así toda la abstracción que aprendió el modelo a través de ImageNet se transfiere a los modelos de EfficientNet-B0. Finalmente se aplana la red usando un vector lineal y se implementa un optimizador como en el caso anterior.

La implementación del modelo preentrenado se realiza de igual modo con Torchvision models (PyTorch, 2025a). Luego de cargado el modelo se congela el cuerpo y se entrenan solo las capas agregadas referidas a cada subproyecto. Al igual que en el modelo anterior se extraen features, y se implementan en la última capa que será la que aprenderá y la que se ajustará a los parámetros requeridos. Se especifica el congelamiento del modelo con los parámetros correspondientes como en el modelo anterior (pretrained=true).

Se ha empleado un learning rate de 5e-4, lote de tamaño 64 muestras y 100 épocas de entrenamiento. La optimización se realizó mediante Adam. Se usa batch normalization integrada que puede favorecer a la generalización del modelo

2.2.4. Programación de red ResNet18

Finalmente, y como consecuencia de la exhaustiva búsqueda del mejoramiento de las métricas obtenidas anteriormente se ha abordado una solución más a esta problemática consistente en el desarrollo de una red neuronal residual con una arquitectura compuesta de 18 capas de profundidad. El entrenamiento de esta red se hizo desde cero y se han obtenido buenos resultados con los datos existentes para este proyecto.

Se optó por una red residual pequeña de 18 capas de profundidad entrenada desde cero debido que es posible lograr con esta arquitectura un equilibrio bueno entre capacidad y eficiencia. Con la cantidad de imágenes disponibles en los tres conjuntos de datos es posible lograr un buen porcentaje de exactitud en comparación con otros modelos testeados desarrollados desde cero.

El entrenamiento se realiza con optimizador AdamW (weight decay, decaimiento de pesos) con lr=3e-4 y weight_decay=1e-4. En este modelo para buscar una optimización se usa AdamW que es un algoritmo utilizado para alcanzar un mejor rendimiento. La principal diferencia entre Adam y AdamW es cómo se aplica la regularización por decaimiento de peso. Mientras que Adam aplica la L2 regularization y el decaimiento de peso de forma conjunta, AdamW los desacopla, aplicando el decaimiento directamente a los pesos después de la actualización del gradiente. Además, AdamW mejora la generalización y ayuda a evitar el sobreajuste. AdamW ha demostrado ser superior al Adam estándar en la práctica, resultando en una convergencia más robusta, un entrenamiento más rápido y una mayor precisión en la validación, especialmente con arquitecturas complejas.

En cuanto al tamaño del batch se optó por 64 bits con un entrenamiento de 50 épocas. No se incorporó la técnica de dropout, dado que la arquitectura ResNet integra capas de batch normalization, las cuales contribuyen a estabilizar el proceso de entrenamiento y generan un efecto regularizador que puede favorecer la capacidad de gene-

ralización del modelo. No se consideró necesario añadir mecanismos adicionales de regularización, ya que durante la etapa experimental no se observaron indicios significativos de sobreajuste. El preprocesamiento de datos para implementar este modelo usa una transformación de 128 x 128 píxeles y se utilizan valores de normalización estándares.

Por otro lado, y con el objetivo de experimentar con diferentes alternativas de funciones de activación en este caso se utiliza SiLU (Unidad Lineal Sigmoidea). Esta es una función de activación utilizada en redes neuronales que ha ganado popularidad por su eficacia y rendimiento. Se trata de una función auto-regulada que combina con elegancia las propiedades de las funciones Sigmoid y Rectified Linear Unit (ReLU). SiLU se introdujo en el artículo "Searching for Activation Functions" (Ramachandran et al., 2017), donde originalmente se denominó Swish. Sus propiedades únicas, como la suavidad y la no monotonicidad, le permiten superar a menudo a funciones de activación tradicionales como ReLU en modelos profundos, lo que se traduce en una mayor precisión y una convergencia más rápida durante el entrenamiento del modelo.

