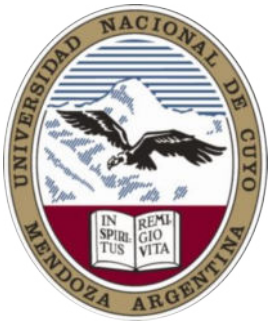
TESINA FINAL DE GRADO

Licenciatura en Ciencias de la Computación

Evaluación de daños en las ciudades producidos por desastres naturales utilizando Machine Learning

Autor:	Martín Cogo Belver
Tutor:	Dra. Ana Carolina Olivera



UNIVERSIDAD NACIONAL DE CUYO

Facultad de Ingeniería

Diciembre 2024

Agradecimientos

Quiero expresar mi más sincero agradecimiento a mi tutora de tesis, la Dra. Ana Carolina Olivera, por su invaluable guía, paciencia y apoyo constante a lo largo de este proceso. Su compromiso y experiencia fueron fundamentales para la realización de este trabajo.

Agradezco también a la Facultad de Ingeniería de la Universidad Nacional de Cuyo por proporcionar las herramientas y el espacio necesarios para llevar a cabo la elaboración de esta tesis final de carrera.

A mi familia, por su apoyo incondicional y por estar siempre presente durante cada etapa de mi formación, haciéndome sentir acompañado incluso en los momentos más desafiantes.

A mis amigos, que estuvieron a mi lado para brindarme ánimos y motivación en los momentos de mayor dificultad. Su compañía fue esencial para superar este camino.

¡Muchas gracias a todos!

Martín Cogo Belver
Mendoza, diciembre de 2024

Resumen

Desde el comienzo de la humanidad, los desastres naturales han sido un desafío constante que el ser humano ha enfrentado para poder sobrevivir. Eventos como huracanes, incendios forestales, inundaciones y sismos o terremotos han sido causantes de daños estructurales en edificios y ciudades, generando un problema de asignación de recursos escasos para el rescate de personas y reparación de daños. En este contexto, las imágenes satelitales son un recurso valioso que proporcionan una visión completa del área afectada por un desastre natural y permiten la cuantificación rápida de edificios dañados.

Teniendo en cuenta los últimos avances tecnológicos de la inteligencia artificial, especialmente en el campo del *Deep Learning*, este trabajo busca evaluar el desempeño de una red neuronal convolucional siamesa en la tarea de detección y clasificación de daños sobre edificios usando imágenes satelitales. Se adopta un enfoque de detección de cambios en imágenes antes y después de un desastre en zonas afectadas. Como parte del aporte de este trabajo se utilizan técnicas para asegurar un entrenamiento riguroso del modelo, una estrategia de muestreo desarrollada en este trabajo basada en un algoritmo voraz (*greedy*) y la implementación de un prototipo de página web que demuestra su aplicabilidad práctica.

Abstract

Since the dawn of humanity, natural disasters have been a constant challenge humans have faced to survive. These events, such as hurricanes, forest fires, floods, and earthquakes, cause significant structural damage to buildings and cities, creating problems in allocating scarce resources for rescue operations and damage repair. In this context, satellite images are a valuable resource, offering a comprehensive view of the area affected by a natural disaster and allowing for the rapid quantification of building damage.

Considering recent advances in artificial intelligence, particularly in Deep Learning, this project aims to evaluate a Siamese convolutional neural network for damage detection and classification on buildings using satellite images. A change detection approach is adopted, using pre- and post-disaster images in affected areas. As key contributions of this work, techniques are employed to ensure rigorous model training, a sampling strategy design for this work based on a greedy algorithm, and the implementation of a web prototype demonstrating its practical applicability.

Tabla de Contenidos

Índice de Tablas

Índice de Figuras

Capítulo 1

Introducción

1.1. Motivación

Desde el comienzo de la humanidad, los desastres naturales en zonas urbanas han tenido un gran impacto en la supervivencia del ser humano. Eventos como huracanes, incendios forestales, inundaciones y terremotos han causado daños estructurales significativos en edificios y ciudades [1–4].

En este contexto, las naciones enfrentan el desafío de asignar recursos escasos de manera efectiva para el rescate de personas y la reparación de daños. Para estas tareas, es crucial contar con una rápida y eficiente cuantificación de los daños estructurales en las áreas afectadas [5–7].

Las imágenes satelitales son un recurso valioso que proporciona una vista global del lugar de los acontecimientos a especialistas y permite a los expertos determinar cuántas y cuáles construcciones urbanas han sido afectadas, su ubicación y la gravedad de los daños [8].

Actualmente, la búsqueda de métodos para el análisis de imágenes satelitales es un campo activo de investigación. Considerando los grandes avances en Inteligencia Artificial (AI, *Artificial Intelligence*), Aprendizaje Automático (ML, *Machine Learning*) y especialmente en Aprendizaje Profundo (DL, *Deep Learning*), surge la oportunidad de aprovechar estas tecnologías para dar solución a este problema en el campo de la visión por computadora [9–12].

En esta tesina de grado, se propone analizar y evaluar el desempeño de un modelo de Aprendizaje Profundo (DL, *Deep Learning*) para la detección y clasificación de daños en estructuras urbanas a partir de imágenes satelitales. La arquitectura seleccionada, basada en una Red Convolucional Neuronal Siamesa (SCNN, *Siamese Convolucional Neural Network*), es ampliamente utilizada en problemas que requieren la comparación de imágenes o la detección de cambios a lo largo del tiempo. Además, se implementa un prototipo funcional de una página web que permitirá demostrar la aplicabilidad práctica de los modelos propuestos.

Durante la elaboración de esta tesis, se emplearon herramientas de inteligencia artificial, como *ChatGPT*, exclusivamente para optimizar la redacción en secciones específicas del documento. Se puso especial cuidado en garantizar que el uso de estas herramientas no afectara la precisión ni la fiabilidad del contenido presentado.

1.2. Objetivos

El objetivo principal de esta Tesina Final de Grado consiste en evaluar y comparar cambios de una arquitectura de una Red Convolucional Neuronal Siamesa (SCNN, *Siamese Convolucional Neural Network*) aplicada a la segmentación de imágenes satelitales con el fin de detectar daños producidos por un desastre natural (sismos, incendios forestales, inundaciones, huracanes, temporales, etc.) que afectan y/o destruyen edificios y casas en zonas urbanas.

Con base en lo anterior, como objetivos específicos se pueden mencionar:

- **Objetivo 1:** Entender, aplicar y evaluar diferentes modelos algorítmicos existentes en la literatura para la detección y clasificación de daños en imágenes satelitales.
- **Objetivo 2:** Entender, aplicar y evaluar el impacto de diferentes técnicas a la fase de entrenamiento y como afectan la capacidad de inferencia del modelo.
- **Objetivo 3:** En cuanto a la aplicación de los algoritmos, se espera implementar un prototipo de sistema que permita ingresar dos imágenes satelitales (pre y post desastre) de una zona urbana y logre clasificar el nivel de daño de las estructuras edilicias que aparecen en ellas.

1.3. Estructura general del documento

El presente documento se encuentra organizado de la siguiente manera:

- **Capítulo 1:** Presenta la motivación, los objetivos y la estructura del trabajo desarrollado.
- **Capítulo 2:** Brinda un marco teórico sobre conceptos de Inteligencia Artificial (AI, *Artificial Intelligence*), Redes Neuronales Artificiales (ANNs, *Artificial Neural Network*s), Redes Neuronales Convolucionales (CNNs, *Convolutional Neural Network*s) centrándose en las Red Convolucional Neuronal Siamesa (SCNN, *Siamese Convolucional Neural Network*) utilizadas para la detección y clasificación de daños. A su vez, este capítulo describe los trabajos relacionados en esta área.
- **Capítulo 3:** Detalla el diseño y desarrollo del trabajo, explicando la metodología utilizada para la implementación del sistema. Además, presenta el algoritmo seleccionado y define las diferentes instancias de prueba, mencionando las tecnologías y herramientas utilizadas.

- **Capítulo 4:** Expone los principales resultados obtenidos a partir de los diferentes experimentos realizados y el diseño e implementación de la herramienta de visualización.
- **Capítulo 5:** Presenta las conclusiones del trabajo, dificultades encontradas durante el transcurso del mismo y trabajos futuros.

Capítulo 2

Marco Teórico y Antecedentes

En este capítulo se hace una revisión de los conceptos teóricos fundamentales que dan base al proyecto realizado. Además, se lleva a cabo una breve exploración del estado del arte en relación con la clasificación de daños utilizando imágenes satelitales luego de un desastre natural sobre zonas urbanas.

Secciones del capítulo

2.1. Procesamiento de imágenes

Según Rafael y Richard [13], el *procesamiento de imágenes digitales* busca mejorar la calidad de una imagen para optimizar su interpretación o su posterior uso. En este contexto, una *imagen digital* es una función $f(x, y)$, donde x e y son coordenadas espaciales discretas, y la amplitud de f se denomina nivel de gris.

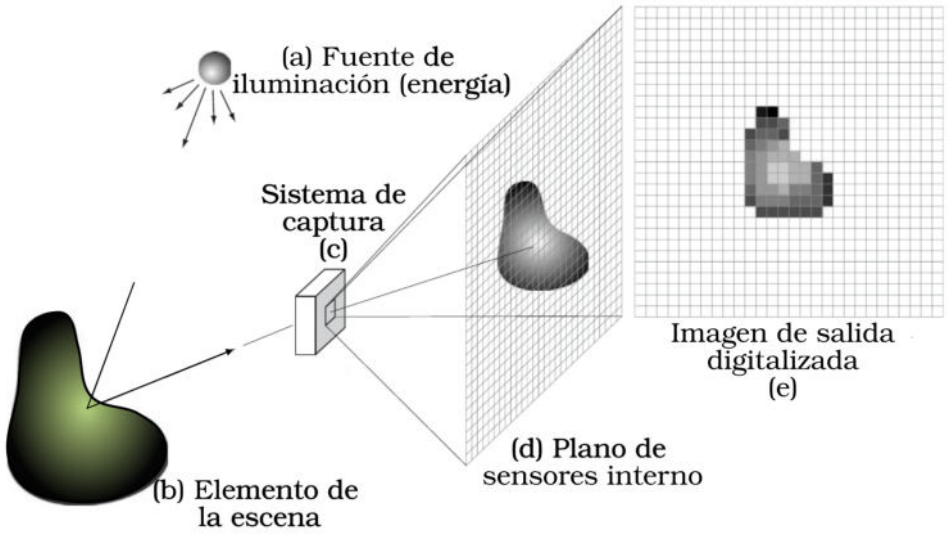


Figura 2.1: Proceso de adquisición de una imagen digital. Figura tomada de "*Digital image processing*" Rafael y Richard [13].

Muestreo y Cuantificación El proceso de captura de una imagen y su digitalización, conceptualizado en la Figura 2.1, involucra dos conceptos, *sampling* y *quantization*, ilustrados en la Figura 2.2, que son determinantes en la calidad final de una imagen digital. El muestreo (*sampling*) se refiere a la discretización de los ejes coordenados (x, y) de una imagen continua. El grado de muestreo afecta la cantidad de detalles espaciales que la imagen puede representar. Mientras que la cuantificación (*quantization*) se refiere a la discretización de la intensidad continua de una imagen en niveles, generando una *escala de grises*. La escala determina los cambios mínimos perceptibles en la intensidad de una imagen.

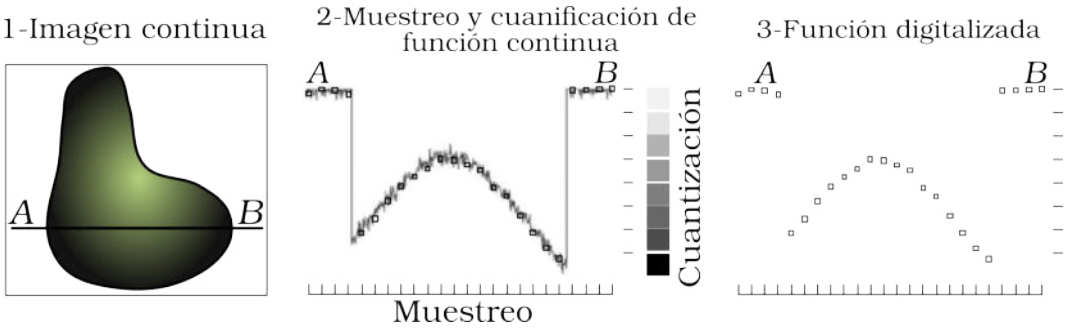


Figura 2.2: Muestreo y Cuantificación de una función continua.

2.1.1. Características de una imagen

Resolución de imagen En la práctica, las imágenes digitales se representan como un arreglo o matriz donde cada uno de sus elementos se denomina *píxel*. El concepto de *resolución* se refiere a las dimensiones de una imagen, expresadas como ancho N por alto M, es decir, $N \times M$. Por ejemplo, una imagen de resolución de 360×720 píxeles, donde $N = 360$ y $M = 720$, se representa como

$$\text{Imagen}_{N\times M} = \begin{bmatrix} p_{0,0} & \cdots & p_{0,359} \\ \vdots & \vdots & \vdots \\ p_{719,0} & \cdots & p_{719,359} \end{bmatrix}$$

Profundidad de color La *resolución de intensidad*, también conocida como profundidad de color o escala de grises, indica los cambios más pequeños en la intensidad que una imagen puede representar [13]. Debido a las limitaciones de hardware y almacenamiento, el número de niveles de intensidad suele ser una potencia de dos, 2^k donde k indica la cantidad de bits de un pixel. Por ejemplo, en una imagen de 8 bits, cada píxel puede tener un valor entre 0 y 255.

Formato de archivo Un *formato de archivo de imagen* es un estándar que organiza y almacena los datos de una imagen. Además, determina qué tipo de compresión se utiliza, reduciendo así la cantidad de datos necesarios para representarlas. Entre los formatos más comunes y utilizados tenemos *Tagged Image File Format* (TIFF), *Joint Photographic Experts Group* (JPEG), *Portable Network Graphics* (PNG), *Portable Document Format* (PDF) y *Graphics Interchange Format* (GIF) entre otros [14].

2.1.2. Modelos de color

La definición de imagen en escala de grises, en las que cada píxel codifica un nivel de intensidad de luz, puede ampliarse para incluir imágenes en que utilizan un modelo de color. El objetivo de un modelo de color es especificar colores mediante un sistema de coordenadas, donde cada color se representa como un punto. En procesamiento de imágenes, el sistema más común es el *Red Green Blue* (RGB), que se basa en la combinación de los tres colores primarios (Rojo, Verde y Azul) con distintas intensidades para generar todos los colores visibles (ver Figura 2.3). Aunque, existen otros modelos, como *Cyan Magenta Yellow* (CMY), *Cyan Magenta Yellow Black* (CMYK) y *Hue Saturation Intesity* (HSI), utilizados en diferentes aplicaciones.

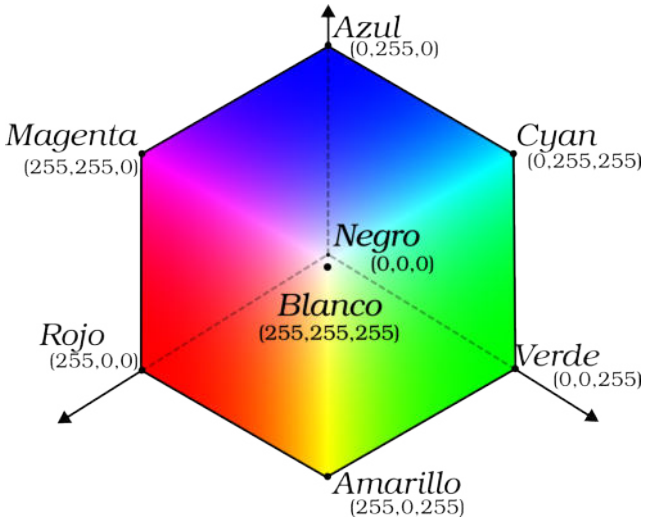


Figura 2.3: Esquema del sistema de coordenadas del modelo *Red Green Blue* (RGB) de 8-bits.

La Figura 2.4 ilustra una de las diferencias fundamentales entre las imágenes en formato RGB y aquellas en escala de grises: la cantidad de canales empleados en su representación. Las imágenes RGB están compuestas por tres canales, cada uno de los cuales corresponde a un color primario. El color de un píxel de la imagen es obtenido al combinar el valor de intensidad de los tres canales.

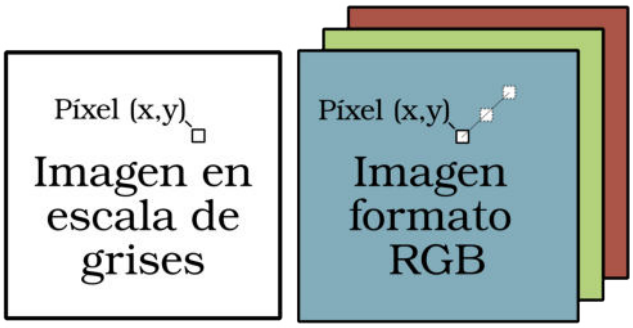


Figura 2.4: Comparación del número de canales entre una imagen en escala de grises y una imagen RGB.

2.1.3. Operaciones y Transformaciones

En esta sección se describen operaciones aritméticas, lógicas y transformaciones empleadas para el procesamiento de imágenes digitales utilizadas en esta tesis:

1. *Sustracción entre imágenes*, empleada como parte de la arquitectura del modelo de AI.
2. *Operaciones lógicas entre conjuntos*, utilizadas para el cálculo de métricas como *Intersection Over Union*.
3. *Transformaciones afines*, aplicadas durante el preprocesamiento para la aumentación de imágenes.

A continuación, se detallan cada una de estas operaciones.

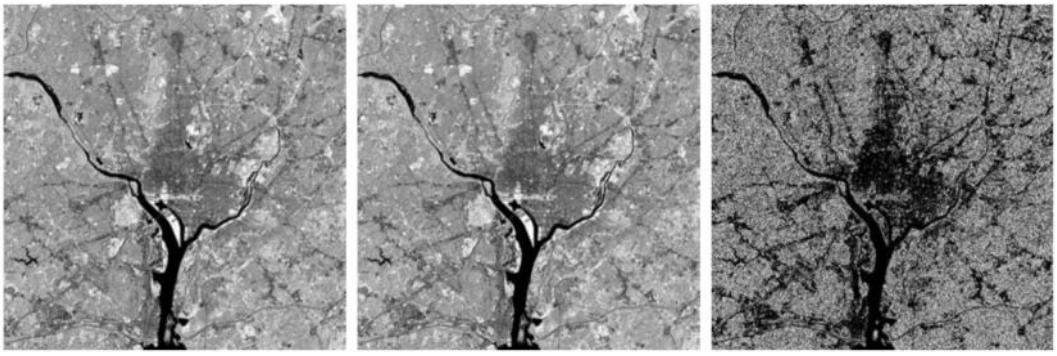


Figura 2.5: La imagen izquierda muestra la ciudad de Washington, EE.UU. La del centro resulta de poner en cero aleatoriamente el bit menos significativo de los píxeles. La derecha es la resta entre la imagen izquierda y la del centro. Figura tomada de "*Digital image processing*" Rafael y Richard [13].

Sustracción de imágenes La sustracción entre imágenes suele utilizarse para la comparación entre imágenes digitales y se interpreta como una operación de resta entre arreglos, que se lleva a cabo píxel por píxel. Es decir, siendo A y B dos imágenes de 4×4, la operación se realiza como sigue

$$A_{4\times4} - B_{4\times4} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} - \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} \\ a_{21} - b_{21} & a_{22} - b_{22} \end{bmatrix}$$

En la Figura 2.5, se observa que la diferencia entre la imagen de la izquierda y la del centro no es perceptible a simple vista. Sin embargo, al aplicar la operación de resta, obtenemos la imagen de la derecha, donde los píxeles idénticos tienen ahora un valor de cero, representado por píxeles negros.

Operaciones lógicas entre conjuntos Las regiones o conjuntos de píxeles de una imagen se identifican utilizando una máscara binaria, asignando el valor 1 a los píxeles que pertenecen a la región y el valor 0 a los que no. En lógica, interpretando el valor 1 como verdadero y 0 como falso, las operaciones entre conjuntos como Unión $A \cup B$, Intersección $A \cap B$ y Complemento A' tienen su equivalente en los operadores lógicos *OR* ($\vee$), *AND* ($\wedge$) y *NOT* ($\neg$), respectivamente. En la Figura 2.6, se representan las operaciones lógicas entre dos máscaras binarías de una imagen:

- *NOT A*: Contiene los píxeles que no están en A.
- *A AND B*: Contiene los píxeles comunes a A y B.
- *A OR B*: Contiene los píxeles que pertenecen a A, B, o ambos.
- *A AND NOT B*: Contiene los píxeles que están en A pero no en B.
- *A XOR B*: Contiene los píxeles que están en A o B, pero no en ambos.

Operación basada en transformaciones afines En el procesamiento de imágenes, las transformaciones modifican la disposición espacial de los píxeles utilizando una *transformación afín*, permitiendo escalar, rotar, trasladar o distorsionar un conjunto de coordenadas en función de los valores de una matriz **T**, como se ve en la Tabla 2.1.

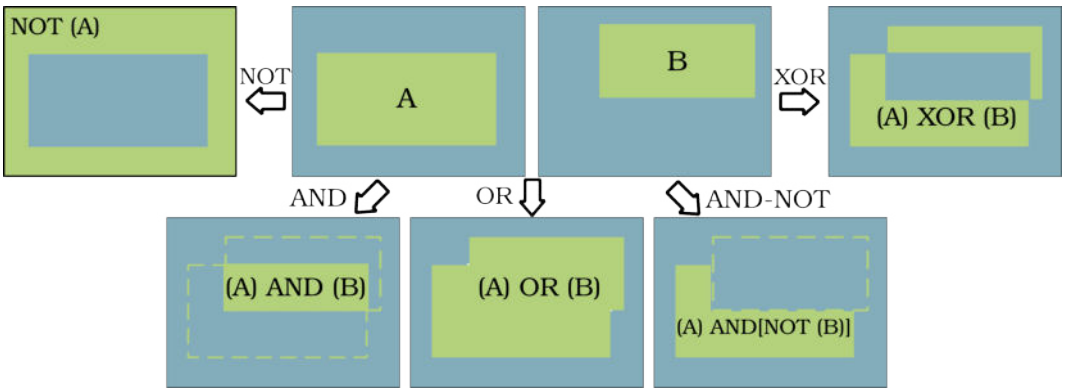


Figura 2.6: Representación gráfica de las operaciones lógicas aplicadas sobre dos regiones binarias A y B.

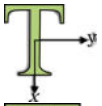
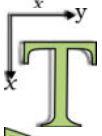
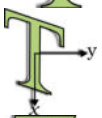
Nombre de transformación	Matriz Afín T	Ecuaciones de coordenadas	Ejemplo
Identidad	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Escalado	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotación	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
Translación	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Corte vertical	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Corte vertical	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = w$ $y = s_h v + w$	

Tabla 2.1: Tabla que muestra los diferentes tipos de transformaciones afines que pueden obtenerse cambiando los valores de la matriz T.

2.2. Imágenes Satelitales

Las imágenes satelitales son obtenidas mediante la teledetección. Chuvieco [14] define la teledetección (*remote sensing*) como el proceso de observar la Tierra desde lejos utilizando fotografías aéreas o satelitales. La Figura 2.7 muestra los principales elementos que interactúan en la tarea de teledetección. Primero, una fuente de energía, como el Sol, emite radiación electromagnética hacia la superficie terrestre y la atmósfera. Parte de esta radiación es reflejada tanto por la atmósfera como por la superficie terrestre y es captada por un sensor a bordo de una plataforma satelital. La plataforma controla la órbita, la energía y las comunicaciones del sensor. Los

datos crudos captados por el satélite son recibidos por un sistema encargado de su formateo y almacenamiento, para posteriormente ser procesados digitalmente y extraer información útil para diversas aplicaciones.