En referencia al algoritmo, en primer lugar, se define una función que es llamada en el siguiente bloque. Esta devuelve una red secuencial con una convolución aplicando batch normalization y una función de activación SiLU. Posteriormente se define una estructura BasicBlock (bloque residual más simple) con una convolución, normalización por lotes, activación de ReLU y conexiones de acceso directo (residuales) que llama a la primera función. La red se construye completamente con cuatro capas de BasicBlocks, donde aumenta el número de canales y se realiza un muestreo descendente. Cada capa llama a la función de BasicBlocks que a su vez llama a la función inicial. Las capas residuales son aplicadas sobre cada una de las cuatro capas principales. Por último, la capa inicial se define con una función secuencial con una convolución donde se aplica la entrada de datos con un kernel de 7 x 7, se aplica normalización por lotes, activación de SiLU y la agrupación máxima. Y la capa de salida se define con un global Average Pooling que reduce cada mapa de características a un solo valor promedio, más una capa totalmente conectada de 512 (número de clases 4 o 5), y una función Softmax (implícita en la pérdida CrossEntropy) para obtener probabilidades por clase.

2.3. Base de Datos

Los datos para este proyecto son extraídos de un sistema web en producción (7 x 24) de una empresa que es contratada para controlar la importación y exportación de autos 0 KM de Argentina (*Austral Marine Services*, 2025). Los pasos para el preprocesamiento de imágenes para entrenar los modelos en este artículo se describen en (Flores & Neil, 2025). Como reseña se menciona que los datos corresponden a los últimos 10 años de inspecciones en puerto en la descarga y carga a un buque, y en el ingreso y salida de puerto o estiba. Las imágenes con las que se entrenaron los modelos fueron de 32 x 32 a 256 x 256 píxeles. Antes de introducir las imágenes en los modelos y someterlas a las capas correspondientes, las imágenes se preprocesaron para garantizar que los datos estén en un formato óptimo para el aprendizaje: cambios de tamaño, normalización, centrado y calibrado, y aumento de datos.

2.4. Herramientas de Software

El detalle de los programas utilizados se describe en (Flores & Neil, 2025). Cabe destacar que todas las herramientas utilizadas son librerías y frameworks de uso estándar en tecnología, y desarrollados por distintas empresas. El lenguaje de programación utilizado principalmente fue Python porque es una de las herramientas más utilizadas y desarrollados en ciencia de datos, inteligencia artificial y aprendizaje automático.

2.5. Herramientas de Hardware

El hardware usado para el entrenamiento se describe en (Flores & Neil, 2025). Brevemente se menciona que se han utilizado para lograr los objetivos de este artículo 3 CPU Intel Core, con RAM entre 12 y 16 GB. Y las pruebas en tiempo real vía web se realizaron con un CPU 4 Intel Core con 4 GB de RAM.

2.6. Implementación de Modelos

En (Flores & Neil, 2025) se describe como se identifican las imágenes útiles y cómo se realizan dos filtrados para cada conjunto de datos (Muestra 1). Se filtran, entre otras, imágenes etiquetadas de identificación con técnicas de reconocimiento de caracteres.

Muestra 1. Imágenes con código de barras descartadas.

Según lo publicado en (Flores & Neil, 2025) y luego del filtrado, el primer conjunto de datos contiene 50545 daños clasificados en 5 grupos balanceados de 10109 cada uno: “Dented”, “Scratched”, “Chipped”, “Missing” y “Others” (Muestra 2).

Muestra 2. Tipos de daños.

El conjunto 2 consta de 42270 partes dañadas dividida en 5 categorías balanceadas de 8454 ejemplares para cada grupo: “'Front bumper”, “Rear bumper”, “Front door”, “Rear door” y “Miscellaneous” (Muestra 3).

Muestra 3. Partes de autos con daño.

El tercer conjunto de datos consta de 52644 registros de lugares donde los daños son producidos y categorizados en 4 grupos balanceados de 13161 imágenes: “PSD” (Damage reported at preloading survey report), “SSTD” (Damage at some stage of transit), “STOW” (In stow damage) y “NTD” (Non transit damages) (Muestra 4).

Muestra 4. Lugares donde los daños son producidos.

2.7. Evaluación de Rendimientos

En este trabajo se desarrollaron funciones en Python a través de las cuales se introdujeron diferentes formatos de entrada de datos, capas convolucionales y filtros (Flores & Neil, 2025). También se hicieron pruebas con modelos preentrenados usando funciones de Python. Al igual que con los modelos desarrollados desde cero, con los modelos preentrenados se fueron ajustando los parámetros hasta obtener los mejores resultados. Y con el modelo desarrollado desde cero usando técnicas de ResNet también se hicieron pruebas con diferentes números de capas y se testearon diferentes configuraciones de parámetros hasta concluir que los mejores resultados fueron obtenidos con el modelo de 18 capas. Los modelos de mayor cantidad de horas demoraron 96 horas, mientras que los más eficientes y eficaces para los objetivos de este trabajo demoraron entre 8 y 12 horas máquina. La etapa de entrenamiento duró aproximadamente cuatro meses.

En adición, para evaluar el rendimiento de los modelos para cada uno de ellos se calcularon los valores de Precision, Recall y F1-Score. Se debe tener en cuenta que para los tres subproyectos se parte de la base de que el tipo de daño existe, o la parte dañada existe o bien el lugar donde sucedió corresponde a un daño tomado realmente. La Precision y el Recall se calcularon utilizando verdaderos positivos, falsos positivos y falsos negativos. Para las Accuracy se utilizaron verdaderos negativos y verdaderos positivos. Mientras que la métrica F1-Score se calculó para cada modelo utilizando la Precision y el Recall. F1-Score aportó generalización a los resultados debido a que tiene en cuenta el desbalance de las clases (Flores & Neil, 2025).

Finalmente, para ampliar la evaluación y análisis de los resultados obtenidos en estos modelos multiclases para cada uno se agregan las curvas ROC (Receiver Operating Characteristic) y sus correspondientes métricas asociadas de AUC (Area Under the ROC Curve). En estas figuras se puede ver la relación entre verdaderos positivos (Recall) y falsos positivos (1-Specificity) a medida que varía el umbral de decisión para lo cual se construye un gráfico con eje Y de la tasa de verdaderos positivos y eje X con la tasa de falsos positivos.

3. Resultados

Se ejecutaron más de 70 modelos diferentes de redes neuronales convolucionales entrenados desde cero (Flores & Neil, 2025). Además, se han ejecutado más de 30 diferentes variantes parametrizadas de modelos peentrenados tanto ResNet como EfficientNet. En general no se tuvo problemas ni de overfitting ni de underfitting puesto que se abordaron las problemáticas contemplando las soluciones a esos problemas desde la programación; y el filtrado y preprocesamiento de las bases de datos, y las correspondientes categorizaciones. Para mejorar los resultados obtenidos desde el inicio se testearon variaciones en los hiperparámetros como la cantidad de capas ocultas, el número de neuronas por capas y la tasa de aprendizaje aplicada a cada modelo; y esos cambios fueron cambiando las métricas más relevantes como muestran las Tablas 1, 2 y 3 de cada problema abordado. En este artículo se exponen los resultados de los mejores modelos.

A continuación, se definen las nomenclaturas utilizadas en las tablas citadas:

1. Redes Neuronales Convolucionales desde cero: **RNC1**
2. Redes Neuronales Convolucionales ResNet101: **RNC2**
3. Redes Neuronales Convolucionales EfficientNet-B0: **RNC3**
4. Redes Neuronales Convolucionales desde cero ResNet18: **RNC4**

Tabla 1. Métricas de los modelos de clasificación de tipos de daños.

Modelo	Accuracy	Precision	Recall	F1 Score
RNC1 Daños	83,00	0.95	0.89	0.92
RNC2 Daños	68,00	0.86	0.79	0.82
RNC3 Daños	**91,00**	**0.98**	**0.93**	**0.95**
RNC4 Daños	70,00	0.80	0.83	0.82

Tabla 2. Métricas de los modelos de clasificación de partes dañadas.