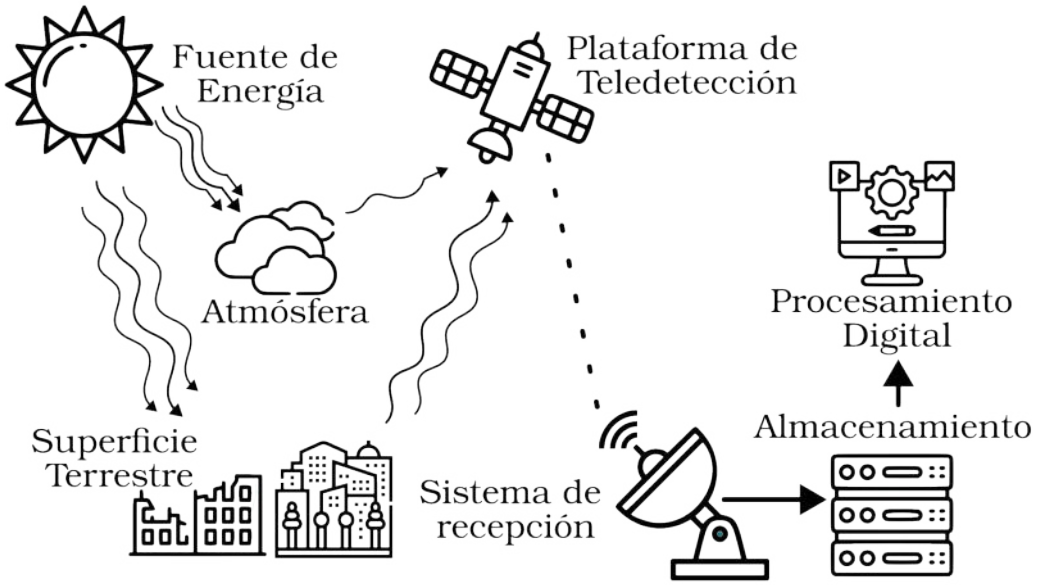


Figura 2.7: Ilustración de los principales componentes involucrados en el proceso de teledetección.

2.2.1. Causas de errores en el proceso de adquisición

Las imágenes obtenidas a través del proceso de teledetección suelen contener errores, que pueden incluir distorsiones geométricas o espaciales, así como errores en la medición de la radiación. Los errores geométricos más comunes se pueden observar Figura 2.8 y se agrupan en tres categorías [14]:

- *Errores causados por la plataforma del sensor*: Variaciones en altitud, velocidad o en orientación del sensor, provocan cambios en la escala o posición de los objetos observados.
- *Distorsión producida por los movimientos de rotación de la Tierra*: Más evidentes a medida que aumenta la distancia entre el satélite y la superficie terrestre.
- *Problemas causados por la operación de sensores*: Las distorsiones geométricas y radiométricas pueden generarse debido a fallos en el funcionamiento de los sensores.

Estos tipos de errores son comúnmente encontrados en conjuntos de datos de imágenes satelitales utilizadas para el entrenamiento de modelos de AI, y especialmente en el conjunto de datos utilizado en esta tesis.

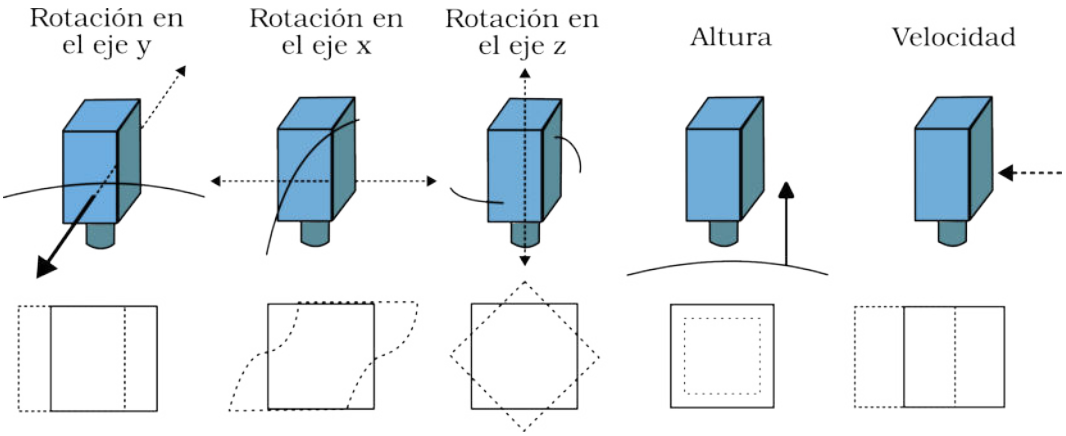


Figura 2.8: Fuente de los errores geométricos en imágenes satelitales.

2.2.2. Formatos de intercambio para datos y metadatos

El formato *Well Known Text* (WKT) se utiliza para el intercambio de datos geométricos en formato *American Standard Code for Information Interchange* (ASCII) [15, 16], siendo cadenas de texto fácilmente legibles. Permite la definición de elementos geométricos como $POINT(x, y)$ y $POLYGON((x, y), (w, z), ...)$. En las imágenes satelitales, los puntos (x, y) suelen ser coordenadas geográficas, con latitudes y longitudes medidas desde el centro de la Tierra.

JavaScript Object Notation (JSON) es un formato liviano de intercambio de datos, fácil de leer y escribir para humanos, y simple de parsear y generar para las máquinas [17]. Es ideal para el intercambio de metadatos de imágenes satelitales y puede combinarse con el formato WKT.

Ambos formatos de intercambio se utilizan para representar datos y metadatos de imágenes satelitales, siendo especialmente relevantes para el conjunto de datos empleado en esta tesis.

2.2.3. Detección de objetos en imágenes satelitales

La detección de objetos en imágenes satelitales es una tarea compleja que tiene como objetivo localizar objetos e identificar las categorías a las que pertenecen. Como expone Li et al. [18], la detección de objetos en imágenes satelitales enfrenta tres principales desafíos.

- La *extracción de características* de objetos en imágenes satelitales presenta dificultades debido a las variaciones en escala y frecuencia de los objetos, lo que exige que los modelos sean capaces de interpretar tanto la escala como la orientación.

- El *fondo de las imágenes satelitales* suele ocupar la mayor parte de la escena, lo que dificulta que el detector identifique correctamente los objetos y extraiga sus características. Esto se debe al desbalance entre la cantidad de píxeles de fondo y los píxeles pertenecientes a los objetos.

- La *anotación de instancias* en imágenes satelitales es un proceso arduo y que consume mucho tiempo. Realizar un etiquetado preciso sin errores es difícil

debido a la complejidad del fondo de las imágenes y a la variabilidad en la frecuencia y el tamaño de los objetos. Los errores de etiquetado pueden afectar significativamente el desempeño del modelo.

Segmentación semántica La tarea de *segmentación semántica* consiste en predecir y asignar una etiqueta a cada píxel de una imagen, identificando si pertenece o no a una clase semántica de interés [19]. En tareas de teledetección, los métodos de detección tradicionales dependen de la experiencia de expertos y características diseñadas manualmente. Los modelos de *Deep Learning* basados en segmentación semántica han demostrado tener mejor desempeño que los métodos tradicionales y resultan cruciales en campos enfocados en la evaluación del impacto de desastres naturales [20].

Segmentación de instancias La tarea de *segmentación de instancias* consiste en encontrar todos los objetos relevantes en una imagen y producir una máscara que resalte sus regiones visibles [19]. Aunque la segmentación semántica utilizando técnicas de *Deep Learning* ha demostrado buenos resultados, los modelos de segmentación semántica no permiten distinguir entre diferentes objetos de la misma categoría, presentando dificultades para, por ejemplo, la detección de edificios en imágenes satelitales que se encuentran muy cercanos entre sí. En comparación, los modelos de segmentación de instancias no solo pueden identificar los edificios en las imágenes individualmente, sino además clasificar cada edificio con una categoría correspondiente [21].

En este trabajo, se habría preferido, idealmente, un enfoque basado en la segmentación de instancias. Sin embargo, se adopta un enfoque de segmentación semántica multiclase para la detección y clasificación de daños en edificios, dada su simplicidad y su amplia aceptación en trabajos relacionados [9, 10, 22–24]. No obstante, se realiza una evaluación del desempeño del modelo tanto en la tarea de segmentación semántica como en la de segmentación de instancias.

2.3. Aprendizaje Automático

A continuación se introducen los campos de investigación Inteligencia Artificial, Aprendizaje Automático y Aprendizaje Profundo que fundamentan esta tesis, y su relación se ilustra en la Figura 2.9.

Inteligencia Artificial La Inteligencia Artificial (AI, *Artificial Intelligence*) estudia la creación y análisis de agentes computacionales que actúan de manera inteligente. Un *agente* toma decisiones basadas en computación y actúa en un entorno. *Aprender* es la habilidad de un agente para mejorar su comportamiento a partir de experiencias, como describe Mackworth [25].

Aprendizaje Computacional El Aprendizaje Automático (ML, *Machine Learning*) es una rama de la AI enfocada en el desarrollo de *modelos* que permite a los agentes aprender y mejorar automáticamente a partir de la experiencia. Un modelo es una representación abstracta del mundo, creada a partir de grandes volúmenes de datos, que utiliza técnicas estadísticas y matemáticas para identificar

patrones. Estos patrones permiten al agente hacer predicciones o tomar decisiones sobre datos no vistos [25].

Aprendizaje Profundo El Aprendizaje Profundo (DL, *Deep Learning*) es un subcampo del ML que se centra en el uso de redes neuronales artificiales con múltiples capas, conocidas como redes neuronales profundas, para modelar y resolver problemas complejos. El entrenamiento de estas redes se realiza mediante grandes conjuntos de datos y algoritmos de optimización, lo que permite que las redes neuronales profundas detecten patrones y realicen predicciones con una alta precisión. Este campo ha tenido grandes avances en los últimos años [26].

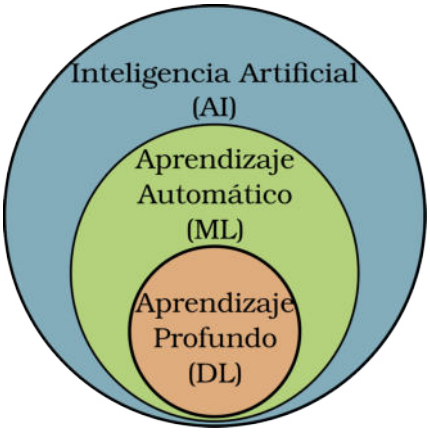


Figura 2.9: Relación entre la Inteligencia Artificial, el Aprendizaje Automático y el Aprendizaje Profundo.

2.3.1. Tipos de aprendizaje

En esta sección se introducen algunos de los principales enfoques de aprendizaje utilizados en el campo de la AI.

Aprendizaje supervisado El *aprendizaje supervisado* es un enfoque en el que se entrena un modelo utilizando un conjunto de datos (*dataset*) con ejemplos etiquetados para mejorar la capacidad de predicción o clasificación de nuevas instancias. Un conjunto de datos es una colección de pares de características de entrada (*input features*) y características objetivo (*target features*). La salida de la tarea de entrenamiento es un *predictor*, una función que predice características objetivo a partir de características de entrada. Cuando las características objetivo del conjunto de datos son discretas, se dice que el modelo es de clasificación. Por otro lado, si son valores continuos, se dice que el modelo es de regresión [25].

Aprendizaje no supervisado En el *aprendizaje no supervisado* el modelo no aprende a partir de un conjunto predefinido de ejemplos etiquetados, sino que utiliza técnicas estadísticas para analizar los datos. Estas técnicas, como *clustering* y reducción de dimensionalidad, buscan patrones o estructuras inherentes en los datos de entrada. Una de las principales diferencias con el aprendizaje supervisado es que no existe una respuesta conocida o una verdad objetiva con la cual comparar los resultados. Esto hace difícil validar o evaluar el desempeño de los modelos no supervisados [27].

Aprendizaje por refuerzo Los componentes básicos del *aprendizaje por refuerzo* son el agente y el entorno. En el aprendizaje por refuerzo, el agente es capaz de realizar un conjunto de acciones preestablecidas para interactuar con su entorno, que mediante un sistema de recompensas y penalizaciones busca que el agente optimice la toma de decisiones en función de su entorno cambiante. El entorno puede ser totalmente o parcialmente observable, lo cual condiciona la información que posee el agente para determinar cuál es la mejor acción a tomar en determinados momentos [28].

En esta tesis se utiliza un modelo de redes neuronales con un enfoque en aprendizaje supervisado para resolver un problema de segmentación semántica multiclase. Este modelo utiliza un conjunto de entrenamiento compuesto por imágenes satelitales como características de entrada y máscaras de segmentación multiclase como características objetivo.

2.3.2. Evaluación de predicciones

Para evaluar el desempeño de un modelo en sus predicciones, se utiliza una función de pérdida (*loss function*) que calcula el error entre la predicción y el valor real (*ground truth*) [28]. A continuación, se detallan algunas de estas funciones.

2.3.2.1. Funciones de perdida para modelos de regresión

Sea Y un predictor obtenido tras el entrenamiento de un modelo, y sea e una muestra perteneciente a un conjunto de datos D de tamaño N, es decir, $e \in D$. Definimos $y = Y(e)$ como la predicción del modelo y $\hat{y} = \hat{Y}(e)$ como el valor real. En los modelos de regresión, se emplean las siguientes funciones de pérdida:

- Error Absoluto Medio (MAE, *Mean Absolute Error*): Es una función que mide que tan lejos se encuentra la predicción del valor real. No tiene en cuenta que tan incorrectas son las predicciones o cuáles son mejores que otras.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}\left|y_i - \hat{y}_i\right| \tag{2.1}$$

- Error Cuadratico Medio (MSE, *Mean Square Error*): Similar a MAE pero es derivable, siempre no negativa y tienen en cuenta que predicciones son mejores que otras.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{2.2}$$

- Ráiz del Error Cuadratico Medio (RMSE, *Root Mean Squeared Error*): Tiene una interpretación más directa que MSE ya que se encuentra en la misma unidad que la predicción. Además, Penaliza más las predicciones incorrectas.

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2} \tag{2.3}$$

2.3.2.2. Funciones de perdida para modelos de clasificación

Siendo N tamaño de un conjunto de muestras, C la cantidad de etiquetas posibles. $y_{i,c}$ es un valor binario que indica si la clase c es la etiqueta real para cada muestra i. p un vector donde $p(i,c)$ es la probabilidad predicha por el modelo de que la muestra i pertenezca a la clase c. Entonces, las funciones de pérdida más comunes son:

- Entropía Cruzada binaria (BCE, *Binary Cross-Entropy*): Mide la disconformidad entre la distribución real de las etiquetas y la distribución de probabilidad predicha por el modelo. Se utiliza cuando el problema es de clasificación binaria (solo dos etiquetas):

$$\text{BCE} = -\left(y_{i,c}\log(p(i,c)) + (1-y_{i,c})\log(1-p(i,c))\right) \tag{2.4}$$

- Entropía Cruzada Promedio (MCE, *Mean Cross-Entropy*): Se trata del promedio de la entropía cruzada sobre un conjunto de datos de tamaño N y se utiliza en problemas de clasificación multiclase:

$$\text{MCE} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log(p(i,c)) \tag{2.5}$$

En este trabajo utilizaremos la función MCE, dado que el modelo se enfoca en una tarea de segmentación y clasificación multiclase con diferentes etiquetas de daño para cada edificio.

2.3.3. Métricas de desempeño

Para poder medir el desempeño de un modelo se construye una *matriz de confusión* que permite calcular las *métricas de aprendizaje*. La matriz de confusión, mostrada en la Figura 2.10, se construye a partir de las predicciones de un *predictor* $\hat{Y}$ y los valores reales. Esta matriz incluye los Verdaderos Positivos (TPs, *True Positive*s), los Falsos Positivos (FPs, *False Positive*s) (error tipo I), los Falsos Negativos (FNs, *False Negative*s) (error tipo II) y los Verdaderos Negativos (TNs, *True Negative*s).

Las métricas de aprendizaje permiten evaluar el desempeño de un modelo, proporcionan una medida cuantitativa de cómo se comporta el modelo en tareas de predicción. Estas métricas se derivan de la matriz de confusión y se detallan a continuación.

- Exactitud (ACC, *Accuracy*): Porcentaje de predicciones correctas con respecto al total de predicciones realizadas:

$$\textit{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{2.6}$$

- Precisión (P, *Precision*): Porcentaje de predicciones *positivas* correctas respecto a todas las predicciones positivas:

$$\textit{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{2.7}$$

Figura 2.10: Matriz de confusión binaria.

- Sensibilidad (R, *Recall*): Porcentaje de predicciones *positivas* correctas respecto a todos los casos positivos reales:

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.8}$$

- Puntaje F1 (F1, *F1-score*): media armónica entre P y R.

$$F1\text{-}score = 2 \times \frac{\mathrm{P} \times \mathrm{R}}{\mathrm{P} + \mathrm{R}} = \frac{2 \times \mathrm{TP}}{2 \times \mathrm{TP} + FP + FN} \tag{2.9}$$

- Porcentaje de verdaderos negativos o Especificidad (TNR, *True Negative Rate*): Porcentaje de casos negativos reales que fueron correctamente identificados como negativos.

$$True\ Negative\ Rate = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} \tag{2.10}$$

- Porcentaje de falsos positivios (FPR, *False Positive Rate*): Porcentaje de casos negativos que fueron incorrectamente identificados como positivos.

$$False\ Positive\ Rate = \frac{\mathrm{FP}}{\mathrm{TN} + \mathrm{FP}} \tag{2.11}$$

2.3.3.1. Curva ROC

Según Mackworth [25], dado que la predicción de un modelo de *clasificación binaria* es una probabilidad, es necesario establecer un umbral (*threshold*) que determine si una predicción se considera perteneciente a la clase minoritaria (clase 1 o verdadero) o a la clase mayoritaria (clase 0 o falso). Cada valor del *threshold*, dentro del rango de 0 a 1, genera un predictor diferente [29].

Las curvas ROC, ilustradas en la Figura 2.11, muestran cómo varía la métrica FPR en relación con la métrica TNR para diferentes modelos de predicción. Esta

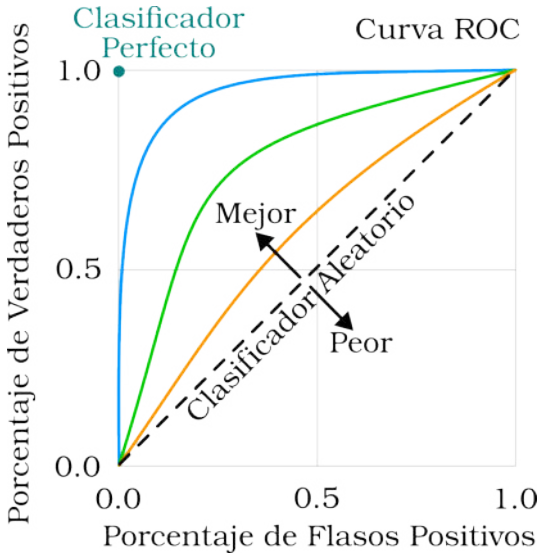


Figura 2.11: En un problema de clasificación, la curva *Receiver Operating Characteristic* (ROC), muestra el equilibrio entre el error de tipo 1 y el error de tipo 2.

curva es formada por los predictores resultantes de ajustar el umbral de clasificación desde 0 hasta 1.

En la Figura 2.11 la curva superior azul representa un modelo con mejor rendimiento que aquel representado con la curva de color naranja o verde y la recta punteada corresponde con un modelo base (*naive baseline*). Este modelo se define como aquel que ignora las *input features* para realizar predicciones.

El valor de *Area Under the ROC Curve* (AUROC) corresponde al área bajo la curva ROC, y es una métrica que permite la comparación del rendimiento entre distintos modelos. Un AUROC más alto indica un mejor desempeño del modelo.

En el caso de un modelo de *clasificación multiclase* se puede crear una curva ROC utilizando un enfoque de *uno contra todos*. Este enfoque consiste en generar una curva ROC para cada clase, transformando el problema multiclase en varios problemas de clasificación binaria Rifkin y Klautau [30].

2.3.3.2. Media armónica de F1

Al evaluar un modelo de clasificación multiclase mediante el enfoque uno contra todos, se crea la necesidad de un único valor que resuma el desempeño del modelo para la clasificación de todas las clases.

En esta tesis, se selecciona la métrica Media Armónica F1 (HMF1, *Harmonic Mean f1-score*), que es particularmente útil para conjuntos de datos desbalanceados, ya que es sensible a clases con valores de F1 bajos [9]. Siendo C el número total de clases y $F1_i$ el valor de F1 para la clase i, se define como:

$$\text{HMF1} = \frac{C}{\sum_{i=1}^{C} \frac{1}{F1_i}} \tag{2.12}$$

2.3.4. Sobreajuste

El objetivo del entrenamiento de un modelo es poder realizar predicciones sobre datos no vistos, es decir, que sea capaz de *generalizar*. Es por eso que el conjunto de datos se suele dividir en un conjunto de entrenamiento, utilizado para optimizar los parámetros del modelo, y un conjunto de prueba, utilizado para evaluar el desempeño del modelo.

El sobreajuste (*overfitting*) ocurre cuando el modelo aprende los datos de entrenamiento de manera tan específica que identifica patrones que no están presentes en el conjunto de prueba o en el mundo real. Un modelo con *overfitting* no tiene un buen rendimiento en términos de generalización. Una de las causas de *overfitting* es la complejidad del modelo, ya que los modelos más complejos tienden a sobreajustar datos más fácilmente que aquellos más simples [31].

El *overfitting* trae como consecuencia el exceso de confianza (*overconfidence*), que ocurre cuando un modelo se muestra excesivamente seguro de sus predicciones, dando una falsa ilusión de buen desempeño.

2.3.5. Validación cruzada

Para detectar cuándo un modelo está realizando *overfitting*, se emplea la técnica de validación cruzada (*cross-validation*). La idea es utilizar una porción de los datos, seleccionados de manera aleatoria, fuera del conjunto de prueba como sustituto para la evaluación. En su forma más simple, se toma una parte del conjunto de entrenamiento para ajustar el modelo, mientras que la porción restante se utiliza como *conjunto de validación* para evaluar el rendimiento del modelo [32].

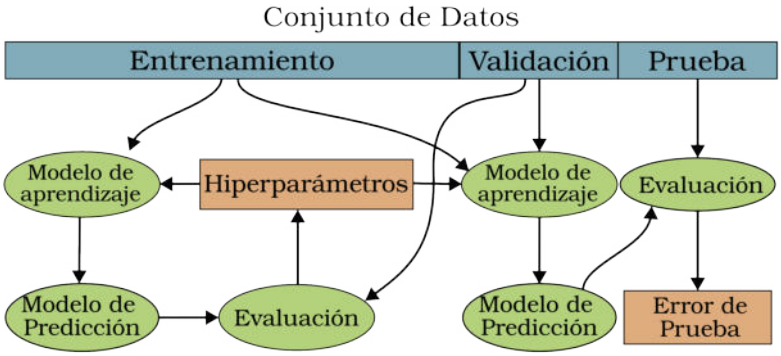


Figura 2.12: Esquema que muestra como se realiza la división de un conjunto de datos para realizar validación cruzada.

En la Figura 2.12 se ilustra cómo se relaciona cada parte del conjunto de datos durante el proceso de entrenamiento y cómo se relaciona con la búsqueda de hiperparámetros, concepto que se explica en la Sección 2.3.6.

Early stoping Durante el entrenamiento de un modelo, la función de pérdida disminuye tanto en el conjunto de entrenamiento como en el conjunto de validación. Sin embargo, esto solo se cumple hasta que el modelo comienza a sobreajustarse a los datos de entrenamiento. Al alcanzar este punto, el error en el conjunto de validación comienza a aumentar. Cuando el entrenamiento se detiene antes de que

el modelo sobreajuste los datos de entrenamiento, se conoce como parada temprana (*Early stopping*) [33].