Modelo	Accuracy	Precision	Recall	F1 Score
RNC1 Partes	92,00	0.99	0.95	0.97
RNC2 Partes	70,00	0.78	0,88	0.83
RNC3 Partes	**97,00**	**0.99**	**0.99**	**0.99**
RNC4 Partes	79,00	0.86	0.91	0.88

Tabla 3. Métricas de los modelos de lugares donde se producen los daños.

Modelo	Accuracy	Precision	Recall	F1 Score
RNC1 Lugares	67,00	0.83	0.74	0.78
RNC2 Lugares	59,00	0.80	0.54	0.64
RNC3 Lugares	**82,00**	**0.91**	**0.84**	**0.87**
RNC4 Lugares	60,00	0.74	0.78	0.76

En los tres subproyectos los mejores resultados se han obtenido con los modelos RNC3 (remarcados) perteneciente al tipo de red preentrenada EfficientNet-B0 con números de capas iniciales de entrada de 224 x 224 neuronas.

Los resultados corresponden al conjunto de datos de prueba. Y en cada modelo los datos han sido particionados explícitamente para entrenamiento, evaluación y prueba; excepto en el modelo RNC4 donde se particionaron los datos en dos: entrenamiento y test. A continuación, se presentan detalladamente los resultados obtenidos para los mejores modelos de los tres tipos de clasificaciones propuestos.

3.1. Modelo de clasificación de tipos de daños (RNC3)

Aquí se obtuvo una exactitud global del 91%. La exactitud de "Missing" fue del 100%, "Dented" 97%, "Chipped" 89% y "Others" 100%; con esos valores se satisface la propuesta del proyecto. En los daños tipo "Scratched" (70%) se evidencia la necesidad de continuar estudiando para mejorar el porcentaje, posiblemente ciencia de datos para procesar mejores fotos desde las bases de datos (Figura 1).

Figura 1. Exactitud de clasificación de tipos de daños.

En referencia a Precision (0.98), Recall (0.93) y F1-Score (0.95), se observan valores óptimos. El desempeño del algoritmo se ve en la matriz de confusión (Figura 2). Si bien el modelo reconoce cada clase, se puede observar que el bajo porcentaje de aciertos de "scratched" se debe a que se confunde con "chipped" y en este caso es lógico puesto que en realidad son daños muy parecidos físicamente.

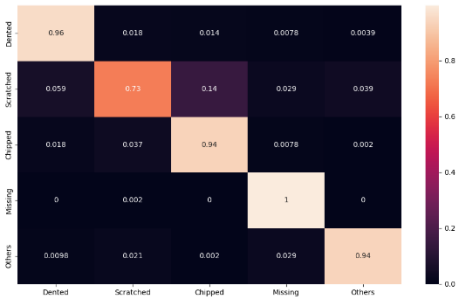

Figura 2. Matriz de confusión de tipos de daños.

Con relación a la curva ROC relacionada con los "Dented" (0.98) se puede observar que estos daños son bien distinguidos con un alto porcentaje de AUC, pero no ocurre lo mismo con los "Scratched" (0.92) donde la curva se ve más aplanada. En referencia a la curva ROC tanto de los "Chipped" (0.98) como de los "Missing" (1) se puede ver que ambos ranquean bien los positivos mostrando un alto porcentaje de AUC. Finalmente, los daños clasificados como "Others" (0.99) también muestran un valor alto de AUC, lo cual muestra que el modelo distingue este tipo de daños (Figura 3).

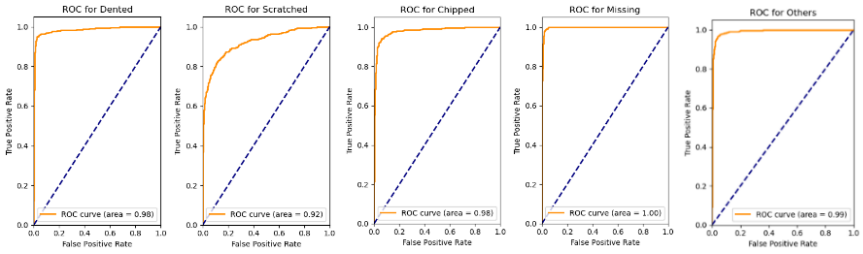

Figura 3. Curva ROC de tipos de daños.