Puntos débiles de la Validación cruzada Aunque esta técnica es relativamente simple, presenta dos puntos débiles significativos:

- La estimación del error sobre el conjunto de validación puede mostrar una alta variabilidad, ya que depende de las muestras seleccionadas para cada subconjunto.
- Solo se utilizan los datos del conjunto de entrenamiento para ajustar el modelo, excluyendo los del conjunto de validación. Entrenar con pocos datos puede conducir a un rendimiento subóptimo, produciendo que la evaluación del error sobre el conjunto de validación sobreestime el error real del modelo.

2.3.5.1. Leave-One-Out Cross-Validation

La técnica *Leave-One-Out Cross-Validation* consiste en entrenar el modelo n veces, donde n es el número de muestras del conjunto de datos. En cada iteración, una muestra se reserva para validación, mientras que el resto se utiliza para entrenamiento.

Esta metodología proporciona una estimación precisa del rendimiento del modelo al promediar los errores de cada iteración, aunque a un alto costo computacional, ya que requiere tantos entrenamientos como muestras. Por su eficiencia, *k-fold Cross Validation* es una alternativa común, ya que reduce el número de entrenamientos necesarios sin sacrificar la precisión de la evaluación.

2.3.5.2. Validación cruzada de k subdivisiones

La técnica Validación Cruzada con k subdivisiones (k-fold, *k-fold Cross Validation*) consiste en dividir el conjunto de datos en k partes iguales denominadas pliegues (*folds*) o subdivisiones. En total se realizan k entrenamientos, donde en cada uno de ellos un pliegue es utilizado como conjunto de validación y los demás como parte del conjunto de entrenamiento. De esta forma, cada uno de los pliegues es utilizado únicamente una vez como conjunto de validación.

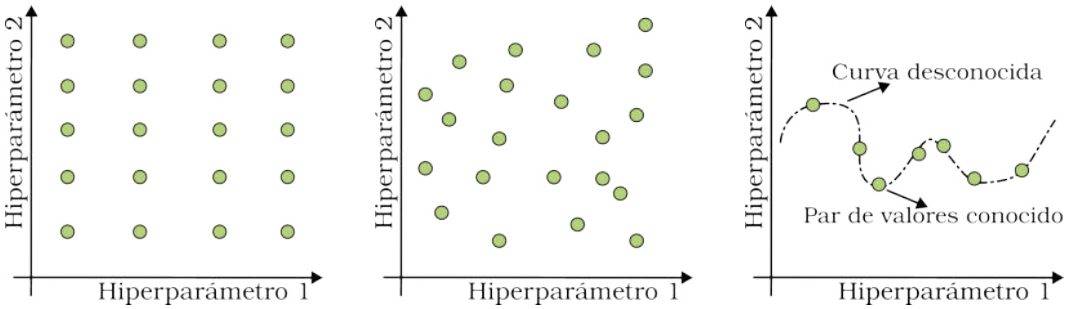
Esta técnica no es computacionalmente tan costosa como la técnica *leve-one-out* y brinda una estimación más robusta del rendimiento del modelo utilizando todas las muestras del conjunto de datos que no son parte del conjunto de prueba [34].

En esta tesis, se utiliza k-fold para obtener métricas de desempeño más rigurosas de los modelos entrenados durante la búsqueda de hiperparámetros, concepto que se detallará en la próxima sección.

2.3.6. Búsqueda de hiperparámetros

El objetivo del entrenamiento en Aprendizaje Automático (ML, *Machine Learning*) es obtener un modelo capaz de generalizar. Los *parámetros* de un modelo son aquellos que se aprenden a partir de los datos durante el entrenamiento. Por otro lado, los *hiperparámetros* son configurados antes de iniciar el entrenamiento y controlan cómo el modelo aprende a partir de los datos, influyendo directamente en su rendimiento final.

A cada conjunto de hiperparámetros con valores específicos asignados, se le denomina *configuración*. El proceso de búsqueda de la configuración que optimiza una métrica de desempeño de un modelo se denomina sintonización de hiperparámetros (*hyperparameter tuning*) [35]. Para evitar el sobreajuste, esta búsqueda se realiza ajustando los *hiperparámetros* en función del rendimiento del modelo sobre el conjunto de validación.



(a) *Grid Search* (b) *Random Search* (c) *Bayesian Optimization*

Figura 2.13: Los puntos representan una configuración de dos hiperparámetros en su dominio.

A continuación se mencionan algunas de las técnicas de búsqueda de configuraciones de hiperparámetros más usadas, las cuales se ilustran en la Figura 2.13.

- Búsqueda manual: Es el método más simple y no se trata de un algoritmo. No existen reglas definidas de cómo se realiza y está basada en el instinto del desarrollador. Consiste en, de manera artesanal, ajustar los parámetros en cada iteración de entrenamiento hasta estar satisfecho con el desempeño.

- Búsqueda de cuadrícula (*Grid Search*): Es el método automático más simple. Consiste en probar todas las combinaciones de hiperparámetros de todo el espacio o dominio. Este método tiene un alto costo computacional.

- Búsqueda aleatoria (*Randomized Search*): Es un método simple que funciona bien en la práctica. Consiste en la prueba de conjuntos de hiperparámetros seleccionados de manera aleatoria. Tiene menor costo computacional que *Grid Search*.

- Optimización bayesiana (*Bayesian Optimization*): Este método, a diferencia de los anteriores, aprende luego de la prueba de cada conjunto de hiperparámetros. Utiliza un modelo probabilístico de regresión para guiar la búsqueda de los valores óptimos de los hiperparámetros.

En esta tesis se utiliza k-fold con $k = 5$ durante la búsqueda de hiperparámetros para probar cada configuración. Esto permite obtener una evaluación más precisa del desempeño del modelo con cada configuración.

2.3.7. Técnicas para tratar datos desbalanceados

Un conjunto de datos desbalanceado (*imbalanced dataset*) es aquel en el que las etiquetas tienen una distribución desproporcionada o sesgada [36]. Esto puede dificultar el aprendizaje de clases minoritarias y generar exceso de confianza (*overconfidence*) en métricas como ACC. En esta sección, se abordan las diversas técnicas empleadas en esta tesis para tratar el desbalance de clases en el conjunto de datos, siguiendo las recomendaciones y técnicas presentadas por Abhishek y Abdelaziz [36]. A continuación, se detallan las principales estrategias utilizadas:

- La técnica de sobremuestreo aleatorio (*Random Oversampling*) consiste en disminuir el desbalance entre las clases duplicando muestras de manera aleatoria que pertenezcan a las clases minoritarias.

- La técnica de submuestreo aleatorio (*Random Undersampling*) consiste en disminuir el desbalance entre las clases eliminando muestras de manera aleatoria que pertenezcan a las clases mayoritarias.

- La técnica de ponderación de clases (*Class Weighting*) ajusta la función de pérdida asignando pesos a por cada clase, de manera que los errores en las clases minoritarias impacten más en el ajuste del modelo. El peso de cada clase se calcula con la Ecuación 2.13 donde N es el tamaño total del conjunto de datos y n_i es el tamaño de la clase i.

$$\mathrm{W}_i = \frac{N}{n_i} \tag{2.13}$$

2.3.7.1. Aumentación de datos para tratamiento de datos desbalanceados

El aumento de datos (*Data Augmentation*) permite ampliar el conjunto de datos al sintetizar nuevas muestras, lo cual suele aplicarse en diferentes situaciones.

- Para problemas donde se requieren más datos y la recopilación nuevos datos puede ser difícil o costoso.

- Como método de regularización para evitar sobre ajuste y mejorar generalización de un modelo. Especialmente para conjuntos imágenes, se utilizan transformaciones como rotación, escalado, recorte, desenfoque y adición de ruido al generar las nuevas muestras.

- Para balancear conjuntos de datos. Se pueden aplicar técnicas específicas de aumentación de datos con el fin de crear nuevas instancias que balanceen el conjunto de datos. Una de las técnicas más utilizadas en la literatura es CutMix [37], que consiste en generar nuevas instancias superponiendo un recorte de una muestra sobre otra. (Ver figura Figura 2.14). Normalmente, se aplican cambios para que los recortes contribuyan a disminuir el desbalance como se realiza en Shen et al. [10].

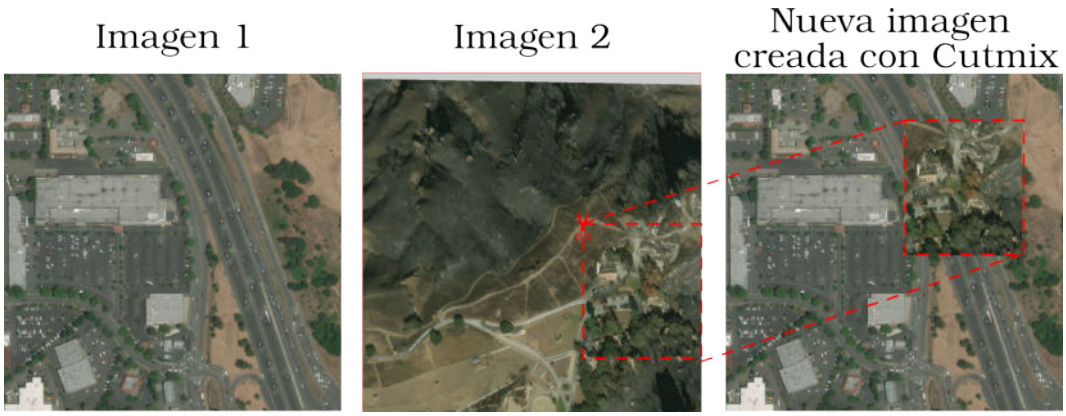


Figura 2.14: Ejemplo de realizar CutMix sobre 2 imágenes para crear una nueva.

2.3.7.2. Propuesta de técnica para tratamiento de datos desbalanceados

En esta tesis se evalúan dos enfoques para tratar el desbalance en el conjunto de datos. El primer enfoque está basado en CutMix y su implementación se detalla en la Sección 3.5.1. El segundo enfoque está basado en una estrategia de muestreo para obtener un conjunto de datos balanceado que ha sido especialmente diseñada para esta tesina final de grado.

Dado que la técnica utilizada es un submuestreo (*undersampling*) basado en optimización multiobjetivo y un enfoque voraz (*greedy*), esta sección presenta los fundamentos teóricos que la respaldan, mientras que los detalles de su implementación se describen en la Sección 3.5.2.

Optimización multiobjetivo Los *problemas de optimización multiobjetivo* buscan una solución que satisfaga varios criterios, que a menudo son conflictivos o contradictorios entre sí. En este tipo de problemas, no existe un método que lleve a una única solución que sea mejor que todas las otras. Por esta razón, las soluciones a estos problemas se describen como *eficientes* en lugar de óptimas. Cada función que representa una variable a optimizar o un criterio se denomina *variables de decisión*. La función *objetivo* es aquella que se obtiene como resultado de una combinación de todas las funciones de cada variable de decisión [38, 39].

Algoritmo voraz (*greedy*) Los algoritmos voraces son algoritmos que no exploran todo el espacio de posibles soluciones y no garantizan encontrar la solución óptima, sino que se encargan de crear la solución tomando la mejor decisión en un momento determinado. Estas decisiones son óptimas de manera local, y se espera que conduzcan a una solución cercana a la solución óptima [40].

2.4. Aprendizaje Profundo

En el área de DL, se introduce el concepto de *multilayer perceptrons* o *multilayer neural network* [41], debido a que, como explica Bishop y Bishop [42], los modelos lineales no son suficientemente expresivos como para aprender patrones complejos en gran cantidad de datos.

Un *multilayer perceptron* se construye a partir de una unidad básica llamada *neurona*, inspirada en el funcionamiento de las neuronas del cerebro humano,

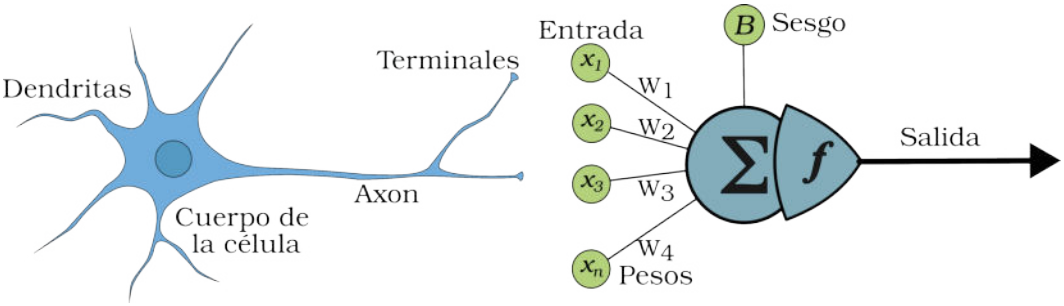


Figura 2.15: Diagrama comparativo entre la neurona real y la neurona de una red de DL.

las cuales se ordenan en capas y conectan entre sí [25]. Cada neurona puede considerarse una función de regresión que tiene parámetros ajustables, como se representa en la Figura 2.15, cuyas variables de entrada son la salida de todas las neuronas de la capa anterior.

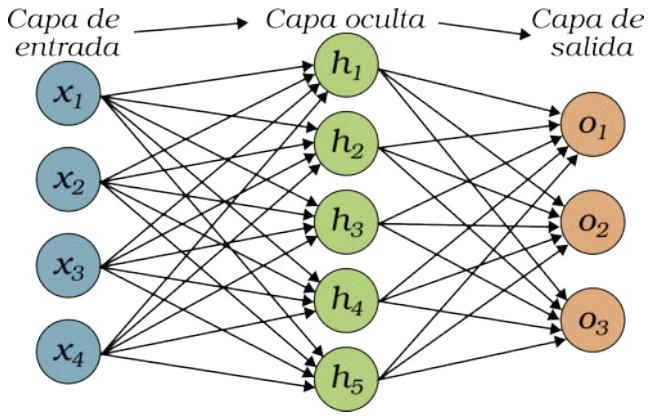


Figura 2.16: Diagrama representativo de un *Multilayer Perceptron*.

La Figura 2.16 muestra que la capa de entrada (*input layer*) es la primera capa de la red donde cada nodo representa las características de entrada de una muestra de un conjunto de datos. La capa de salida (*output layer*) es la última capa de la red y su salida corresponde con las predicciones realizadas por el modelo. Todas las capas intermedias se denominan capas ocultas (*hidden layers*). La cantidad de neuronas que posee una capa de una red se denomina *ancho*. Por otro lado, la cantidad de capas que posee la red se denomina *profundidad* de la red, y este último concepto es aquel al que se le hace referencia en con el término *Deep Learning*.

2.4.1. Funciones de activación

Una *función de activación* es una función que se aplica a la salida de las neuronas de una capa. Es una función no lineal, derivable y diferenciable que permite a una red el aprendizaje de patrones modelados con funciones no lineales [25].

2.4.1.1. Función identidad

La función identidad, graficada en Figura 2.17, es utilizada comúnmente en la capa de salida de modelos donde la salida debe ser un valor real. Esta función

permite que el valor de entrada se transmita directamente como salida, sin realizar ninguna transformación adicional.

$$\text{Identity}(x) = x \tag{2.14}$$

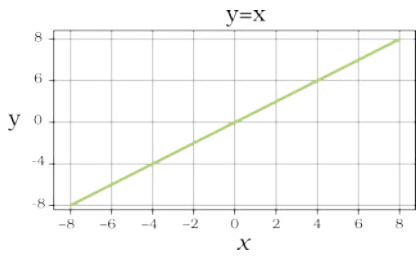


Figura 2.17: Gráfica de la función identidad.

2.4.1.2. Función ReLU (ReLU, *Rectified Lineal Unit*)

La Función ReLU (ReLU, *Rectified Lineal Unit*), graficada en Figura 2.18, que se utiliza comúnmente en las capas ocultas, ya que transforma todo valor negativo en 0, lo que ayuda a la convergencia del modelo.

$$\text{ReLU}(x) = \text{máx}(x, 0) \tag{2.15}$$

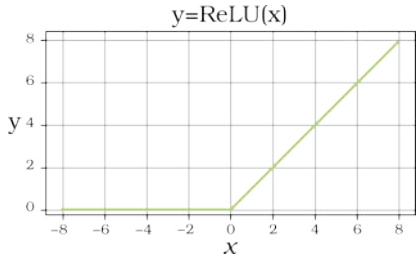


Figura 2.18: Gráfica de la función ReLU.

2.4.1.3. Función Sigmoide (Sigmoid, *Sigmoid function*)

La Función Sigmoide (Sigmoid, *Sigmoid function*), graficada en la Figura 2.19, que es utilizada en la capa de salida de modelos donde el valor de salida es una probabilidad o el valor debe estar entre 0 y 1.

$$\text{Sigmoid}(x) = \frac{1}{1 + exp(-x)} \tag{2.16}$$

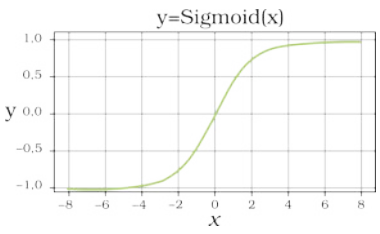


Figura 2.19: Gráfica de la función Sigmoid.

2.4.1.4. Función Tangente Hipérbolica (tanh, *hyperbolic tangent function*)

La Función Tangente Hipérbolica (tanh, *hyperbolic tangent function*), graficada en la Figura 2.20, es utilizada en la capa de salida cuando el valor de salida debe estar entre -1 y 1.

$$\tanh(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)} \tag{2.17}$$

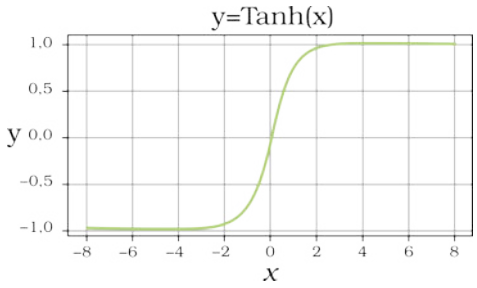


Figura 2.20: Gráfica de la tanh.

2.4.1.5. Función Softmax (Softmax, *Softmax function*)

Esta función es especialmente útil en problemas de clasificación multiclase. Se aplica a las salidas de la capa de salida, transformando los valores en probabilidades que están entre 0 y 1, y asegura que la suma de todas las probabilidades sea igual a 1, como se muestra en la Figura 2.21.

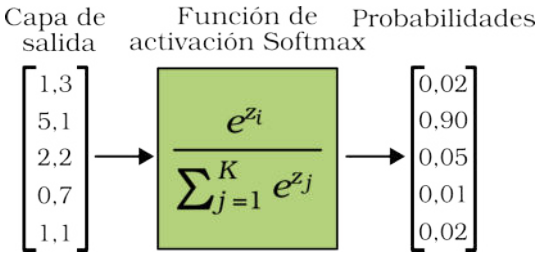


Figura 2.21: Representación de la entrada y la salida de la función softmax

2.4.2. Algoritmos de optimización gradiente descendente

El algoritmo de gradiente descendente (*gradient descent*) es fundamental para el entrenamiento de modelos en DL [43]. Este algoritmo trabaja iterativamente para minimizar el error, ajustando los parámetros del modelo en la dirección opuesta al gradiente de la función de pérdida. A continuación, se describen diferentes implementaciones de la técnica de gradiente descendente.

***Stochastic Gradient Descent* (SGD)** La forma más simple de gradiente descendente actualiza los pesos después de pasar todas las muestras del conjunto de entrenamiento por la red, lo que es ineficiente para conjuntos de datos grandes. Para solucionar esto, SGD, ilustrado conceptualmente en la Figura 2.22, divide el conjunto de entrenamiento en muestras aleatorias llamadas lotes (*minibatches*). El paso de un lote por la red y la actualización de sus pesos se denomina paso (*step*); cuando se procesan todos los lotes se denomina época (*epoch*).

***Root Mean Square Propagation* (RMSprop)** La idea de este método es ajustar el tamaño del paso de cada peso en función de su historial de cambios. Es un algoritmo de optimización que adapta el Factor de aprendizaje (LR, *Learning Rate*) de cada peso de manera independiente. Esto se logra dividiendo el LR por una media móvil de las magnitudes recientes de los gradientes para cada peso. Este método tiende a converger a la solución más rápidamente que SGD.

***Adaptive Moment Estimation* (Adam)** Este optimizador utiliza el concepto de momentum junto con las ventajas de RMSprop combinadas. Este método presenta una convergencia más rápida que RMSprop.

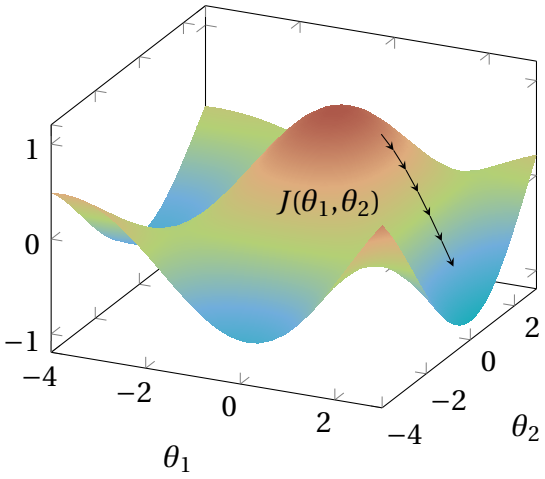


Figura 2.22: Optimización del gradiente descendente para dos parámetros θ y donde J es la función de perdida.

2.4.3. Factor de aprendizaje

El Factor de aprendizaje (LR, *Learning Rate*) es un hiperparámetro crucial que influye significativamente en la capacidad de aprendizaje de un modelo de DL [26]. Su función principal es controlar la sensibilidad con la que el modelo ajusta sus parámetros en función de los errores que comete durante el entrenamiento.

Un factor de aprendizaje pequeño provoca que el modelo minimice el error de manera menos pronunciada, generando cambios en el gradiente con menor variabilidad. Esto permite que el modelo se acerque a un mínimo absoluto de forma más lenta pero precisa. Sin embargo, si los pasos son demasiado pequeños, el proceso de optimización puede tardar excesivamente en converger hacia una solución efectiva.

Por otro lado, un factor de aprendizaje alto provoca cambios en el gradiente con gran variabilidad, lo que facilita una exploración más amplia del espacio de parámetros. No obstante, este enfoque puede resultar en saltos sobre un mínimo local, afectando negativamente el rendimiento del modelo y llevando a una solución peor que la inicial, perdiendo así la oportunidad de encontrar el mínimo deseado. Además, se puede experimentar un comportamiento errático, donde los valores de los parámetros oscilan sin converger adecuadamente a una solución óptima.

En la práctica existen diversas estrategias para ajustar el LR durante el entrenamiento.

- Descomposición *(learning rate Decay)*: Se establece una cantidad fija de épocas después de las cuales el factor de aprendizaje decrece, permitiendo que el modelo realice ajustes más finos a medida que se pasan las épocas.

- Planificador de factor de aprendizaje (*Learning Rate Schedules*): Esta técnica implica utilizar un plan para cambiar el factor de aprendizaje en función de las épocas de entrenamiento o del rendimiento del modelo sobre el conjunto de validación. Un ejemplo de planificador de aprendizaje es el *ReduceOnPlanteu* (ROPL), que reduce el factor de aprendizaje después de una cantidad específica de épocas, conocida como paciencia, durante las cuales la función de pérdida no ha mostrado mejoras significativas. Este enfoque permite ajustar dinámicamente el LR, favoreciendo una optimización más efectiva al prevenir que el modelo se estanque en un mínimo local.

- Factor de aprendizaje adaptativo (*Adaptative Learning Rates*): Son técnicas que ajustan el factor de aprendizaje de manera dinámica en función del rendimiento reciente del modelo. Por ejemplo, los algoritmos de optimización como *Adam* o *RMSprop*.

En esta tesis se realizan pruebas con el planificador de aprendizaje *ReduceOnPlanteu* (ROPL).

2.4.4. Normalización por lotes

La explosión del gradiente (*exploding gradient*) es un fenómeno que sucede cuando el error se propaga a lo largo de la red neuronal, incrementando exponencialmente de capa a capa. En estos casos, el gradiente es exponencialmente más grande con respecto a los parámetros en las capas más profundas que con respecto a aquellos en las capas menos profundas. Esto hace que la red tenga dificultades para entrenarse si es muy profunda [44]. De manera análoga, el desvanecimiento del gradiente (*vanishing gradient*) es el fenómeno que ocurre cuando los gradientes son extremadamente pequeños.

La Normalización por lotes (*batch normalization*) ajusta la salida de cada capa oculta para que tenga una media de cero y una varianza de uno. Esta técnica estandariza las activaciones, lo que facilita el aprendizaje y mejora la eficiencia del entrenamiento [42]. A diferencia de la normalización de los datos de entrada, esta debe realizarse repetidamente durante el entrenamiento. Además, *batch normalization* ayuda a mitigar problemas de *vanishing gradients* y *exploding gradients* en redes profundas, donde los gradientes pueden volverse extremos y dificultar la convergencia. En esta tesis, la normalización por lotes es una parte importante de la arquitectura seleccionada.

2.4.5. Inicialización de pesos

La inicialización de los pesos de la red afecta su tiempo de convergencia [43] durante el entrenamiento. Es por eso que existen diferentes métodos para configurar los pesos iniciales de una red:

- *Inicializador Cero*: Asigna todos los pesos a 0, aunque esto no es recomendable debido a que genera problemas de convergencia.

- *Inicializador Aleatorio Normal*: Selecciona valores de una distribución normal con una desviación estándar pequeña.

- *Inicializador Xavier o Glorot Uniform*: Este inicializador asigna valores a los pesos que son obtenidos de una distribución normal con desviación estándar uno y media cero.

En esta tesis se utiliza el iniciador *Xavier Uniform* inspirado en la implementación de Caleb Robinson [45].

2.5. Aprendizaje profundo para visión computacional

Según Sucar y Gómez [46], la *visión* es un proceso en el cual, a partir de imágenes del mundo exterior, se produce una descripción que tiene información relevante. La Visión Computacional (*Computational Vision*) es el campo de estudio de los procesos que permiten reconocer y localizar objetos en el ambiente mediante el procesamiento de imágenes.

Lograr que una computadora describa una imagen como lo haría una persona es especialmente difícil, esto se debe a que este procedimiento implica resolver un problema donde se extrae conocimiento a partir de información insuficiente. En el campo de la visión computacional se han desarrollado técnicas basadas en DL para abordar este problema.

2.5.1. Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (CNNs, *Convolutional Neural Networks*) son redes neuronales utilizadas para tratar problemas donde se requiere aprovechar información espacial con alta dimensionalidad como las imágenes [42].

En la Figura 2.23, se observa que una CNN está compuesta por varias capas, entre las que destacan las capas convolucionales, capas de pooling y capas totalmente conectadas, las cuales se describen a continuación.

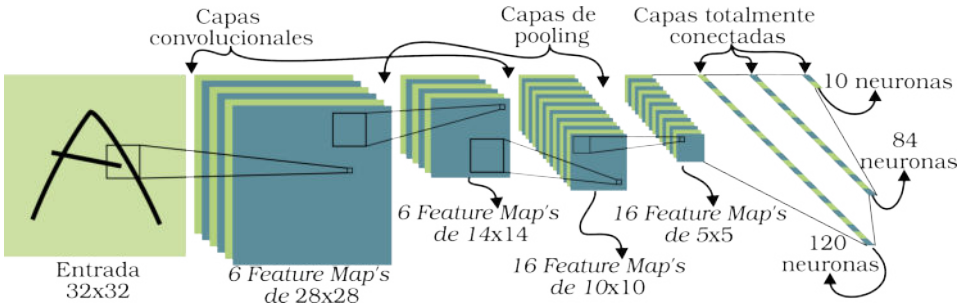


Figura 2.23: Esquema representativo de una arquitectura de una CNN.

Capa Convolucional Una capa convolucional (*Convolution layer*) tiene asociado un filtro (*kernel*) que es un operador lineal aprendido que se aplica de manera local a todos los píxeles de una imagen utilizando la operación aritmética convolución, explicada en la Sección 2.5.1. Su función principal en una CNN es la extracción de características de una imagen. El resultado de la aplicación de un filtro sobre una imagen es un mapa de características (*feature map*) como en la Figura 2.24.

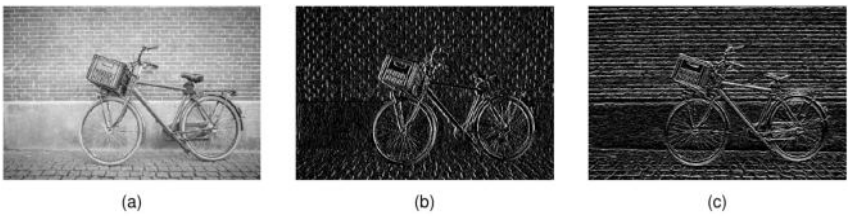


Figura 2.24: Mapas de características. (a) imagen original. (b) resultado de aplicar un filtro para detectar bordes verticales. (c) resultado de aplicar un filtro para detectar bordes horizontales. Figura tomada de "*Deep learning: Foundations and concepts*" Bishop y Bishop [42].

Capa de submuestreo El campo receptivo (*receptive field*), ilustrado en la Figura 2.25, es la porción de la imagen de entrada que activa una neurona específica en una red neuronal. En una CNN, el objetivo es que, a medida que se avanza en las capas más profundas de la red, el campo receptivo de las neuronas aumente progresivamente, lo que permite aprender características más globales de la imagen de entrada [43].

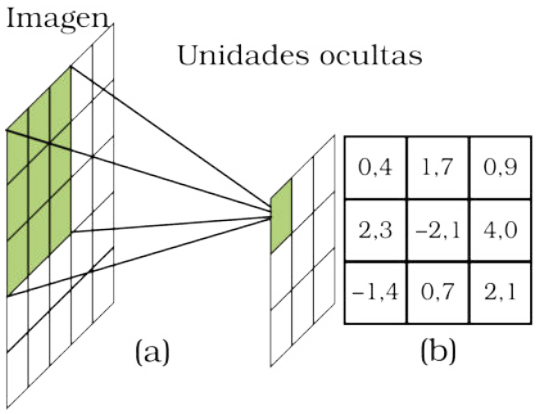


Figura 2.25: Gráfico del campo receptivo entren una imagen de entrada y una capa oculta de la red.

Las capas de submuestreo (*pooling layers*) tienen como propósito reducir las dimensiones de los mapas de características a lo largo de la red. Esto ayuda a disminuir la cantidad de parámetros, preservando la información relevante. A diferencia de las capas convolucionales, las capas de submuestreo no utilizan filtros; en cambio, aplican operadores deterministas que típicamente calculan el valor máximo dentro de una ventana (*max pooling*), como se muestra en la Figura 2.26, o el promedio de los valores (*average pooling*).

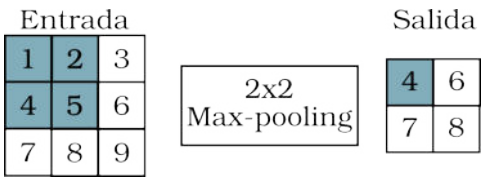


Figura 2.26: Max-pooling con un tamaño de ventana de pooling de 2x2. Las áreas sombreadas corresponden al primer elemento de salida.

Capa totalmente conectada Las últimas capas de una CNN son totalmente conectadas. Para ingresar la salida de las capas de submuestreo o convolucionales, se aplica una operación de aplanamiento (*flatten*) que convierte la salida en un vector adecuado. Esta red genera la salida final, ya sea para clasificación o regresión, basándose en las características extraídas por las capas convolucionales, como se ilustra en la Figura 2.23.

Operación aritmética convolución para imágenes Dada una imagen digital representada como una matriz I y una matriz, $K_{M\times N}$ se define la convolución, simbolizada con $(*)$, de las matrices I y K como [47]:

$$(I * K)(i,j) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I(i+m, j+n) \times K(m,n) \tag{2.18}$$

2.5.1.1. Padding

Al aplicar la convolución entre un filtro y una imagen, se pierden píxeles en los bordes de la imagen dependiendo del tamaño del filtro. Para preservar la resolución de la imagen, se utiliza *padding*, que consiste en añadir ceros en los bordes de la imagen [43]. Como se ilustra en la Figura 2.27, el *padding* garantiza que la salida de la convolución mantenga las dimensiones originales de la imagen.

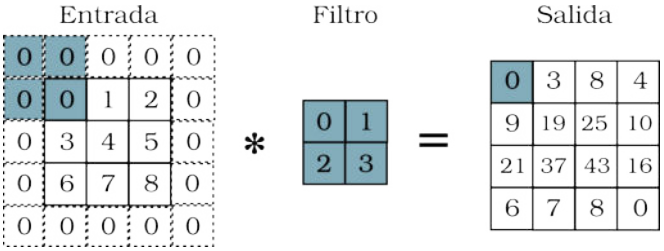


Figura 2.27: Convolución de una imagen de entrada y un filtro con *padding*.

2.5.1.2. Stride

En la Figura 2.28 se muestra el *stride*, el cual es un valor que determina el número de píxeles que se desplaza la ventana del filtro sobre la imagen en cada paso. Su utilidad principal es controlar la resolución espacial de la salida y sin necesidad de realizar una capa de *pooling* posterior [43].

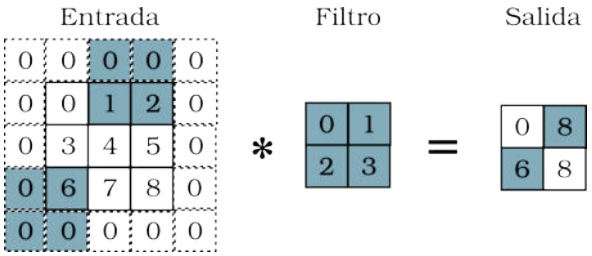


Figura 2.28: Resultado de aplicar un filtro con *stride* de 3 de altura y 2 de ancho.

2.5.1.3. Red Neuronal Convolucional aplicada a imágenes con múltiples canales

La Figura 2.29 ilustra de manera esquemática cómo se aplica un filtro a una imagen con múltiples canales. Los filtros deben tener el mismo número de canales que la entrada, y la cantidad de filtros aplicados determina el número de canales del mapa de características resultante.

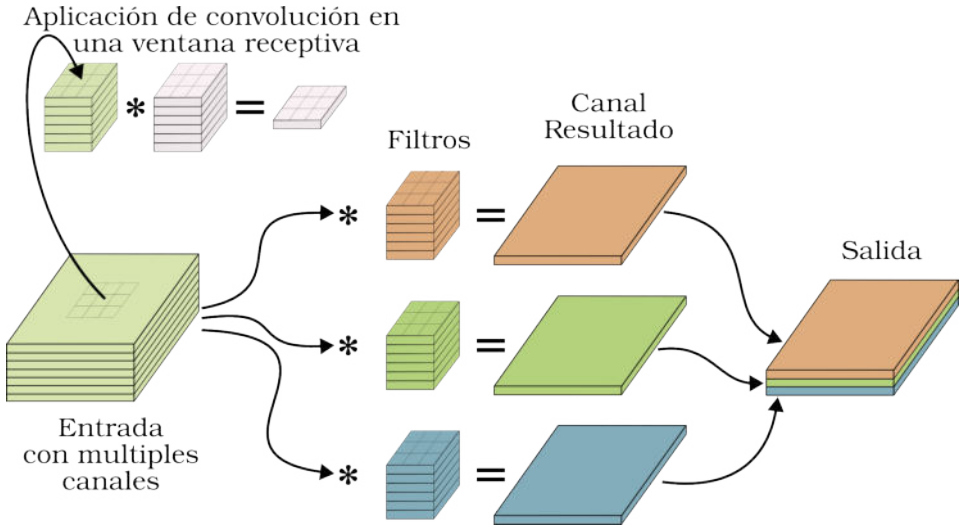


Figura 2.29: Representación del resultado de multiplicar varios filtros a una imagen con múltiples canales.

La Figura 2.30 provee un ejemplo de una convolución de dos dimensiones con dos canales de entrada. La porción sombreada contiene los elementos utilizados en la operación.

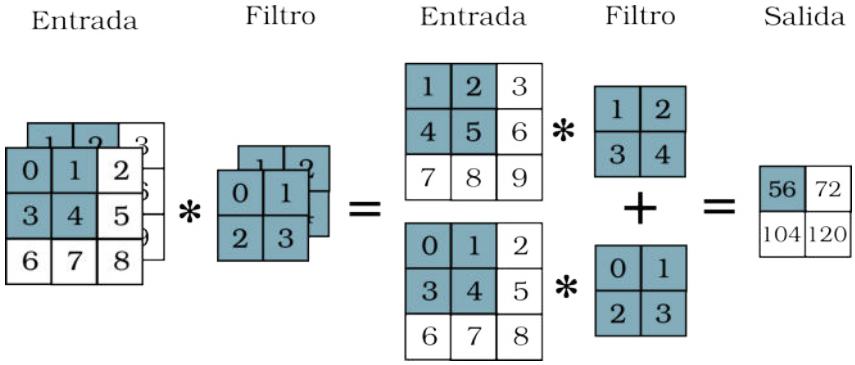


Figura 2.30: Cálculo de una convolución con dos canales de entrada.

2.5.2. Red Neuronal Totalmente Convolucional

A diferencia de las CNNs, que tienden a reducir las dimensiones de la entrada, las *Totally Convolutional Neural Networks* (TCNNs) se utilizan para resolver problemas donde la salida de la red debe conservar o aumentar la resolución de la entrada, generando una imagen o mapa de características de mayor tamaño. Se dice que estas redes son totalmente convolucionales, ya que no poseen capas totalmente conectadas.

Capa de sobremuestreo Las capas de sobremuestreo (*un-pooling layer*) son capas utilizadas para aumentar las dimensiones de su entrada [42]. Al igual que con las capas de *pooling*, se puede distinguir entre dos tipos. La capa *average-unpooling* consiste en copiar el valor de entrada en todas las unidades de salida y aplicar un promedio con todos los valores en aquellas unidades donde se presenten colisiones. Por otro lado, la capa *max-unpooling* consiste en copiar el valor de entrada en la primera unidad del bloque de salida y establecer los valores restantes en cero. La Figura 2.31 ilustra el campo receptivo de una operación de sobremuestreo.

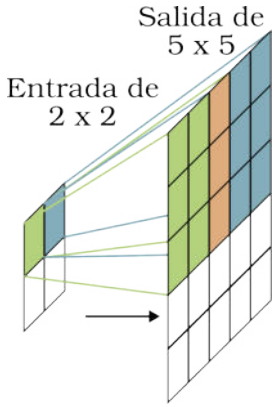


Figura 2.31: Ilustración de un sobremuestreo utilizando para un filtro de 3×3 y un *stride* de 2. Los elementos de las celdas que se superponen en la salida se calculan con el promedio de cada elemento individual solapado.

2.5.3. Arquitectura U-Net

La Figura 2.32 representa una arquitectura *U-Net*, un tipo de TCNN ampliamente utilizado en problemas de segmentación semántica [48–50]. Esta arquitectura se estructura en tres componentes principales: el codificador (*encoder*), el cuello de botella (*bottleneck*) y el decodificador (*decoder*). El codificador se encarga de la extracción de características y de la reducción de la resolución de la imagen de entrada. A continuación, el cuello de botella, que presenta el mayor campo receptivo, se enfoca en el aprendizaje de las características más generales y abstractas de la imagen. Finalmente, el decodificador tiene la responsabilidad de recuperar la resolución de la imagen de entrada a partir de la salida del cuello de botella.

El uso de capas de *pooling* y *un-pooling* en redes convolucionales profundas (CNNs) conlleva una pérdida de información espacial, tolerable en tareas de clasificación de imágenes, pero crítica en segmentación semántica. Para abordar este problema, se emplean las *skip-layer connections*, que conectan cada capa de *pooling* en la fase de codificación con su capa de *un-pooling* correspondiente en la fase de decodificación. Esto permite concatenar la salida de cada capa de *pooling* con la entrada de la capa de *un-pooling*, facilitando la recuperación de la resolución espacial perdida.

En las tareas de segmentación semántica, donde la salida es una máscara de clases, se utiliza una capa convolucional con un filtro de 1x1 que reduce el número de canales al número de clases de salida y se utiliza una función de activación *softmax* para seleccionar la clase con la probabilidad más alta.

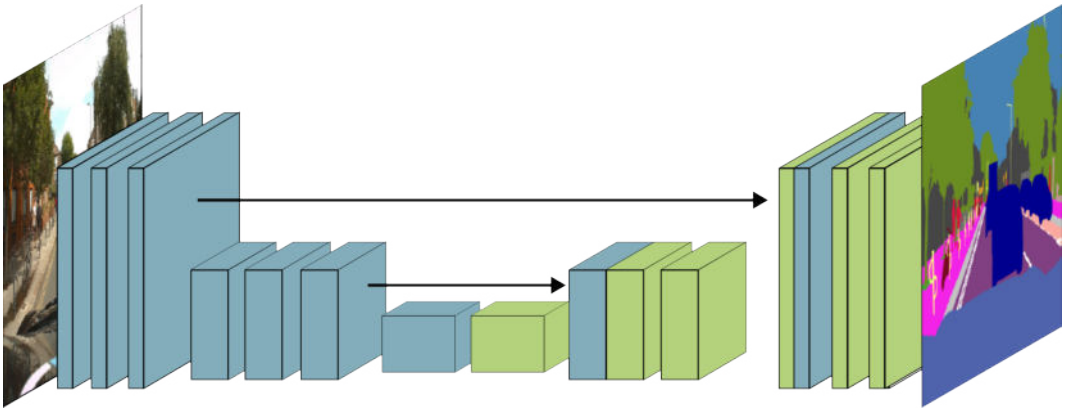


Figura 2.32: La arquitectura U-net tiene un arreglo simétrico de capas de submuestreo y sobremuestreo, conectadas por *skip-connections*. Figura tomada de "*Deep learning: Foundations and concepts*" Bishop y Bishop [42].

2.5.4. Arquitectura Siamesa

Una Red Neuronal Siamesa (SNN, *Siamese Neural Network*) es un tipo de arquitectura utilizada principalmente en tareas de detección de cambios (*change detection*) [51–53]. Gran parte de los problemas de detección de cambios a lo largo del tiempo involucran imágenes satelitales que consisten principalmente en la comparación entre dos o más imágenes de entrada.

Las SCNNs constituyen un tipo de TCNN que se distingue por procesar múltiples imágenes de entrada mediante ramas de red que comparten los mismos pesos. Para su implementación, es común definir una función de similitud destinada a comparar las entradas entre sí. Un ejemplo de esta función podría ser la resta (-) entre dos imágenes.

2.6. Antecedentes

En esta tesis, se implementa un enfoque a la Clasificación de Daños en Edificios después de un Desastre Natural (CDEN) basado en la comparación entre imágenes satelitales tomadas antes y después de la ocurrencia de un desastre natural. Este tipo de enfoque se aborda generalmente como un problema en el área de la detección de cambios a lo largo del tiempo (*change detection*). Estos tipos de problemas se enfocan en identificar un mismo objeto en varias imágenes, evaluando los cambios que experimenta a lo largo del tiempo.

En los últimos años, se han logrado avances significativos en esta área mediante el uso de DL, especialmente con la aplicación de SCNN. En [53], por ejemplo, se propone una arquitectura SCNN que integra un método multi-escala, lo cual facilita la captura de características relevantes a diferentes resoluciones. También se puede mencionar el trabajo de [52], que presenta una SCNN de tres componentes, diseñada con un marco de aprendizaje dual para optimizar tanto la representación de características de cada imagen individual como la comparación entre imágenes, mejorando así la precisión en la detección de cambios. La principal contribución de [54] radica en la implementación de muestras simuladas combinadas con una

arquitectura SCNN para abordar el desafío del desequilibrio en los datos, logrando así la detección de cambios específicos en edificios a lo largo del tiempo.

Los conjuntos de datos más utilizados para la detección de cambios en edificios son LEVIR-CD[1], OS-CD[2] y WHO-CD[3]. Sin embargo, estos conjuntos de datos no son aplicables al problema de Clasificación de Daños en Edificios después de un Desastre Natural (CDEN). En su lugar, el conjunto de datos xBD es el más adoptado en el estado del arte y fue presentado en el trabajo de Gupta et al. [22] junto con una arquitectura siamesa llamada *Rescuenet*, ampliamente utilizada para realizar comparativas de rendimiento.

En los últimos años, numerosos estudios han abordado el problema de CDEN y propuesto arquitecturas siamesas basadas en DL entrenadas con el conjunto de datos xBD, que presentan mejoras con respecto a *Rescuenet*. Algunos estudios adoptan un enfoque de entrenamiento en dos etapas. Por ejemplo, [55] entrenaron inicialmente una red *Mask R-CNN* para localizar edificios en el conjunto de datos xBD y luego transfirieron los pesos aprendidos a los *encoders* de una U-Net siamesa para la clasificación de daños. De manera similar, [10] presentaron la arquitectura *BDANet*, en la cual primero se entrenaron los pesos de una U-Net para segmentar edificios, que luego se transfirieron a una U-Net siamesa enfocada en la clasificación de daños. Esto permitió obtener un modelo que optimiza tanto la localización como la precisión en la evaluación de daños. En particular, *BDANet* incluye un módulo de fusión de características multi-escala (*MFF*) y módulos de atención cruzada.

En [23] se realiza una comparación entre dos modelos SCNN *UNet-EfficientNet-B0* y *UNet-EfficientNet-B5*. La salida del modelo *UNet-EfficientNet-B0* es una única máscara de daños con todas las clases de xBD y la salida del otro modelo *UNet-EfficientNet-B5* son varias máscaras, una por cada clase. El trabajo demostró que el mejor rendimiento en clasificación de daños se obtiene con *UNet-EfficientNet-B0*.

Otros trabajos adoptan un enfoque de aprendizaje multi-tarea para la segmentación y clasificación de daños, como el modelo *Siam-U-Net-Attn-diff* propuesto por [9], que integra mecanismos de atención para realizar ambas tareas en una única etapa. Otro trabajo relevante es el de [56] que introduce la arquitectura de aprendizaje multi-tarea *L-UNet* aplicada a *Change Detection*. Esta combina una red de segmentación semántica U-Net con un mecanismo de atención *LSTM* completamente convolucional, lo que incrementa la precisión en la detección de cambios temporales.

Algunos trabajos combinan las clases *major-damage* y *destroyed* del conjunto de datos xBD para simplificar la tarea de clasificación, como es el caso de [24] y [57]. Ambos estudios emplean una arquitectura no siamesa basada en una U-Net para extraer mapas de características de las imágenes pre y post-desastre, los cuales luego se envían a una CNN que clasifica el nivel de daño para cada edificio. Este enfoque permite un procesamiento más eficiente y preciso en la clasificación de daños en escenarios post-desastre.