3.2. Modelo de clasificación de partes con daños (RNC3)

Este modelo tiene los mejores porcentajes de exactitud llegando a una exactitud global del 97%. En el caso de los "Rear bumper" (100%), "Front door" (100%), "Rear door" (100%) y "Front bumper" (100%) se cumplieron las expectativas. Y en cuanto a "Miscelleanous" (94%) es necesario seguir estudiando para mejorar (Figura 4).

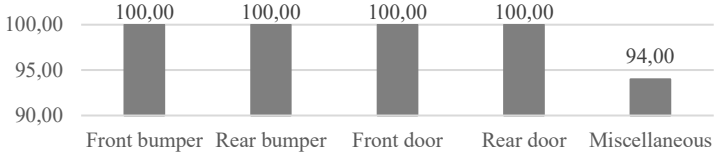

Figura 4. Exactitud de clasificación de partes dañadas.

En lo que respecta a Precision (1), Recall (1) y F1-Score (1) se observa que los valores han mejorado con respecto la red anterior. Se expone el desempeño del algoritmo en su matriz de confusión (Figura 5). Se puede observar que el porcentaje de errores en la clasificación es más bajo y equilibrado para todas las clases en general.

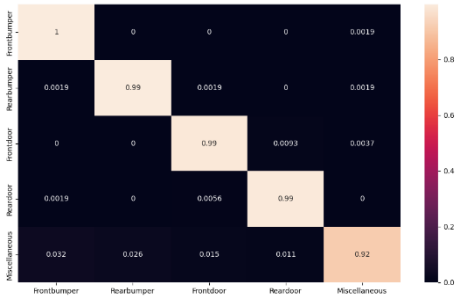

Figura 5. Matriz de confusión de partes dañadas.

En cuanto a la curva ROC relacionada con los "Front bumper" (1) se puede observar que estas partes dañadas son bien distinguidas con un alto porcentaje de AUC, y lo mismo sucede con los "Rear bumper" (1) donde la curva se ve próxima al vértice superior. En referencia a la curva ROC tanto de los "Front door" (1) como de los "Rear door" (1) se puede ver que con ambos se puede obtener un buen grado de discriminación mostrando un alto porcentaje de AUC. Por último, los daños clasificados como

"Miscellaneous" (0.99) también muestran un valor alto de AUC, lo cual prueba que el modelo distingue también este tipo de daño (Figura 6).

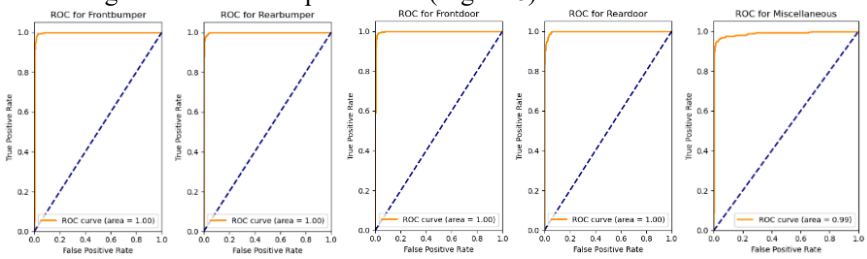

Figura 6. Curva ROC de partes dañadas.

3.3. Modelo de clasificación de lugares donde se producen daños (RNC3)

La clasificación de lugares es la más compleja de resolver y arroja valores bajos de exactitud 82% ("STOW" 89%, "PSD" 72%, "SSTD" 96% y "NTD" 90%; Figura 7).