Dado que el conjunto de datos xBD presenta un notable desbalance en la cantidad

[1] https://chenhao.in/LEVIR/

[2] https://rcdaudt.github.io/oscd/

[3] http://gpcv.whu.edu.cn/data/

Trabajo	Arquitectura	Cantidad de parches de xBD	Hardware	Cantidad de épocas
Gupta et al. [22]	*RescueNet*	9 168	8 NVIDIA GTX-1080	100
Hao et al. [9]	*Siam-U-Net-Attn-diff*	2 799	-	100
Zhao y Zhang [55]	*Mask R-CNN + SCNN*	12 030	1 NVIDIA Tesla P100	50
Shen et al. [10]	*BDANet*	9 168+	2 NVIDIA TITAN-V	segmentación 150 clasificación 25
May et al. [23]	*UNet-EfficientNet-B0*	9 168	-	200
Wang et al. [24]	*U-Net + CNN*	385	-	50
Kaur et al. [58]	*DAHiTrA*	9 168	4 NVIDIA Tesla V100	6 horas, 200
Dong y Zhao [59]	*Bi-DAUnet*	9 168	4 GeForce RTX 2080Ti	100
Risso et al. [57]	*U-Net + CNN*	9 168	4 NVIDIA A5000	30
Chen et al. [60]	*HRTBDA*	9 168	1 NVIDIA RTX-309	segmentación 150 clasificación 30

Tabla 2.2: Resumen del estado del arte en evaluación de daños en edificios. Para elaborar se ha asumido que los trabajos en los que no se ha mencionado la cantidad de imágenes de xBD, se ha utilizado la misma cantidad que en el modelo *Baseline*.

de edificios, presentes en las imágenes satelitales, de cada clase de daño de los edificios, diversos trabajos han propuesto enfoques y técnicas para mitigar este problema.

Entre las estrategias de tratamiento a nivel de datos o preprocesamiento para abordar el desbalance, se destacan el trabajo [55] donde se aplica una técnica de *oversampling*, que consiste en replicar imágenes de edificios de clases minoritarias hasta lograr una distribución más equilibrada de los datos. Por otro lado, [10] implementa una variante modificada de la técnica *CutMix*, orientada a aumentar las muestras de las clases minoritarias, mostrando resultados que mejoran significativamente la capacidad de aprendizaje del modelo.

También se han propuesto técnicas para abordar el desbalance durante la fase de entrenamiento, como en el trabajo de [9], que utiliza pesos de entrenamiento seleccionados empíricamente o [24], donde se implementa una técnica de muestreo denominada *normality imposition subsets*, combinada con *incremental learning*. Esta estrategia permite introducir de manera gradual conjuntos de datos pequeños y balanceados durante el entrenamiento, facilitando así una mejor convergencia del modelo.

En los últimos años, las arquitecturas que combinan SCNN con *Transformers* han demostrado un mejor desempeño que aquellas basadas únicamente en capas convolucionales como en [58] donde se presenta *DAHiTrA*, una arquitectura SCNN basada en U-Net que aplica *Transformers* para aprender las características jerárquicas entre las imágenes pre y post desastre o [59] que presenta la arquitectura *Bi-DAUnet* con bloques de Transformers denominados *BiFormer*. Ambas arquitecturas han demostrado un mejor desempeño que otras arquitecturas presentadas en años anteriores como *RescueNet*, *Siam-U-Net-Attn-diff* y *BDANet*.

La investigación más reciente sobre CDEN [60] presenta una arquitectura basada en *Transformers* entrenada con el conjunto de datos xBD y *xFBD*[1], un conjunto de datos complementario a xBD que ha mostrado los mejores resultados hasta la fecha.

[1] https://arxiv.org/abs/2212.13876

Relacionados con el CDEN se pueden mencionar trabajos que contribuyen en otros aspectos además de proponer una arquitectura, como por ejemplo [61] donde se presentan técnicas de postprocesamiento para la arquitectura *BDANet*[10], aplicadas a datos satelitales del terremoto en Turquía. Estas técnicas permiten obtener contornos de edificios más precisos y mejorar los bordes en cada máscara de nivel de daño. Por otro lado, ELEKS [11] propone una estrategia de aumentación de datos para abordar el desbalance en el conjunto de datos mediante diversas transformaciones, lo cual ha inspirado parcialmente la implementación en esta tesis.

La Tabla 2.2 presenta un resumen de la arquitectura, hardware y cantidad de épocas utilizadas por cada uno de los trabajos de CDEN explorados.

En la Tabla 2.3 se ilustra el nombre de la técnica de balance de clases utilizada en cada trabajo.

Trabajo	**Estrategia para tratar desbalance**
Gupta et al. [22]	Ninguna
Hao et al. [9]	Entrenamiento con pesos de clase
Zhao y Zhang [55]	Estrategia de *oversampling*
Shen et al. [10]	*Cutmix* modificado para tratar desbalance
May et al. [23]	Muestreo estratégico de minibatch y *balanced cross-entropy*
Wang et al. [24]	*Incremental learning with normality imposition sampling*
Kaur et al. [58]	*Focal Loss + Dice Loss*
Dong y Zhao [59]	*Aumentación de datos*
Risso et al. [57]	Ninguna
Chen et al. [60]	*Focal Loss + Dice Loss*

Tabla 2.3: Resumen de estrategias para tratar el desbalance del conjunto de datos xBD.

Capítulo 3

Diseño y desarrollo del trabajo

En este capítulo se detalla el diseño metodológico adoptado durante el desarrollo de este trabajo. Se describen las características del conjunto de datos seleccionado y las etapas realizadas para el entrenamiento del modelo de DL utilizado, y las herramientas empleadas para la implementación del proyecto.

Secciones del capítulo

3.1. Planificación de Tareas

Del análisis del estado del arte surgen numerosos desafíos para la utilización de imágenes pre y post desastre natural para evaluar los daños en construcciones

urbanas, los cuales delinean las actividades planificadas en el marco de este plan de Tesina de Grado que se muestran en la Figura 3.1.

- **Actividad A**: Revisión bibliográfica sobre DL aplicado al procesamiento de imágenes, segmentación y clasificación de objetos.
- **Actividad B**: Revisión bibliográfica de modelos de DL aplicados a CDEN.
- **Actividad C**: Revisión bibliográfica y selección de modelo para la tarea de segmentación aplicado a CDEN.
- **Actividad D**: Revisión bibliográfica y selección de modelo para la tarea de clasificación aplicado a CDEN.
- **Actividad E**: Implementación de los modelos de segmentación y clasificación analizados en la Actividad C y D.
- **Actividad F**: Implementación y evaluación de la arquitectura propuesta utilizando en primera instancia las imágenes de los conjuntos de datos xBD Gupta et al. [22].
- **Actividad G**: Construcción del prototipo de un sistema de segmentación y clasificación que permita el ingreso de la imagen pre-evento y la imagen post-desastre, y que produzca como resultado la detección de los daños en cada construcción urbana afectada.
- **Actividad H**: Redacción de la memoria de la Tesina Final de Carrera.

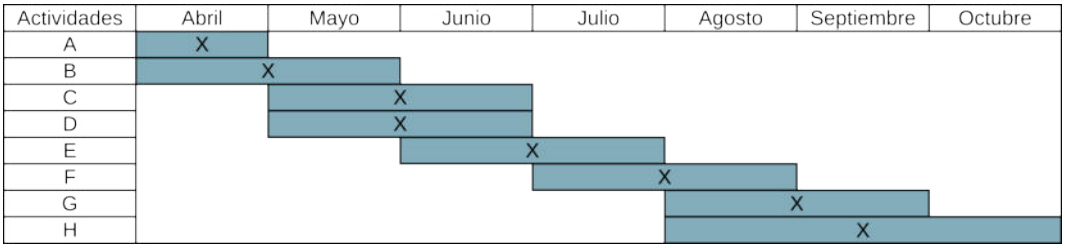


Figura 3.1: Gantt de actividades

3.2. Flujo de trabajo del proyecto

La Figura 3.2 es un diagrama que muestra todas las etapas en el flujo de trabajo empleado durante el entrenamiento del modelo de DL en esta tesis. A continuación, se describe cada etapa:

1. *Obtención del conjunto de datos*: A partir de la investigación de los trabajos relacionados, se selecciona un conjunto de datos adecuado para resolver el problema.

2. *Análisis y exploración de datos*: El análisis y exploración de datos se realiza para entender su distribución de clases, detectar casos de desbalance y realizar una limpieza de datos.

3. *Preprocesamiento de datos*: En este paso se realizan modificaciones a los datos o imágenes para dejarlos en condiciones adecuadas para el entrenamiento del modelo. Este paso incluye normalización, balanceo y aumento de datos.

4. *Entrenamiento del modelo*: Este paso consiste en la implementación y configuración del procedimiento necesario para entrenar el modelo de DL. Incluye la implementación de estrategias de aprendizaje y la búsqueda de hiperparámetros.

5. *Evaluación y validación*: Evaluación del rendimiento del modelo entrenado con datos de prueba y análisis de resultados de cada experimento realizado.

6. *Despliegue*: Si la evaluación ha sido exitosa, se cargan los pesos en el modelo que se utiliza en la página web de muestra implementada en el proyecto.

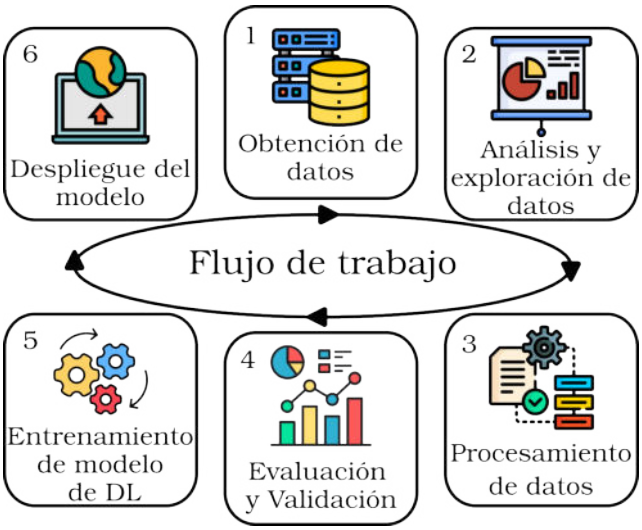


Figura 3.2: Flujo de trabajo.

3.3. Obtención del conjunto de datos

A partir del análisis de los trabajos que constituyen el estado del arte, se observa que el conjunto de datos xBD [22], proporcionado por *Maxar/DigitalGlobe* para la competencia *xView2*, es ampliamente utilizado para entrenar modelos de DL aplicados a Clasificación de Daños en Edificios después de un Desastre Natural (CDEN). La competencia fue organizada por *Defense Innovation Unit* en 2019, buscando promover el desarrollo de métodos de detección y cuantificación de daños causados por desastres naturales en imágenes satelitales, incentivando el avance en técnicas de análisis y segmentación de daños.

Este conjunto de datos ofrece imágenes satelitales de áreas afectadas por diversos desastres naturales con etiquetado realizado por expertos y se encuentra disponible gratuitamente a través del portal de xView2 [62]. Este portal permite descargar las imágenes y sus metadatos que ocupan un total aproximado de 51 GB (en formato comprimido). En la Figura 3.3 se pueden observar las zonas y los tipos de desastres naturales incluidos en el conjunto de datos.

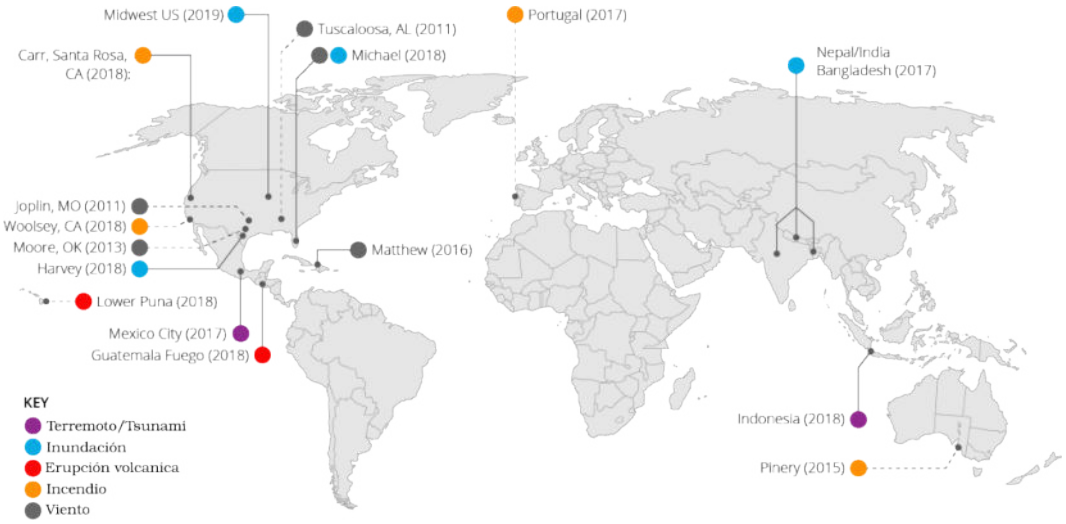


Figura 3.3: Mapa que muestra los seis tipos de desastre natural del conjunto de imágenes y las localizaciones donde han sido capturadas las imágenes satelitales. Figura tomada de "*xView2 web page*" xView [62].

3.3.1. Composición del conjunto de datos xBD

El conjunto de datos xBD proporcionado en la competencia se encuentra dividido en varios subconjuntos. En esta tesis, de manera similar a la implementación de Caleb Robinson [45], se utilizan todos los subconjuntos del concurso de xBD para entrenar el modelo.

- *train*: Datos proporcionados a los participantes para el entrenamiento.
- *test*: Imágenes utilizadas para generar una tabla de puntajes pública durante el concurso.
- *hold*: Subconjunto reservado para evaluar el desempeño de generalización de los modelos finales.
- *tier3*: Conjunto de datos adicional de entrenamiento disponible durante el desafío, que abarca una mayor variedad de desastres y áreas geográficas.

Cada subconjunto está compuesto por pares de imágenes previas y posteriores a cada desastre natural y metadatos que indican la forma y la etiqueta correspondiente a cada edificio presente en las imágenes. Cada subconjunto contiene archivos estructurados de la siguiente manera:

- El *directorio* IMAGES: Contiene imágenes satelitales *Red Green Blue* (RGB) de 8 bits con una resolución de 1024 × 1024 píxeles en formato JPEG. Cada imagen corresponde a un parche asociado a un desastre natural. El directorio contienen tanto imágenes previas como posteriores a diferentes desastres naturales.
- El *directorio* LABELS: Contiene archivos en formato JSON que describen los edificios de la imagen mediante polígonos etiquetados según su nivel de daño. Cada polígono está representado en formato WKT, con sus vértices definidos tanto en *coordenadas geográficas* como en índices de píxel correspondientes a la imagen.

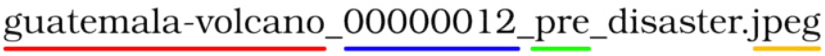


Figura 3.4: Nomenclatura de archivos de xBD.

La Figura 3.4 ilustra la nomenclatura utilizada en el conjunto xBD para relacionar las imágenes y archivos de etiqueta con los desastres naturales específicos a los que pertenecen.

- Identificador de desastre (rojo): Identificador de la zona y el tipo de desastre natural. Por ejemplo: *palu-tsunami*, *santa-rosa-wildfire* o *hurricane-harvey*.
- Identificador de parche (azul): Es un número que identifica el parche dentro del conjunto de imágenes correspondientes a un desastre natural especifico. (Este identificador puede repetirse entre desastres).
- Identificador de tiempo (verde): Indica si el parche fue tomado antes (pre) o después (post) de un desastre natural.
- Tipo de archivo (naranja): Según si se trata de un archivo JPEG o JSON podemos identificar si se trata de una imagen o un archivo con metadatos y etiquetas.

3.4. Análisis exploratorio de xBD

Antes de emplear el conjunto de datos xBD para el entrenamiento del modelo, en esta sección se lleva a cabo un análisis exploratorio de los datos con el fin de identificar tendencias y características que puedan influir en el proceso de aprendizaje.

Etiqueta	**Nombre**	**Descripción visual de la estructura**
1	Sin daños (*no-damage*)	Edificio sin signos visibles de alteración. No muestra daños estructurales, presencia de agua, pérdida de tejas ni marcas de incendio.
2	Daño menor (*minor-damage*)	Estructura con daños parciales, tales como quemaduras, presencia de agua alrededor, flujo volcánico cercano, secciones del techo ausentes o grietas visibles.
3	Daño mayor (*major-damage*)	Colapso parcial en muros o techo, con posible invasión de flujo volcánico o rodeada de agua o barro.
4	Destruido (*destroyed*)	Estructura derrumbada o ausente, completamente colapsada o cubierta parcial o totalmente por agua o barro.
5	Sin clasificar (*un-classified*)	Información insuficiente para confirmar la presencia de un edificio o clasificar el nivel de daño.

Tabla 3.1: Tabla que describe el criterio bajo el cual un edificio en una imagen satelital se considera como parte de una de las clases del conjunto de datos xBD.

En la Tabla 3.1 se muestra una tabla que especifica el color, número, nombre y criterio bajo el cual cada edificio se clasifica dentro de una clase en el conjunto de datos xBD. En la sección de resultados, se utilizarán los nombres en inglés para identificar cada clase, así como sus colores correspondientes para su representación en los gráficos.

Es importante destacar que en el conjunto de datos xBD existe también una etiqueta sin clasificar (*un-classified*) que es utilizada para aquellas estructuras que no se pudo determinar si se trataba de un edificio. Esta etiqueta se considera únicamente durante el análisis del conjunto de datos y se elimina posteriormente en la etapa de preprocesamiento. Se descarta porque no corresponde a una clase que el modelo deba aprender del conjunto de datos.

En esta tesina final de carrera el término PARCHE será utilizado para hacer referencia al conjunto de imágenes que corresponde con la imagen pre-desastre, post-desastre, y sus correspondientes máscaras de clase. Por ejemplo, en la Figura 3.5 se muestra un parche con sus imágenes y máscaras correspondientes.

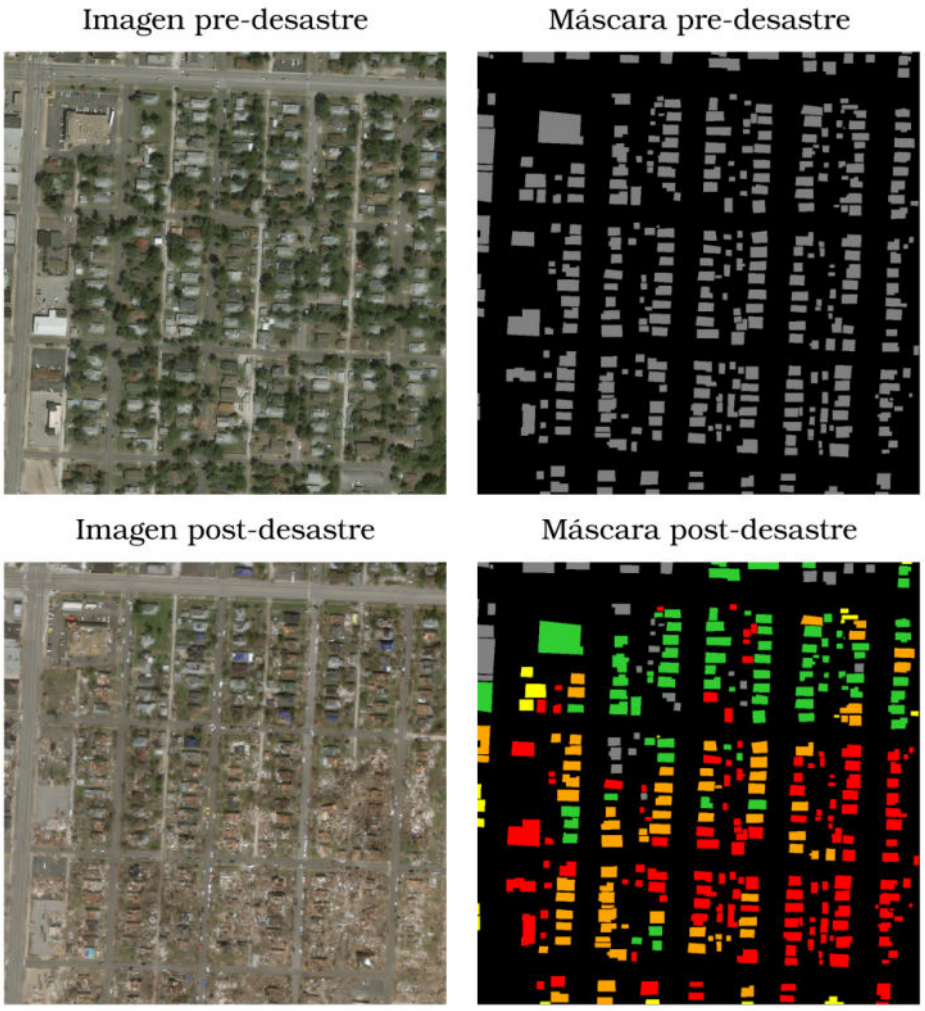


Figura 3.5: Parche 120 del desastre natural identificado como joplin-tornado en el subconjunto tier3 del conjunto de datos xBD.

Las imágenes satelitales del conjunto xBD presentan errores como deformaciones geométricas, desplazamientos de los edificios y distorsiones en la medición de radiación. Estos problemas introducen desafíos adicionales en el proceso de aprendizaje del modelo para la tarea de CDEN, complicando la capacidad del modelo para generalizar correctamente.

La Figura 3.6 muestra algunos de estos errores de adquisición en el conjunto xBD. La imagen superior izquierda ilustra recortes parciales en algunas imágenes, mientras que la imagen superior derecha evidencia una desalineación entre los edificios y sus respectivas máscaras de clase post-desastre. Además, en la imagen inferior derecha, se observan diferencias notables de iluminación entre las imágenes pre y post-desastre, lo que resalta variaciones en la calidad de las imágenes en el conjunto de datos.

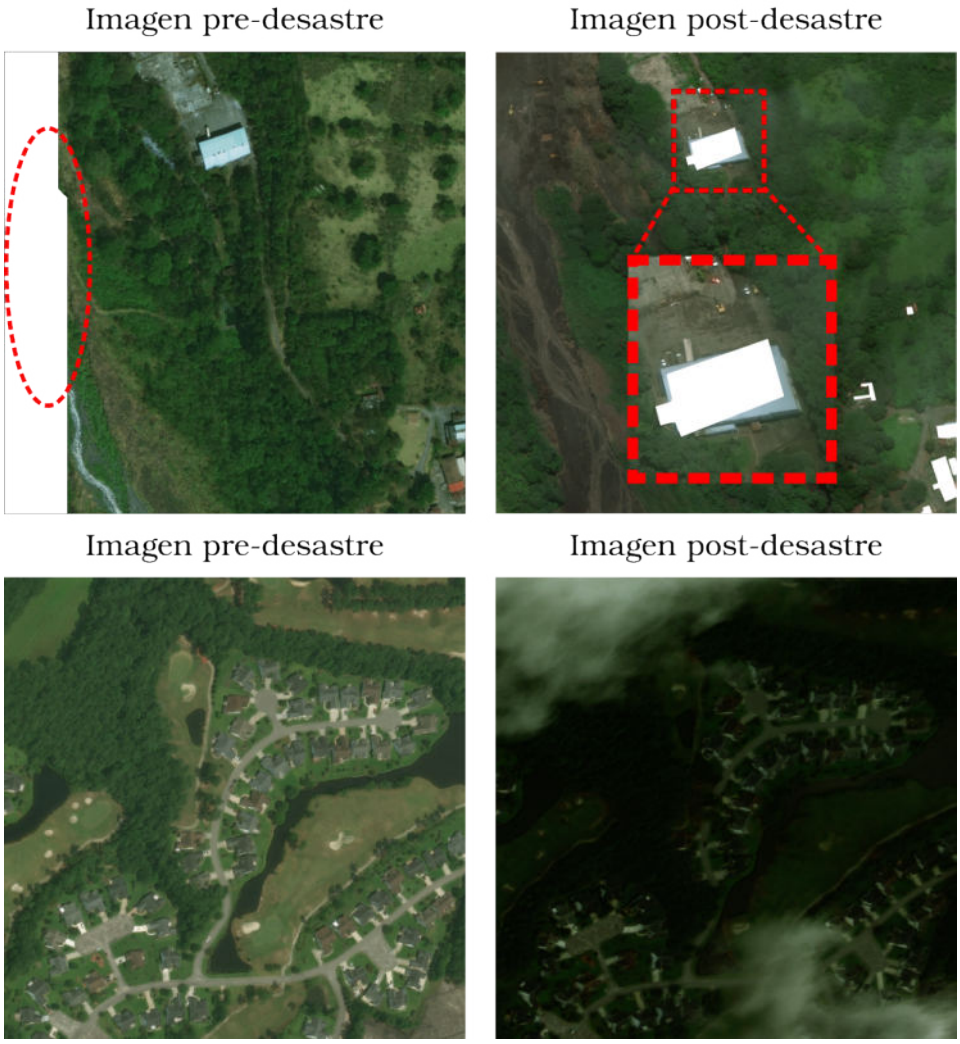


Figura 3.6: Ejemplos de imágenes con errores de captura del conjunto de datos xBD. Las imágenes superiores corresponden al parche 0 del desastre *guatemala-volcano*. Por otro lado, las imágenes inferiores pertenecen al parche 3 del desastre *hurricane-florence*.

3.4.0.1. Cantidad de imágenes por desastre natural

En la Tabla 3.2, se presenta la cantidad de parches que tiene cada desastre natural en el conjunto de datos xBD. Es importante destacar que por cada parche existen dos imágenes previas y posteriores al desastre en cuestión y se crearán dos máscaras de edificios. Se puede observar un desbalance en la distribución de parches entre los diferentes tipos de desastres. Hay una mayor cantidad de zonas correspondientes a huracanes y tornados en comparación con terremotos o erupciones volcánicas.

Tipo de desastre	Nombre de desastre	Cantidad de parches	Total
Earthquake/Tsunami (Terremoto/Tsunami)	sunda-tsunami	148	**537**
	mexico-earthquake	193	
	palu-tsunami	196	
Flooding (Inundación)	midwest-flooding	445	**1064**
	nepal-flooding	619	
Tornado/Huracane (Tornado/Huracán)	joplin-tornado	149	**2742**
	moore-tornado	227	
	tuscaloosa-tornado	343	
	hurricane-matthew	405	
	hurricane-harvey	522	
	hurricane-florence	546	
	hurricane-michael	550	
Volcanic Eruption (Erupción Volcanica)	guatemala-volcano	28	**319**
	lower-puna-volcano	291	
Wildfire (Incendio forestal)	santa-rosa-wildfire	377	**6372**
	woolsey-fire	878	
	socal-fire	1403	
	pinery-bushfire	1845	
	portugal-wildfire	1869	

Tabla 3.2: Cantidad de parches que hay por cada desastre natural del conjunto de datos xBD

Observando la Figura 3.7 se puede notar que la cantidad de edificios que se presentan en las imágenes varía considerablemente. Por ejemplo, la imagen representativa de *Hurricane Harvey* o *Joplin Tornado* presentan gran cantidad de edificaciones en ellas, muestras que las imagenes representativas de *Pinery Bushfire* o *Lowe Puna Volcano* apenas ocupan un pequeño porcentaje.

3.4.0.2. Cantidad de edificios por desastre natural

En la Tabla 3.3 se resume el conteo de la cantidad de edificios de cada clase que hay en cada desastre natural de conjunto de datos xBD y se puede ver que en total hay 425.368 edificios. Además, en la Figura 3.8, se ilustra la notable desproporción entre las clases del conjunto de datos, especialmente debido a la abundancia de edificios clasificados como *minor damage*. Por otro lado, observando Figura 3.9, también podemos notar que hay un gran desbalance en la cantidad de edificios que hay por cada desastre natural.

Figura 3.7: Una imagen representativa de cada desastre natural en el conjunto de datos en xBD.

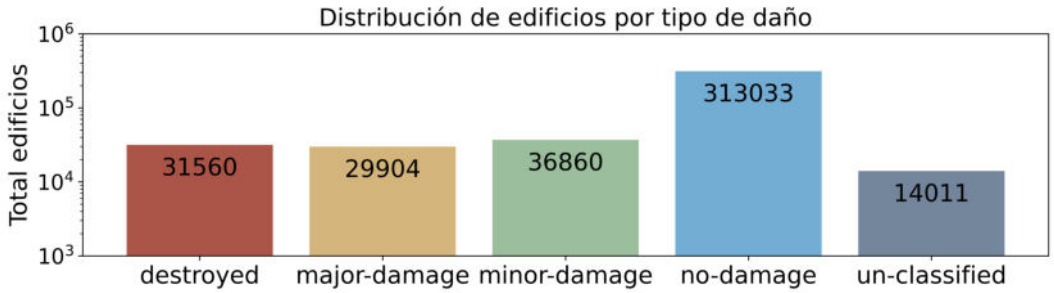


Figura 3.8: Cantidad de edificios por nivel de daño en xBD.

Etiqueta / Desastre	*no-damage*	*minor-damage*	*major-damage*	*destroyed*	*un-classified*	**Total**
guatemala-volcano	731	26	23	33	178	991
hurricane-florence	8 466	232	1 949	81	820	11 548
hurricane-harvey	18 638	4 510	**13 378**	848	581	37 955
hurricane-matthew	4 058	**12 331**	2 717	3 524	1 334	23 964
hurricane-michael	22 692	8 292	2 919	1 225	373	35 501
joplin-tornado	8 225	2 192	1 005	3 274	656	15 352
lower-puna-volcano	2 277	49	26	504	554	3 410
mexico-earthquake	51 084	221	54	3	111	51 473
midwest-flooding	12 819	246	193	165	473	13 896
moore-tornado	19 453	886	449	1 584	586	22 958
nepal-flooding	31 225	5 134	4 721	502	1 683	43 265
palu-tsunami	**46 796**	1	1 178	**7 203**	611	**55 789**
pinery-bushfire	5 027	82	99	229	524	5 961
portugal-wildfire	20 787	176	296	1 090	1 064	23 413
santa-rosa-wildfire	15 843	121	95	5 810	86	21 955
socal-fire	15 697	136	110	2 333	693	18 969
sunda-tsunami	14 078	0	100	179	**2 590**	16 947
tuscaloosa-tornado	10 499	2 036	466	1 097	908	15 006
woolsey-fire	4 638	189	126	1 876	186	7 015
Total	**313 033**	**36 860**	**29 904**	**31 560**	**14 011**	**425 368**

Tabla 3.3: Tabla que muestra la cantidad de edificios de cada tipo por cada desastre y sus totales.

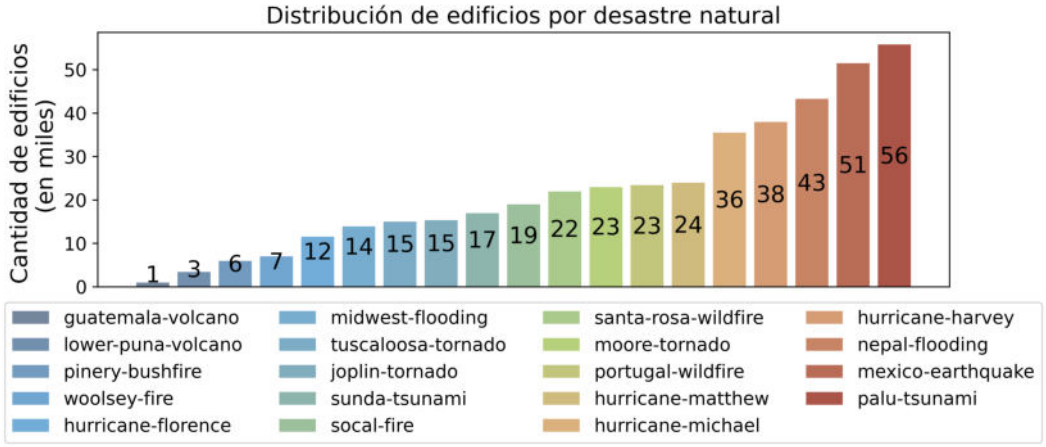


Figura 3.9: Cantidad de edificios por desastre natural en xBD.

3.4.0.3. Cantidad de edificios por imagen

La Tabla 3.4 muestra un resumen estadístico de la frecuencia que tiene la cantidad de edificios de cada clase que aparecen en cada parche. A partir de la tabla, se pueden extraer las siguientes conclusiones:

- Los valores máximos para las categorías son notablemente altos lo que sugiere la existencia de eventos aislados que han causado daños extremos.
- El promedio muestra que la presencia de edificios en las categorías *destroyed*, *major-damage* y *minor-damage* es significativamente menor que en la categoría *no-damage*, lo que sugiere que las imágenes tienden a cubrir áreas donde los edificios no han sufrido daños.
- La alta desviación estándar en las clases *destroyed* y *no-damage* indica una gran variabilidad en la cantidad de edificios de estas categorías en las imágenes, lo que sugiere que los edificios destruidos están concentrados en zonas específicas.
- Los percentiles revelan que el 75 % de las imágenes del conjunto de datos no presentan edificios clasificados como *destroyed*, *major-damage* o *minor-damage*.
- Además, se observa que los edificios clasificados como *un-classified* aparecen con mayor frecuencia que en las otras categorías, lo que podría indicar una dificultad en la clasificación en ciertas imágenes.

Complementando la Tabla 3.4 tenemos la Figura 3.10, la cual muestra un gráfico de torta que ilustra la distribución de los diferentes niveles de daño presentes en las imágenes del conjunto de datos xBD. En este gráfico se observa que el 35 % de las imágenes no contienen ningún edificio, mientras que el 65 % de ellas contienen al menos un edificio. Además, se observa que la probabilidad de que una imagen contenga edificios con varios tipos de daño disminuye conforme aumenta la cantidad de tipos de daño presentes.

	destroyed	**major-damage**	**minor-damage**	**no-damage**	**un-classified**
$\bar{x}$	2,86	2,71	3,34	28,37	1,27
s	20,12	14,34	18,81	80,09	6,63
min	0	0	0	0	0
Q_1	0	0	0	0	0
Q_2	0	0	0	1	0
Q_3	0	0	0	20	1
max	1540	441	618	1766	260

Tabla 3.4: Resumen estadístico de los niveles de daño en el conjunto de datos xBD. La tabla muestra el promedio ($\bar{x}$), desviación estándar (s), valores mínimos (min), percentiles (Q1, Q2, Q3) y valores máximos (max) para cada categoría de daño: *destroyed*, *major-damage*, *minor-damage*, *no-damage* y *un-classified*.

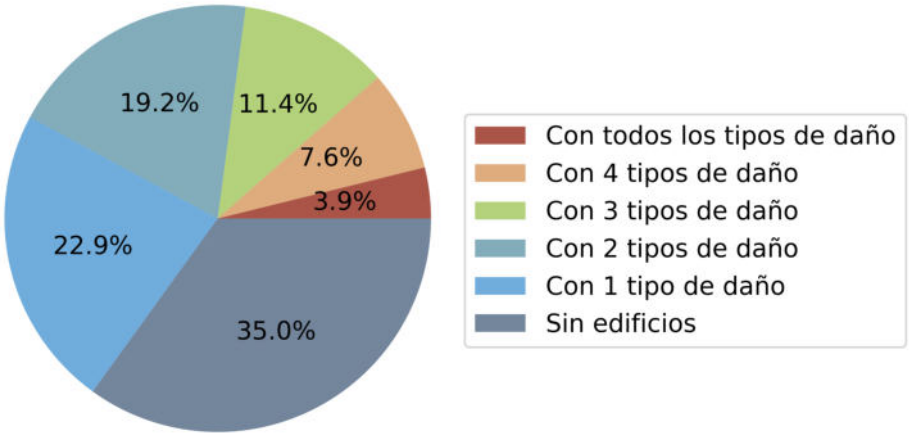


Figura 3.10: Gráfico de torta de la distribución de la cantidad de clases de daño en edificios de las imágenes de xBD.

3.4.0.4. xBD es un conjunto de datos desbalanceado

El análisis exploratorio evidencia que el conjunto de datos xBD presenta desbalance en las siguientes tres dimensiones:

- Hay desbalance en la cantidad de parches de cada tipo de desastre natural.
- Hay desproporción en la cantidad de parches de cada zona afectada por un desastre natural.
- Hay desproporción en la cantidad de edificios de cada tipo de daño presentes en el conjunto de datos. Este último es especialmente severo en algunas zonas de desastre natural como *mexico-earthquake* que especialmente solo tiene 3 edificios de tipo *destroyed* y 51084 edificios de tipo *no-damage*.

Al ser un conjunto de datos tan desbalanceado, esto produce dificultades durante el entrenamiento del modelo de manera significativa. Por lo tanto, para poder entrenar un modelo y que este sea capaz de generalizar correctamente, en esta tesis se realizarán pruebas con las siguientes tres técnicas para tratamiento del desbalance:

- Función de perdida con pesos para penalizar los errores en clases minoritarias.
- Aumento de imágenes con CutMix para tartar el desbalance de cantidad de edificios por imagen.
- Estrategia de muestreo greedy para seleccionar un conjunto de datos balanceado.

3.5. Preprocesamiento de los datos

1. Como primer paso es necesario generar las máscaras necesarias para utilizarlas como *target feature*. Es por eso que se crea un directorio `targets` que contiene las máscaras de segmentación de edificios y máscaras de clase para cada parche en el directorio IMAGES. Las máscaras son una representación

gráfica de los archivos etiquetados del directorio LABELS. En las máscaras correspondientes con las imágenes pre-desastre se representan los polígonos o *building footprints* de cada edificio. En las máscaras post-desastre son idénticas a la máscara pre-desastre, pero cada polígono está rellenado con un valor que indica el nivel de daño sufrido luego del desastre.

2. *Cálculo de estadísticas*: Durante el entrenamiento, antes de ingresar la imagen a la red se realiza una etapa de normalización de los colores de la imagen. Para realizar esta normalización es necesario saber el promedio y la desviación estándar de cada canal de colores de la imagen a normalizar, es por eso que en este paso se calculan estos valores para todas las imágenes del conjunto de datos y se almacenan para su posterior utilización.

3. *Muestreo estratégico greedy*: Si se habilita este método se encarga de obtener una muestra del conjunto completo que cumple con una cantidad n de imágenes determinada y presenta un balance entre la cantidad de edificios presentes por cada clase y el desbalance de la muestra. Este paso es importante para tratar el desbalance de clases.

4. *División del conjunto de datos*: En función de los desastres seleccionados para el entrenamiento y la cantidad total determinada de imágenes a utilizar, se divide el conjunto de datos xBD en dos subconjuntos: un conjunto de entrenamiento que contiene el 90 % de las imágenes y un conjunto de prueba que comprende el 10 % restante. No se crea la división como conjunto de validación debido a que se utiliza k-fold durante el entrenamiento. La forma de selección de las imágenes se realiza de manera aleatoria, a menos que se utiliza algún criterio de selección.

5. *Aplicar aumentación de datos*: Si se utiliza la aumentación de datos basada en CutMix el procedimiento se realiza en esta etapa. Esta etapa se utilizó durante el experimento 3, y consisten en a partir de un conjunto de imágenes generar nuevas imágenes que permitan balancear la cantidad de edificios de cada clase del conjunto de entrenamiento.

6. *Recortes de imágenes*: Debido a que la red admite recortes (*chips*) de resolución 256×256, en esta etapa, se recortan todas las imágenes de 1024×1024 en 16 partes. Esto resulta en la creación de un nuevo conjunto de entrenamiento y prueba compuestos por recortes.

7. *Cálculo de pesos*:Para utilizar la técnica de ponderación de clases, en esta tesina, se calculan los pesos para cada clase en función de la cantidad de píxeles presentes en todas las máscaras de clasificación de daños del conjunto de entrenamiento. Utilizando la Ecuación 2.13, se cuentan todos los píxeles de cada clase y se calculan los pesos correspondientes para la rama de clasificación como para la rama de segmentación.

3.5.1. Aumento de datos basado en CutMix

En este proyecto, se implementa una técnica inspirada en CutMix para sintetizar imágenes que equilibren la cantidad de edificios en cada clase del conjunto de entrenamiento. Dado que el enfoque original de CutMix recorta parches de forma aleatoria, no resulta eficaz para corregir el desbalance de datos, por lo que se diseñó un algoritmo adaptado al problema específico.

Este algoritmo garantiza que los edificios insertados mantengan coherencia contextual, evitando, por ejemplo, la inserción de edificios destruidos en zonas sin daños. La idea principal es generar nuevas imágenes que complementen las existentes y equilibren las clases. Sin embargo, debido a las características del problema, al pegar parches de una imagen sobre otra, inevitablemente se produce un sobremuestreo de la clase mayoritaria. Para mitigar este efecto, se optó por reemplazar edificios de manera individual, creando nuevas instancias sin alterar el balance de manera desproporcionada.

El algoritmo selecciona una imagen que contenga al menos un edificio de la clase subrepresentada y reemplaza un porcentaje de sus edificios con otros de la clase objetivo, tomados aleatoriamente de cualquier imagen del dataset. Este proceso puede presentar inconvenientes cuando el número total de edificios en una clase es muy bajo, lo que aumenta el riesgo de sobreajuste. Para prevenirlo, se aplican técnicas de aumento de datos a las imágenes sintetizadas, inspiradas en las utilizadas por ELEKS [11]. La Figura 3.11 muestra un ejemplo de una imagen generada con esta técnica.

Las transformaciones aplicadas son:

- Volteado aleatorio vertical
- Volteado aleatorio horizontal
- Recortado aleatorio (*Random croping*): Recorte aleatorio de 50x50 píxeles.
- Traslación de la imagen (*Image translation*): Desplazamiento aleatorio entre -5% y +5%
- Escalamiento aleatorio (*Random scaling*): Escalamiento aleatorio entre 0.9 y 1.2 en ambos ejes
- Rotación aleatoria (*Random rotation*): Rotación aleatoria entre -25 y +25 grados
- Modificación aleatoria de Hue y Saturación (*Random Hue and saturation*): Multiplicar los valores de matiz y saturación por un valor aleatorio entre (0.6, 1.4)
- Modificación aleatoria de contraste (*Random contrast modification*): Cambio aleatorio de contraste entre 0.75 y 1.25
- Modificación aleatoria de brillo (*Random brightness modification*): Multiplicar el brillo por un factor aleatorio entre y (0.75, 1.25).

3.5.2. Muestreo utilizando optimización con algoritmo voraz

Cabe destacar que la técnica propuesta en esta sección, basada en un enfoque voraz, se utiliza por primera vez en este trabajo, y no se tiene constancia de que haya sido aplicada al problema de CDEN en otros estudios de la literatura.

Figura 3.11: Imagen resultante de la sintetización de un parche. El identificador 10a00000059 indica que se trata de la décima imagen sintetizada, utilizando como base el parche original 00000059. En este caso, se agregaron edificios de la clase *destroyed*.

Las técnicas analizadas en la literatura para el tratamiento del desbalance de clases en el conjunto de datos xBD — o en sus subconjuntos — no suelen imponer una restricción sobre la cantidad exacta de imágenes resultantes en el conjunto utilizado para el entrenamiento [10, 11, 24].

Nuestra técnica permite establecer de manera precisa tanto el tamaño del conjunto de datos como la proporción de imágenes vacías a incluir. Además, asegura que la muestra de datos resultante contenga una cantidad de edificios de cada clase significativa, sin provocar desbalance severo entre las clases en el conjunto de datos final.

Encontrar una muestra de tamaño n que garantice una cantidad considerable de edificios y minimice el desbalance entre las clases es un problema de optimización que puede resolverse con un enfoque basado en optimización multiobjetivo o multicriterio. Para ello, a continuación se definen las variables de decisión y la función objetivo a optimizar.

Variables de decisión Dado $i = 1, \ldots, m$, donde m es el número total de clases en la muestra x, sea $S_i(x)$ el número total de edificios de la clase i en la muestra x de tamaño n, y $\bar{S}(x)$ el promedio de los valores $S_i(x)$. Las funciones $f_1(x)$ y $f_2(x)$ se definen como:

$$f_1(x) = \sum_{i=1}^{m} S_i(x) \tag{3.1}$$

$$f_2(x) = \frac{1}{n} \sum_{i=1}^{m} \left(S_i(x) - \bar{S}(x)\right)^2 \tag{3.2}$$

En este problema, las variables de decisión son $f_1(x)$, que representa la cantidad total de edificios en una muestra de imágenes (definida en la Ecuación 3.1), y $f_2(x)$, que mide el desbalance entre las clases presentes en la muestra (definida en la Ecuación 3.2).

Función objetivo El objetivo de este problema de optimización es encontrar una solución eficiente que maximice el criterio f_1 y minimice el criterio f_2. La Ecuación 3.3 define la función objetivo $g(x)$, que utiliza el parámetro α, entre 0 y 1, para ponderar ambos criterios. Este parámetro controla el balance entre maximizar la cantidad de edificios y minimizar el desbalance entre clases. Variando α, se pueden explorar diferentes soluciones. Un valor de $\alpha < 0{,}5$ prioriza la maximización del número de edificios, aunque incremente el desbalance, mientras que un $\alpha > 0{,}5$ favorece un mayor balance, incluso si reduce la cantidad de edificios.

$$\text{Maximizar } g(x) = \alpha f_1(x) - (1-\alpha) f_2(x) \tag{3.3}$$

Restricciones Dado que el espacio de soluciones de todas las posibles muestras que se pueden extraer del conjunto de datos xBD es amplio, se han implementado las siguientes restricciones. La primera restricción establece que el conjunto de datos resultante tenga un tamaño fijo n, elegido de manera arbitraría. La segunda restricción impone que la muestra debe estar conformada por una secuencia de imágenes consecutivas, seleccionadas de un conjunto de datos xBD previamente ordenado, utilizando un enfoque *greedy*, según su impacto en el desbalance.

3.5.2.1. Pasos del algoritmo de creación del conjunto de muestras

Paso 1: Separación de Datos El conjunto de datos xBD se divide en dos grupos: imágenes con edificios e imágenes sin edificios. Esta separación es necesaria, ya que el balance de clases se realiza primero en el grupo de imágenes con edificios. Debido a que las imágenes vacías no afectan el balance entre las clases y son necesarias para el entrenamiento del modelo, se almacenan para ser agregadas posteriormente.

Algoritmo 3.1 Algoritmo de ordenamiento greedy

Entrada: Conjunto de datos D
Salida: Conjunto de datos ordenado SD

```
1:  function MEJORCANDIDATO(SD, D)
2:      err ← ∅                                  ▷ Inicializar conjunto de errores
3:      Para todo d ∈ D hacer
4:          e ← MSE({d} ∪ SD)                    ▷ Calcular el error para d al agregarlo a SD
5:          err ← err ∪ {e}                      ▷ Guardar el error calculado
6:      return MIN(err)                          ▷ Devolver el candidato con el menor error
7:  SD ← ∅                                       ▷ Inicializar el conjunto de datos ordenado
8:  Mientras D ≠ ∅ hacer
9:      c ← MEJORCANDIDATO(SD, D)                ▷ Seleccionar el mejor candidato
10:     D ← D \ {c}                              ▷ Eliminar el candidato seleccionado de D
11:     SD ← SD ∪ {c}                            ▷ Agregar el candidato seleccionado a SD
12: return SD                                    ▷ Devolver el conjunto de datos ordenado
```

Paso 2: Ordenamiento de Imágenes Para ordenar el conjunto de datos, se utilizó un algoritmo iterativo inspirado en un enfoque voraz (*greedy*), que selecciona en cada paso la imagen que menos afecte el balance entre clases, es decir, aquella que minimiza el valor de $f_2(x)$. El proceso construye un conjunto de imágenes agregando una instancia por vez y evaluando el desbalance que esta incorporación produce. Como resultado, se obtiene un conjunto de imágenes ordenado de manera tal que, al seleccionar n imágenes consecutivas, el conjunto suele estar balanceado. Sin embargo, esto no garantiza un balance perfecto para todas las muestras, si se selecciona todas las imágenes, la muestra resultante tendrá el desbalance del conjunto original. Esta estrategia de ordenamiento *greedy* se describe con pseudocódigo del Algoritmo 3.1.