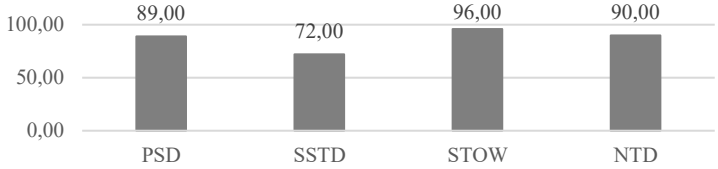

Figura 7. Exactitud de clasificación de lugares donde se producen los daños.

Aquí se intenta solucionar un problema complejo, puesto que los lugares donde se producen daños dependen del criterio de cada compañía o del personal que inspecciona, la aceptación del daño es difícil porque no hay lineamentos claros. Las métricas de precisión (0.91) y recupero (0.84) muestran desbalanceo comparado con las otras redes. El rendimiento con una media armónica de 0.87 indica que es un resultado regular, pero si se tiene en cuenta que uno de los objetivos de este artículo es demostrar la viabilidad técnica de soluciones con RNC, entonces el resultado se podría considerar bueno. La Figura 8 muestra bajo porcentaje de aciertos para "SSTD" y "NTD". En el primer caso confunde con "PSD" y "NTD". En el segundo caso confunde con "SSTD".

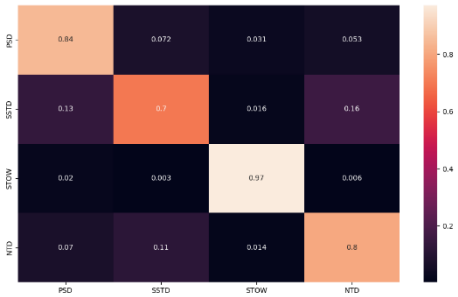

Figura 8. Matriz de confusión de lugares donde se producen los daños.

En la curva ROC de los “PSD” (0.93) y “NTD” (0.92) se puede observar que hay un bajo AUC, y en los “SSTD” (0.90) la curva se aplana más, o sea; no hay buena discriminación. La curva “STOW” (0.99) es la que mejor AUC muestra (Figura 9).

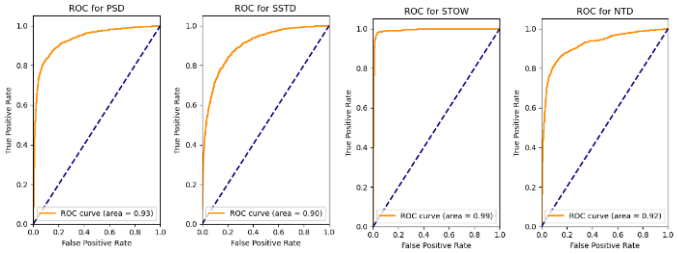

Figura 9. Curva ROC de lugares donde se producen los daños.

4. Conclusiones

Este estudio alcanzó con éxito los objetivos propuestos. Los modelos exhibieron métricas prometedoras, con tasas de exactitud superiores al 95% y se realizaron diversas pruebas obteniéndose buenos resultados. Se atribuye este logro al esfuerzo invertido en el análisis de redes preentrenadas y al estudio de viabilidad técnica de adaptación de estos modelos para el procesamiento de los datos producidos en esta industria. Actualmente se está implementado un proyecto donde se ve en línea si la predicción de datos coincide con la clasificación real vía web (*Austral Marine Services*, 2025). Si bien los tipos de daños de mejor porcentaje de clasificación son del 91% de exactitud, se evidencia la dificultad para discriminar los “Scratched” de los “Chipped”. En las partes de autos dañados los porcentajes mejoran llegando a un 97% de exactitud, pero también se debe mejorar en “Miscelleanous”. Y con respecto a la identificación de los lugares donde los daños fueron producidos (82%), este estudio muestra que se debe continuar investigando y buscando una forma de mejorar los porcentajes de exactitud.