Paso 3: Optimización Multiobjetivo Para definir la función objetivo $g(x)$, es necesario establecer las funciones $f_1(x)$ y $f_2(x)$. Estas funciones se obtienen al evaluar las ecuaciones Ecuación 3.1 y Ecuación 3.2 sobre el dominio de muestras x, que consiste en todas las muestras de parches consecutivos en la secuencia ordenada, generadas mediante un enfoque voraz de tamaño n.

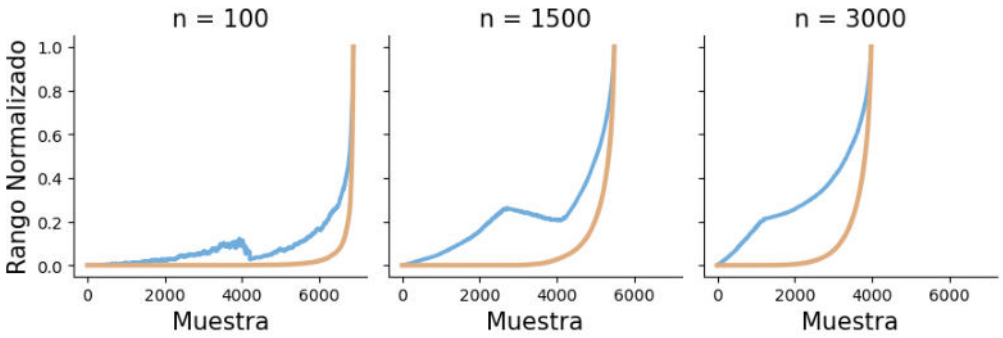


Figura 3.12: Gráfico que muestra las curvas de los criterios $y = f_1(x)$ (en azul) y $y = f_2(x)$ (en marrón), donde el eje x representa una muestra de tamaño n obtenida al seleccionar de manera consecutiva parches del conjunto de datos xBD, utilizando un enfoque voraz. El eje y indica el resultado de la función, normalizado entre 0 y 1.

La Figura 3.12 destaca una característica importante de la secuencia ordenada de muestras de tamaño n. Las muestras compuestas por parches consecutivos al inicio de esta secuencia presentan una menor cantidad de edificios, pero exhiben un desbalance casi perfecto. Por otro lado, al seleccionar muestras de parches consecutivos ubicados al final de la secuencia ordenada, se observa que estas contienen una gran cantidad de edificios de todas las clases. Sin embargo, presentan un desbalance severo.

Entendiendo las propiedades que posee la secuencia de parches ordenada y sabiendo que ambos criterios de selección son contra puestos entre sí, podemos deducir que la solución eficiente al problema es aquella muestra que maximiza la función objetivo $g(x)$. Para encontrar la solución simplemente seleccionamos aquel candidato que maximiza la función de composición de la Ecuación 3.3.

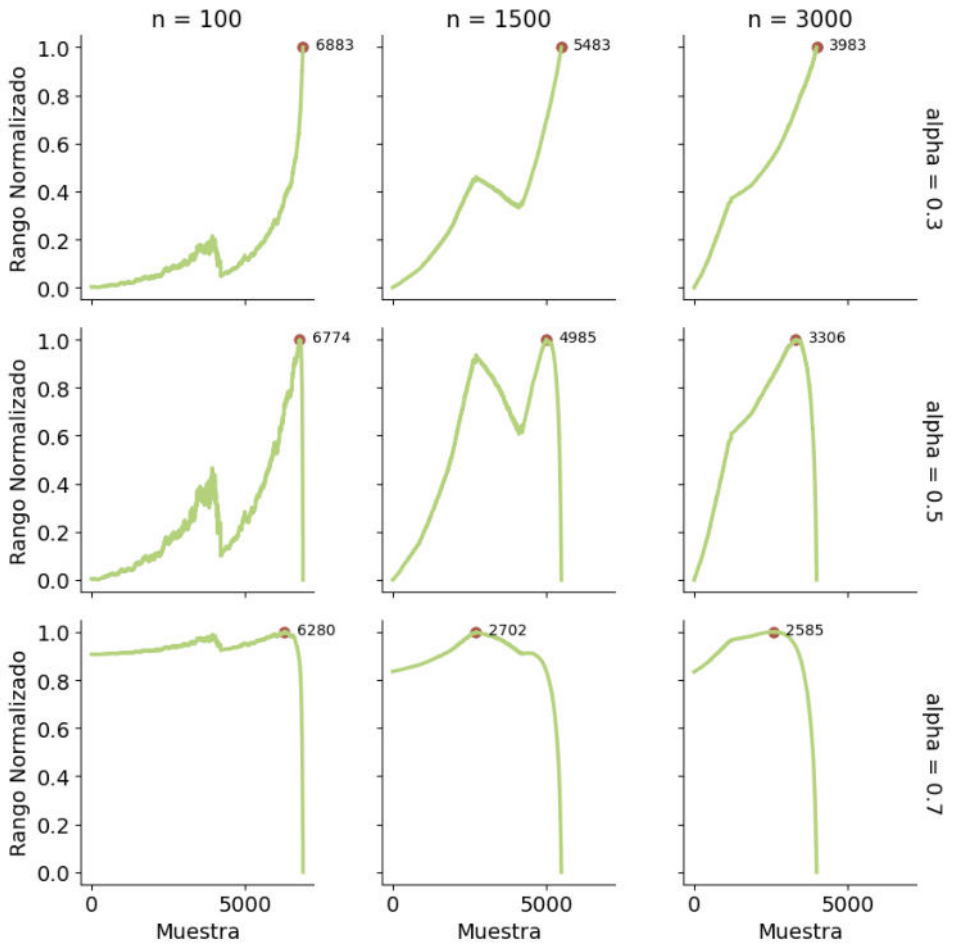


Figura 3.13: Variación de la función objetivo para diferentes tamaños de muestra n y valores de α. En marrón se marca el máximo absoluto de la función y el índice de la muestra seleccionada como solución.

La función objetivo $g(x)$ incluye un parámetro $\alpha \in [0,1]$ que pondera la importancia de los criterios f_1 y f_2. Al ajustar el valor de α, la muestra que maximiza la función objetivo presenta las siguientes características:

- Cuando $\alpha = 0$, se ignora el criterio f_2 y la muestra resultante se selecciona del final de la secuencia de muestras ordenadas. Estas contienen una gran cantidad de edificios, pero presentan un considerable desbalance entre sus clases.
- Cuando $\alpha \to 0$, las muestras resultantes tienen cada vez más edificios, mientras que el desbalance aumenta consecuentemente.
- Cuando $\alpha = 0{,}5$, ambos criterios son igualmente importantes, generando una muestra que mantiene un equilibrio entre el desbalance de clases y la cantidad de edificios, probablemente cerca del centro de la secuencia de parches ordenados.
- Cuando $\alpha \to 1$, las muestras contienen cada vez menos edificios, pero logran un mayor balance entre sus clases.

- Cuando $\alpha = 1$, se ignora el criterio f_1 y la muestra se selecciona del inicio de la secuencia, logrando un balance perfecto entre las clases, aunque con una cantidad muy reducida de edificios.

En al Figura 3.13, se puede apreciar como varía la función objetivo para diferentes tamaños de muestra n y cuál es la solución eficiente para cada caso según se modifique α. En esta tesis el objetivo es obtener un conjunto de datos balanceado con la mayor cantidad de edificios posible, es decir, que se está dispuesto a ceder parcialmente el balance del conjunto de datos, a costa de obtener conjuntos de datos con más edificios. Es por eso que, a partir de la Figura 3.13, se seleccionó $\alpha = 0,7$. Este valor prioriza la cantidad de edificios de la muestra por encima de su balance, pero garantiza que el desbalance del conjunto de datos no sea tan severo como para dificultar el entrenamiento del modelo.

Paso 4: Agregar imágenes vacías Las imágenes sin edificios son fundamentales en la tarea de aprendizaje del modelo para las tareas de CDEN, ya que permiten al modelo distinguir adecuadamente entre los elementos del fondo y los edificios. El análisis del conjunto de datos xBD ha revelado que el 35 % de los parches no contienen edificios de ningún tipo.

En esta tesis, al trabajar con un número reducido de parches, se considera que mantener un 35 % de imágenes sin edificios no aporta ventajas significativas al proceso de aprendizaje. Se ha establecido que un 20 % de parches vacíos es un valor más razonable, el cual ha demostrado, a lo largo de los experimentos, ser suficiente para que el modelo logre un buen desempeño en la tarea de segmentación de edificios.

3.6. Entrenamiento y validación del modelo

Para asegurar un entrenamiento robusto del modelo, se realizó una búsqueda de hiperparámetros mediante k-fold en cada experimento. El conjunto de datos se dividió únicamente en conjunto de entrenamiento y conjunto de prueba, ya que el conjunto de validación se extrae de los *folds* en cada iteración.

La búsqueda de hiperparámetros se llevó a cabo utilizando *grid search*, evaluando la tasa de aprendizaje (*learning rate*), el tamaño del lote (*batch size*) y la cantidad de épocas *epochs*. Durante el entrenamiento, se implementó una técnica de *checkpoints* para guardar periódicamente los pesos del modelo. Esta estrategia permite aplicar *early stopping* de manera efectiva, manteniendo constante el número de épocas y asegurando la selección del mejor modelo.

Con el fin de realizar una exploración más profunda de como afecta el factor de aprendizaje el desempeño del modelo, se implementó la estrategia de reducción del factor de aprendizaje durante el entrenamiento *ReduceOnPlanteu* (ROPL). Esta técnica es aplicada en [45] y suele mejorar el rendimiento. En los experimentos, se evaluó diferentes valores de paciencia (*patience*) para que el planificador ajuste el factor de aprendizaje cuando no se observan mejoras significativas en la función de pérdida del conjunto de validación.

3.6.1. Evaluación de rendimiento

Para la evaluación del rendimiento del modelo se utilizó las métricas Precisión (P, *Precision*), Sensibilidad (R, *Recall*), Exactitud (ACC, *Accuracy*), Puntaje F1 (F1, *F1-score*). Estas métricas se calculan a nivel de píxel, tanto para la máscara de segmentación de edificios como para cada clase en la máscara de clasificación de daños. Con el fin de obtener una única métrica que refleje el rendimiento general de la clasificación, se calcula además la Media Armónica F1 (HMF1, *Harmonic Mean f1-score*) para todas las clases. Esta métrica tiene en cuenta el F1 de cada clase, así como los valores de Sensibilidad (R, *Recall*) y Precisión (P, *Precision*), proporcionando una evaluación integral del rendimiento de la clasificación. Es importante destacar que la selección de la mejor época de un modelo se basa en el valor de la HMF1 sobre el conjunto de validación.

3.6.2. Criterio de selección de la arquitectura

Luego del análisis del estado del arte y los trabajos relacionados con CDEN en esta tesina final de grado, se decidió optar por utilizar un modelo de DL basado en la arquitectura U-Net Siamesa presentada por Hao et al. [9]. Esto se resolvió teniendo en cuenta que el modelo realiza aprendizaje multitarea aprendiendo sobre detección de los edificios y la clasificación de sus daños en una única etapa de entrenamiento, lo que facilita su implementación.

En la arquitectura implementada en esta tesina, se decidió no incluir el mecanismo de atención propuesto en Hao et al. [9]. Esta decisión se tomó debido a la incertidumbre sobre el posible incremento significativo en el tiempo de cómputo que dicho mecanismo podría requerir, especialmente en comparación con las limitaciones de hardware disponibles para el entrenamiento de la red. Como fuente de inspiración y guía se utilizó el código base el repositorio de código de Caleb Robinson [45] en el cual, se implementa el entrenamiento de una arquitectura similar.

3.6.3. Características de la Arquitectura

La arquitectura seleccionada es una SCNN basada en *U-Net* y consta de tres ramas principales, cada una compuesta por un *encoder*, un *bottleneck* y un *decoder*, como puede observarse en Figura 3.15.

La arquitectura recibe como entrada dos imágenes de resolución 256×256 y genera como salida una máscara de segmentación por cada una de las ramas. Tanto la rama de segmentación de edificios pre-desastre como la de segmentación post-desastre comparten pesos en todas sus capas y generan una máscara con las clases de salida *background* y *building*. Por otro lado, la rama de clasificación únicamente cuenta con un *decoder* cuya salida es una máscara de clasificación con las clases *background*, *no-damage*, *minor-damage*, *major-damage* y *destroyed*.

En la Figura 3.15, cada prisma de color naranja claro en el diagrama representa un bloque básico, como el que se ilustra en la Figura 3.14, y los prismas de color naranja oscuro representan capas de *max pooling* que realizan el submuestreo de la imagen. Las líneas de color azul representan las *skip-layer connections* que son la concatenación entre la salida de cada capa del *encoder* con su correspondiente en el

decoder. Por último, las conexiones negro en la Figura 3.15 representan la operación de sustracción que se realiza entre dos salidas, cuyo resultado es luego concatenado a las capas en el *decoder* de la rama de clasificación de daños. La secuencia de bloques se describe detalladamente en la Sección 3.6.3.

La arquitectura está construida a partir de bloques básicos, ilustrados en la Figura 3.14, que están compuestos por dos secuencias de tres capas. La primera es una capa convolucional con un filtro de 3×3, su salida ingresa a una capa de normalización en lotes y finalmente se aplica una la función de activación ReLU. Para el entrenamiento del modelo de DL, tomando como referencia la implementación de Caleb Robinson [45], se utilizó el optimizador *Adam*, el inicializador de pesos *Xavier Uniform* y la función de perdida *Cross Entropy Loss* tanto para la salida de segmentación como de clasificación.

3.7. Arquitectura del modelo

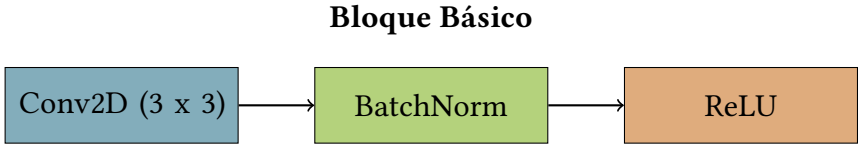


Figura 3.14: Secuencia de capas que componen el bloque básico utilizado para construir la arquitectura del modelo SCNN.

3.7.1. Recursos de Hardware utilizados

En este proyecto se utiliza el Cluster Toko de FCEN-UNCuyo que forma parte del SNCAD-MinCyT, Argentina [1]. En este Cluster se realizaron los primeros experimentos y pruebas para implementación de preprocesamiento. El nodo utilizado consta de 64 núcleos de procesadores AMD Opteron/Epyc y 128 GB (toko01, 05, 06) de memoria RAM sin GPU.

Luego de los primeros experimentos se migró la computación al CCAD de la Universidad Nacional de Córdoba [2], que forma parte del SNCAD del MinCyT de la República Argentina. Esto se realizó considerando ya el CCAD cuenta con GPU disponible. El nodo empleado está equipado con dos procesadores Intel Xeon E5-2680 v2 de 10 núcleos cada uno, 64 GB de memoria RAM DDR3 y dos GPU NVIDIA A30, cada una con 24 GB de memoria de video.

3.8. Postprocesamiento de resultados

El objetivo de la etapa de postprocesamiento es combinar todos los parches predichos que corresponden a imágenes de 1024x1024, de manera que puedan ser mostrados en la página web implementada para esta tesis. Además, se generan cajas delimitadoras (*bounding boxes*) que contienen los edificios detectados, permitiendo realizar el conteo de estos, como se ejemplifica en la Figura 3.16.

[1] https://toko.uncu.edu.ar/
[2] https://ccad.unc.edu.ar/

Parte	Nombre de la Capa	Tamaño de entrada	Tamaño de kernel
Rama de segmentación pre y post desastre			
encoder	1.1 Bloque básico	$3 \times 256 \times 256$	3×3 *padding*=1
	1.2 Bloque básico	$16 \times 256 \times 256$	3×3 *padding*=1
	1 Capa *Max-pooling*	$16 \times 256 \times 256$	2×2 *stride*=2
	2.1 Bloque básico	$16 \times 128 \times 128$	3×3 *padding*=1
	2.2 Bloque básico	$32 \times 128 \times 128$	3×3 *padding*=1
	2 Capa *Max-pooling*	$32 \times 128 \times 128$	2×2 *stride*=2
	3.1 Bloque básico	$32 \times 64 \times 64$	3×3 *padding*=1
	3.2 Bloque básico	$64 \times 64 \times 64$	3×3 *padding*=1
	3 Capa *Max-pooling*	$64 \times 64 \times 64$	2×2 *stride*=2
	4.1 Bloque básico	$64 \times 32 \times 32$	3×3 *padding*=1
	4.2 Bloque básico	$128 \times 32 \times 32$	3×3 *padding*=1
	4 Capa *Max-pooling*	$128 \times 32 \times 32$	2×2 *stride*=2
bottleneck	5.1 Bloque básico	$128 \times 16 \times 16$	3×3 *padding*=1
	5.2 Bloque básico	$256 \times 16 \times 16$	3×3 *padding*=1
decoder	4 Capa *unpooling*	$256 \times 16 \times 16$	2×2 *stride*=2
	4.1 Bloque básico	$256 \times 32 \times 32$	3×3 *padding*=1
	4.2 Bloque básico	$128 \times 32 \times 32$	3×3 *padding*=1
	3 Capa *unpooling*	$128 \times 32 \times 32$	2×2 *stride*=2
	3.1 Bloque básico	$128 \times 64 \times 64$	3×3 *padding*=1
	3.2 Bloque básico	$64 \times 64 \times 64$	3×3 *padding*=1
	2 Capa *unpooling*	$64 \times 64 \times 64$	2×2 *stride*=2
	2.1 Bloque básico	$64 \times 128 \times 128$	3×3 *padding*=1
	2.2 Bloque básico	$16 \times 128 \times 128$	3×3 *padding*=1
	1 Capa *unpooling*	$32 \times 128 \times 128$	2×2 *stride*=2
	1.1 Bloque básico	$32 \times 256 \times 256$	3×3 *padding*=1
	1.2 Bloque básico	$16 \times 256 \times 256$	3×3 *padding*=1
salida	Capa convolucional	$16 \times 256 \times 256$	1×1
	Capa Softmax	$2 \times 256 \times 256$	-
Rama de clasificación de daños			
decoder	6 Capa *unpooling*	$256 \times 16 \times 16$	2×2 *stride*=2
	6.1 Bloque básico	$256 \times 32 \times 32$	3×3 *padding*=1
	6.2 Bloque básico	$128 \times 32 \times 32$	3×3 *padding*=1
	7 Capa *unpooling*	$128 \times 32 \times 32$	2×2 *stride*=2
	7.1 Bloque básico	$128 \times 64 \times 64$	3×3 *padding*=1
	7.2 Bloque básico	$64 \times 64 \times 64$	3×3 *padding*=1
	8 Capa *unpooling*	$64 \times 64 \times 64$	2×2 *stride*=2
	8.1 Bloque básico	$64 \times 128 \times 128$	3×3 *padding*=1
	8.2 Bloque básico	$16 \times 128 \times 128$	3×3 *padding*=1
	9 Capa *unpooling*	$32 \times 128 \times 128$	2×2 *stride*=2
	9.1 Bloque básico	$32 \times 256 \times 256$	3×3 *padding*=1
	9.2 Bloque básico	$16 \times 256 \times 256$	3×3 *padding*=1
salida	Capa convolucional	$16 \times 256 \times 256$	1×1
	Capa Softmax	$5 \times 256 \times 256$	-

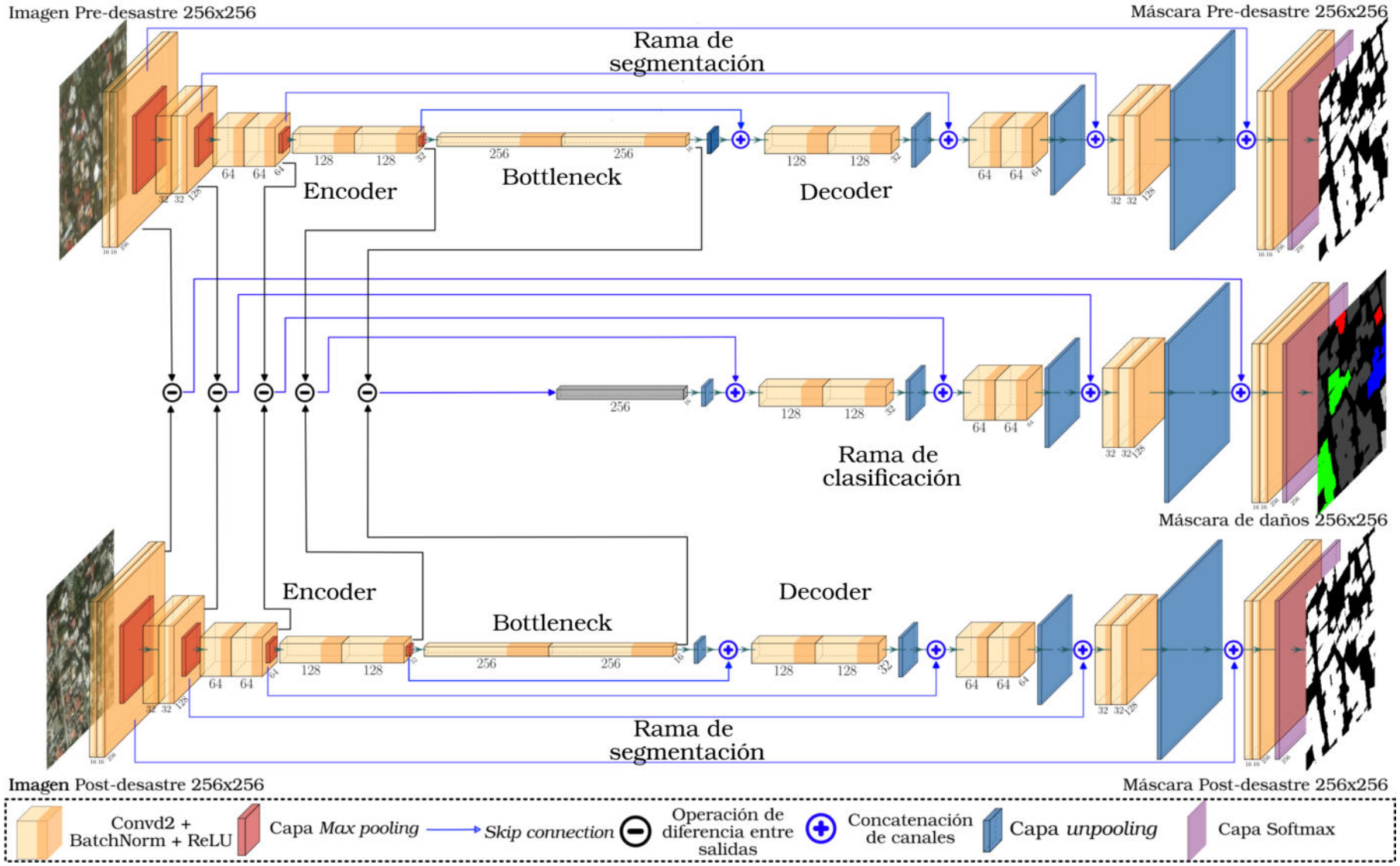


Figura 3.15: Diagrama representativo de la arquitectura del modelo.

Cantidad real de edificios

	Level	Count
0	no-damage	11
1	minor-damage	0
2	major-damage	0
3	destroyed	2

Cantidad predicha de edificios

	Level	Count
0	no-damage	12
1	minor-damage	6
2	major-damage	3
3	destroyed	3

Figura 3.16: Salida de la etapa de postprocesamiento para el parche 731 del desastre *socal-fire*.