Con respecto a los tipos de daños encontrados si bien hay buenos resultados, es necesario mejorar especialmente analizando la matriz de confusión. En referencia a los modelos de clasificación de partes dañadas los resultados obtenidos son muy buenos. Y donde es necesario mejorar, para que el sistema completo clasifique de forma óptima daños encontrados, es en los lugares donde los mismos ocurren. En cuanto a este último ítem este artículo propone mejoras en las capturas de imágenes para que la red entrene y clasifique mejor.

En adición, se puede mencionar que la limitación de este trabajo de investigación en los tres diferentes tipos de modelos está en la cantidad de categorías. O sea, se podrían agregar más categorías de lugares donde se producen daño, también más tipos de daños, e inclusive más partes de autos con daños.

Por otra parte, la fortaleza de este proyecto se centra en el hecho de demostrar a la industria que la utilización de RNC preentrenadas brinda soluciones a uno de los problemas recurrentes en el transporte marítimo de autos que es la producción de daños, y agrega a los procesos y procedimientos actuales de inspección de autos y captura de imágenes una nueva perspectiva para tomar imágenes de daños con mejor precisión.

Con relación a los trabajos futuros se observan tres líneas de investigación para mejorar todo el sistema de control en puerto en la industria marítima. El primero es el relacionado con la captura de imágenes y la clasificación de los daños en los lugares donde se producen donde principalmente es necesario cambiar la forma de captura de las imágenes para mejorar la exactitud del modelo. La segunda línea de investigación propuesta es la vinculada a la recategorización de tipos de daños y partes donde los daños son producidos puesto que es reducido el volumen de imágenes obtenidos para todas las categorías restantes. Finalmente, y con el objetivo de optimizar todo el proceso de búsqueda y registro de daños se propone el desarrollo de un sistema robótico de detección de daños que inicialmente es factible implementar en diferentes playas o patios en los puertos donde los autos son estacionados (preembarque, estiba, patios, etc.). Este proceso de detección se podría llevar a cabo en tiempos preestablecidos por las terminales portuarias permitiendo a un dispositivo robótico recorrer los lugares donde están los autos estacionados buscando daños en zonas bajas como paragolpes y guardabarros. Ese proceso reduciría la cantidad de daños no vistos en lugares de difícil acceso y también mejoraría el proceso de calidad total de vehículos en esta industria con la redistribución de tareas de los especialistas.

5. Referencias

Austral Marine Services. (2025). http://www.australmarine.com.ar/Index_en.html

Flores, H. D., & Neil, C. (2025). Clasificación de daños con visión artificial en transporte marítimo de vehículos: implementación de Redes Neuronales Convolucionales. *JAIIO, Jornadas Argentinas de Informática*, *11*(11), 1–14. https://revistas.unlp.edu.ar/JAIIO/article/view/19556

Flores, H. D., Neil, C., & Delrieux, C. (2026). Clasificación inteligente de daños en transporte marítimo de vehículos basada en visión por computadora y redes convolucionales. *SADIO Electronic Journal of Informatics and Operations Research*, *25*(1), e097–e097. https://doi.org/10.24215/15146774e097

Flores, H. D., & Neil, C. G. (2023). *Detección de daños con visión artificial en inspecciones marítimas: un mapeo sistemático de la literatura*. https://sedici.unlp.edu.ar/handle/10915/164809

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016-December*, 770–778. https://doi.org/10.1109/CVPR.2016.90

ImageNet. (2025). https://www.image-net.org/

PyTorch. (2025a). *EfficientNet — Documentación principal de Torchvision*. https://docs.pytorch.org/vision/main/models/efficientnet.html

PyTorch. (2025b). *ResNet — Torchvision main documentation*. https://docs.pytorch.org/vision/main/models/resnet.html

Ramachandran, P., Zoph, B., & Le Google Brain, Q. V. (2017). Searching for Activation Functions. *6th International Conference on Learning Representations, ICLR 2018 - Workshop Track Proceedings*. https://arxiv.org/pdf/1710.05941

Tan, M., & Le, Q. V. (2019). EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. *36th International Conference on Machine Learning, ICML 2019*, *2019-June*, 10691–10700. https://arxiv.org/pdf/1905.11946