3.9. Herramientas y tecnologías de implementación

En esta sección se describen las herramientas y tecnologías empleadas en el desarrollo del sistema de detección, así como en el entrenamiento y evaluación de la red neuronal. Estas herramientas fueron seleccionadas por su capacidad para cumplir con los requisitos del proyecto y asegurar una implementación eficiente.

El procesamiento de imágenes y el uso de la red neuronal se realizaron con bibliotecas y *frameworks* especializados. *Python*[1] fue seleccionado por su amplia adopción y la disponibilidad de bibliotecas avanzadas para ML. Además, se empleó el administrador de entornos *conda* y el gestor de paquetes *pip*.

Para la implementación y entrenamiento del modelo de DL se utilizaron las siguientes herramientas:

- **Visual Studio Code** fue el entorno principal para codificación y depuración.
- **Conda** facilitó la gestión de dependencias y configuraciones de Python, permitiendo la creación de entornos virtuales y la instalación de paquetes necesarios.
- **Jupyter Notebooks**: Herramienta para la escritura y ejecución interactiva de código.

- El lenguaje de programación es Python (Versión 3.11.9) y sus librerías son:

[1] https://www.python.org/

- **Pip (Versión 23.3.1)**: Administrador de paquetes de python instalado con conda.
- **pyTorch (Versión 2.4.0)**: Utilizado para construir y entrenar la red neuronal.
- **torchvision (Versión 0.19.0)**: Proporciona herramientas y datasets para tareas de visión por computadora.
- **opencv-python (Versión 4.10.0.84)**: Utilizado para tareas de procesamiento de imágenes.
- **scipy (Versión 1.14.1)**: Algoritmo de optimización húngara para métricas a nivel de objeto.
- **scikit-learn (Versión 1.5.1)**: Para tareas adicionales de aprendizaje automático y métricas.
- **numpy (Versión 1.26.4)**: Biblioteca fundamental para el cálculo numérico.
- **pandas (Versión 2.2.2)**: Para la manipulación de datos tabulares.
- **matplotlib (Versión 3.8.4)**: Utilizado para la visualización de datos y resultados.
- **seaborn (Versión 0.13.2)**: Para la visualización estadística de datos.
- **shapely (Versión 2.0.5)**: Utilizado en la generación de máscaras.
- **rasterio (Versión 1.3.10)**: Para la manipulación de datos rasterizados a nivel de objetos.
- **tensorboard (Versión 2.17.1)**: Herramienta para la visualización de métricas durante el entrenamiento.
- **tqdm (Versión 4.66.5)**: Para la visualización de barras de progreso.
- **flake8 (Versión 7.0.0)**: Para asegurar la calidad y estilo del código.

Para la página web, se utilizaron lenguajes como **HTML**, **JavaScript** y Python. Además, se empleó el framework **Flask**, que facilita el desarrollo de aplicaciones web y el fácil despliegue de servidores. Finalmente, para el estilo visual de la página se utilizó **css** y **Boostrap**.

3.10. Sistema prototipo

El sistema prototipo consiste de una página web y ofrece a los usuarios la posibilidad de subir imágenes satelitales para evaluar el rendimiento del modelo entrenado y visualizar su desempeño. Para ello simplemente se seleccionan imágenes almacenadas de manera local y se presiona en el botón de procesar imágenes. El prototipo está diseñado para facilitar a los usuarios la comparación de sus propias imágenes seleccionadas, proporcionando así una experiencia interactiva y funcional. En la Figura 3.17 se puede ver la interfaz con la que el usuario interactúa.

Figura 3.17: Prototipo de página web implementada para brindar acceso a los usuarios a utilizar el modelo de ML.

El repositorio con el código de entrenamiento y la aplicación web se encuentra disponible en gitfront [1].

En la Figura 3.18 se puede ver el resultado de presionar los botones del lado derecho superior que permiten alternar entre el idioma español e inglés y la posibilidad de cambiar el color de la página a un tema claro u obscuro. El logo de la facultad redirige a la página de la institución y el logo de github [63] al repositorio remoto para poder ver el código tanto del entrenamiento del modelo como de la implementación de la página.

La página permite ver las imágenes en pantalla completa al hacer clic sobre la imagen previa posterior o la máscara de clases predicha, como se ve en la figura Figura 3.19.

Al pasar el cursor por encima de las imágenes de la página se muestra una pequeña caja gris que indica la región que se está mostrando en alguna de las 3 cajas según corresponde. La funcionalidad de estas cajas es poder visualizar un mismo edificio en las tres imágenes de manera simultánea. Como se muestra en la Figura 3.20. La tabla inferior izquierda en la página permite ajustar la visibilidad de las cajas flotantes que se muestran sobre la máscara de clases. Por último, la tabla de la derecha muestra un conteo aproximado de la cantidad de edificios de cada clase detectados por el modelo en la máscara de clases.

[1] https://gitfront.io/r/Mrtc101/k4gzPrpN1AZB/Thesis-DL-for-BDA/

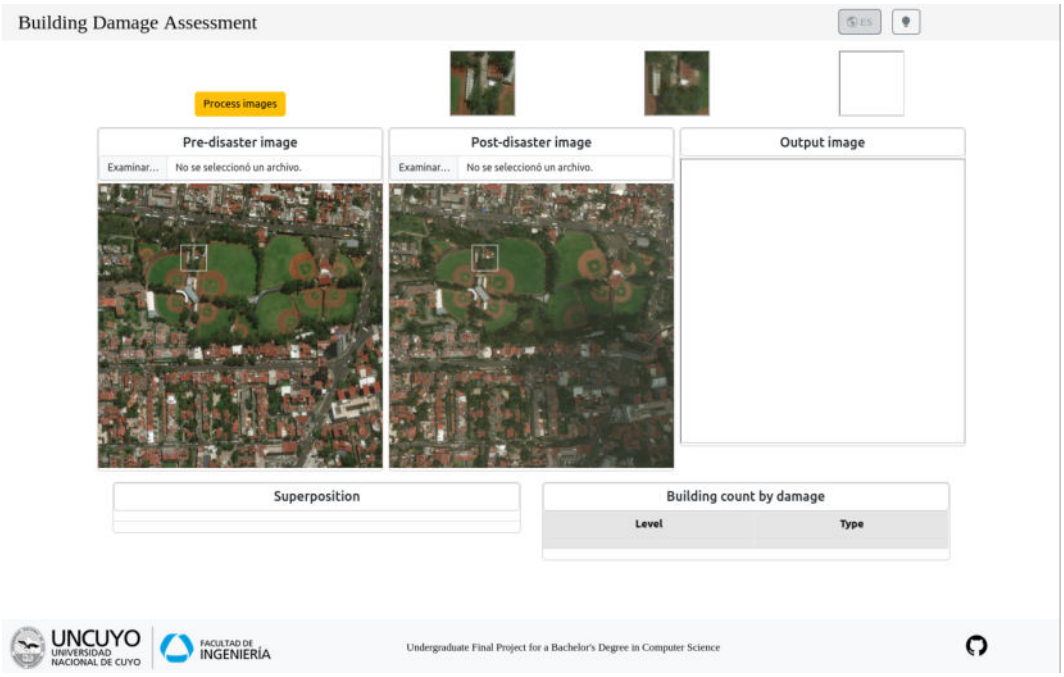


Figura 3.18: Aspecto de la página al cambiar el idioma a inglés y el tema obscuro por claro.

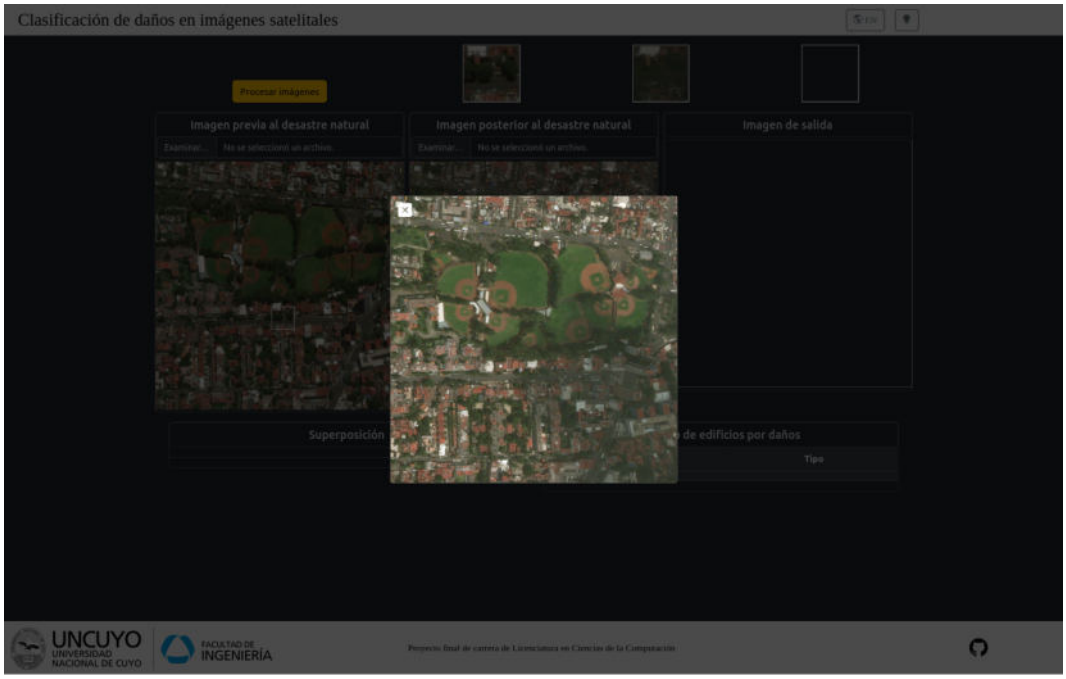


Figura 3.19: Imagen en pantalla completa

Figura 3.20: Página luego de procesar la imagen de entrada.

Capítulo 4

Diseño Experimental y Resultados

En esta sección se presentan los experimentos realizados en esta tesis, acompañados de sus resultados y un análisis detallado.

Secciones del capitulo

4.1. Experimento 1: Entrenamiento con pesos en la función de perdida

El objetivo de este primer experimento (EXP 1) es determinar el desempeño del modelo entrenado usando la técnica de ponderación de clases, una de las técnicas explicadas en la Sección 2.3.7, que utiliza pesos en la función de perdida para tratar con el desbalance de clases en el conjunto de datos.

4.1.1. Descripción del conjunto de entrenamiento

Para el análisis del modelo propuesto en esta tesina se utilizaron en el primer experimento **193** parches pertenecientes al desastre natural *mexico-earthquake* y se dividieron como se muestra en la Tabla 4.1.

	Entrenamiento	**Validación**	**Prueba**	**Total**
Cantidad	152	19	22	**193**

Tabla 4.1: Cantidad de parches en cada subconjunto de datos del experimento 1.

En la Tabla 4.2 se presenta la distribución de edificios por clase en un total de 193 parches. Cabe destacar que el desastre natural *mexico-earthquake* exhibe un marcado desbalance, con tan solo tres edificios clasificados como *destroyed*.

	destroyed	*major-damage*	*minor-damage*	*no-damage*	*un-classified*	**Total**
mexico-earthquake	3	54	221	51 084	111	**51 473**

Tabla 4.2: Cantidad de edificios de cada clase presentes en los 193 parches del desastre *mexico-earthquake*.

4.1.1.1. Asignación de Pesos para Entrenamiento

En la Tabla 4.3 se detallan los pesos asignados para el entrenamiento del modelo. Es importante señalar que los pesos correspondientes a la clase *background* son idénticos en ambas ramas, lo cual se debe a que estos fueron calculados considerando la proporción de píxeles de cada clase presentes en la máscara de clasificación.

4.1.2. Búsqueda y Optimización de Hiperparámetros

Las configuraciones de la Figura 4.2 empleadas en la exploración de hiperparámetros fueron establecidas mediante el método de búsqueda en grilla, basándose en el dominio de hiperparámetros descrito en la Figura 4.1.

Rama	Clase	Peso
Segmentación	*background*	1,92
	building	2,08
Clasificación	*background*	1,92
	no-damage	2,22
	minor-damage	43,99
	major-damage	160,10
	destroyed	2407,35

Tabla 4.3: Pesos calculados en función del desbalance entre píxeles de cada clase en el experimento 1.

Hiperparámetros	Valores
Tamaño de batch:	{16; 32; 64}
Factor de aprendizaje:	{0,001; 0,0001}
Total de épocas:	100

Figura 4.1: Hiperparámetros explorados en el experimento 1.

Conf. N°	BS	LR	Épocas Totales
C0	16	0,001	100
C1	16	0,0001	100
C2	32	0,001	100
C3	32	0,0001	100
C4	64	0,001	100
C5	64	0,0001	100

Figura 4.2: Configuraciones de entrenamiento del experimento 1.

Ejecución	Tiempo de ejecución
Preproc.	00:03:06
C0	08:07:28
C1	08:16:47
C2	07:53:35
C3	08:05:05
C4	07:54:51
C5	07:52:28
Entr. EXP1_C0	01:49:42
Postproc.	00:01:51

Tabla 4.4: Tiempo de ejecución, en (hh:mm:ss), transcurrido para cada parte del experimento 1 en el cluster Mendieta.

4.1.2.1. Tiempo de ejecución

En la Tabla 4.4 se detalla el tiempo de ejecución transcurrido para cada etapa del entrenamiento. Podemos notar que el entrenamiento con cada configuración tarda alrededor de las ocho horas, mientras que el entrenamiento del modelo EXP1_C0 solo dos. Esto se debe a que para probar cada configuración se realizó una validación cruzada con cinco pliegues, es decir, que se entrenó el modelo cinco veces, para obtener una estimación del desempeño más robusta.

4.1.2.2. Selección de la mejor configuración

La Tabla 4.5 resume los resultados obtenidos de la búsqueda de hiperparámetros. La configuración seleccionada para el entrenamiento del modelo EXP1_C0, utilizando todo el conjunto de datos de entrenamiento, es aquella que alcanzó el valor más alto de la métrica HMF1.

Como se puede observar, en este experimento la configuración seleccionada es la número 0.

Config.	*Fold*	Mejor época	Perdida en validación	HMF1 de Clasificación	HMF1 de Segmentación
C0	F0	42	0,1257	0,0000	0,8612
	F1	56	0,1329	0,0000	0,8506
	F2	99	0,1446	**0,0158**	0,8504
	F3	41	0,1269	0,0000	0,8630
	F4	71	0,1350	0,0000	**0,8638**
C1	F0	43	0,1330	0,0000	0,8446
	F1	11	0,1423	0,0037	0,8391
	F2	5	0,1402	0,0004	0,8255
	F3	6	0,1416	0,0000	0,7750
	F4	6	0,1463	0,0000	0,7914
C2	F0	78	0,0695	0,0000	0,8646
	F1	14	0,0698	0,0010	0,8268
	F2	44	0,0680	0,0000	0,8547
	F3	28	0,0673	0,0000	0,8442
	F4	91	0,0690	0,0000	0,8545
C3	F0	1	0,0808	0,0005	0,7774
	F1	48	0,0719	0,0028	0,8295
	F2	54	0,0699	0,0000	0,8180
	F3	72	0,0713	0,0000	0,8192
	F4	19	0,0739	0,0000	0,8464
C4	F0	99	0,0371	0,0049	0,8509
	F1	80	**0,0342**	0,0000	0,8504
	F2	55	0,0346	0,0000	0,8576
	F3	2	0,0377	0,0000	0,8011
	F4	1	0,0424	0,0001	0,7132
C5	F0	88	0,0376	0,0036	0,8098
	F1	78	0,0372	0,0000	0,8033
	F2	26	0,0366	0,0000	0,8353
	F3	64	0,0372	0,0000	0,8272
	F4	10	0,0377	0,0000	0,7892

Tabla 4.5: Resumen de la mejor época lograda durante el entrenamiento de cada *fold* con cada configuración del experimento 1.

4.1.3. Desempeño del modelo EXP1_C0

El modelo EXP1_C0 fue entrenado durante 100 épocas con un tamaño de lote 16 y un factor de aprendizaje de 0,001 (EXP1_C0).

La Figura 4.3, muestra la evolución de la función de pérdida durante el entrenamiento, se observa que la pendiente de la curva de pérdida sobre el conjunto de validación es menos pronunciada y disminuye menos que la del conjunto de entrenamiento a lo largo de las 100 épocas. Esto sugiere un posible sobreajuste.

Este sobreajuste se evidencia con la Figura 4.4, que muestra que la métrica HMF1 sobre el conjunto de validación no supera 0 durante todo el entrenamiento. La falta de mejora en HMF1 para el conjunto de validación se explica por la Figura 4.5, donde la métrica F1 de las clases minoritarias no presenta mejora significativas a lo largo del entrenamiento.

4.1.3.1. Desempeño en la mejor época

En la Tabla 4.6 se presentan las métricas de desempeño del modelo EXP1_C0 obtenido en la época 98. Esta época es aquella en la que el valor de la métrica HMF1 fue más alto. En esta tabla, se observa que la métrica HMF1 para la tarea de clasificación es 0. Esto se debe a que dicha métrica se ve fuertemente penalizada cuando el modelo no logra aprender correctamente sobre alguna clase, resultando en un valor cercano a 0 en la métrica F1 para esas clases.

En este caso en particular el modelo no fue capaz de aprender sobre las clases minoritarias de daño *minor-damage*, *major-damage* y *destroyed*. Este comportamiento es consecuencia de la escasez de edificios pertenecientes a estas clases en los conjuntos de entrenamiento y validación. Sin embargo, se observa que el modelo, tras 100 épocas de entrenamiento, ha logrado cierto aprendizaje en la clasificación de edificios sin daños y en la correcta clasificación de los píxeles correspondientes al fondo de las imágenes.

Adicionalmente, debido a que las métricas se evaluaron usando un enfoque uno contra todos, el valor de la métrica ACC es alto en todas las clases. Esto ocurre porque la cantidad de verdaderos negativos para cada clase incluye la mayoría de los píxeles de las demás clases, inflando artificialmente esta métrica y limitando su capacidad para reflejar el verdadero rendimiento del modelo.

4.1.4. Discusión

Los resultados de este experimento han permitido obtener las siguientes conclusiones:

- El desbalance en la cantidad de edificios que presentan cada tipo de daño dificulta considerablemente la capacidad del modelo para aprender sobre las clases minoritarias.

- La selección de muestras en el momento de realizar la división en el conjunto de entrenamiento, validación y prueba debe utilizar alguna estrategia para garantizar que haya edificios de todas las clases en los diferentes conjuntos, de otra manera la métrica HMF1 podría no aumentar de 0 al evaluar el modelo

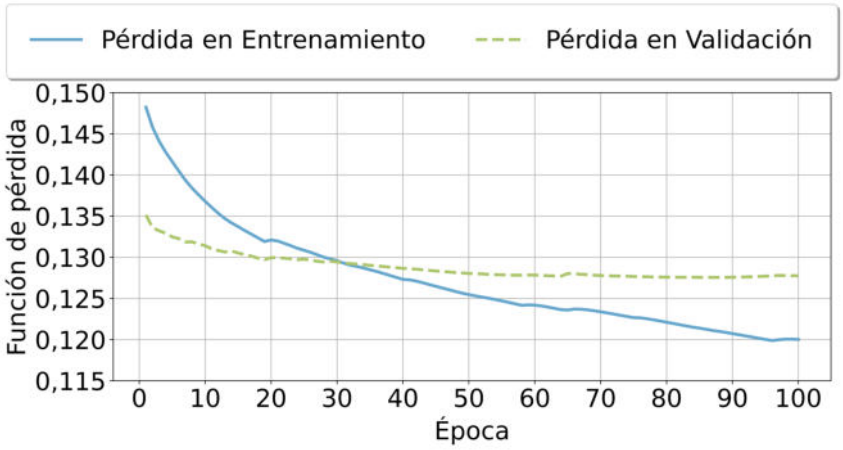


Figura 4.3: Evolución de la función de pérdida sobre el conjunto de entrenamiento y validación durante el experimento 1.

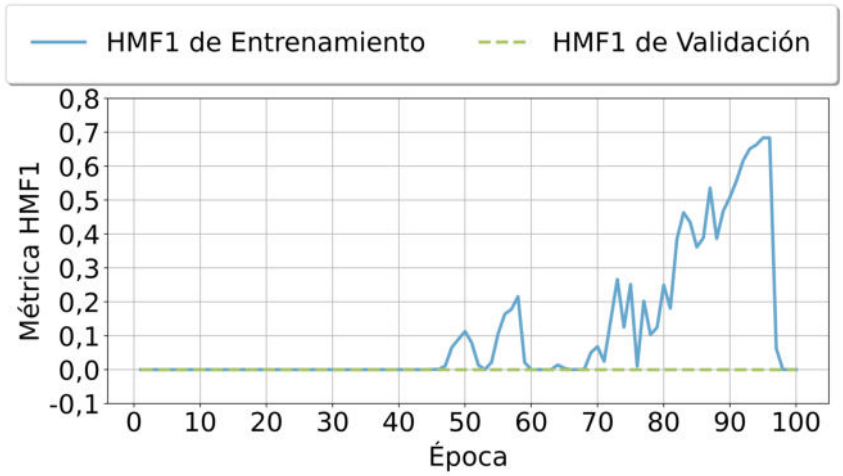


Figura 4.4: Evolución de la métrica *Harmonic Mean f1-score* sobre el conjunto de entrenamiento y validación en el experimento 1.

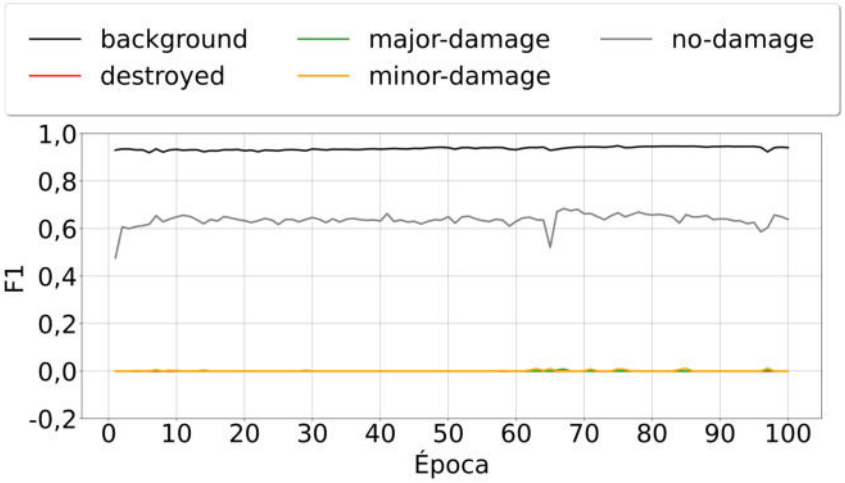


Figura 4.5: Evolución de la métrica *F1-score* para cada clase sobre el conjunto de validación durante el experimento 1.

EXP1_C0 Mejor época: 98 *val_loss* = 0.1278						
Conjunto	**Clase**	**HMF1**	**P**	**R**	**F1**	**ACC**
Validación	*background*	0,7754	0,9298	0,9517	0,9406	0,8989
	building		0,7066	0,6183	0,6595	0,8989
	background	0,0000	0,9301	0,9515	0,9407	0,8989
	no-damage		0,7046	0,6158	0,6572	0,8993
	minor-damage		0,0000	0,0000	0,0000	0,9992
	major-damage		0,0000	0,0000	0,0000	0,9988
	destroyed		0,0000	0,0000	0,0000	1,0000

Tabla 4.6: Métricas resultado de la ejecución de la mejor época con la mejor configuración 1 sobre el conjunto de *validación*.

en conjunto sin alguna clase presente, como sucedió con la clase *destroyed* en la Tabla 4.6.

- El desempeño del modelo para detectar edificios es considerablemente mejor que el de clasificación de daños, sobre todo para detectar píxeles que no pertenecen a un edificio.

4.2. Experimento 2: Aumentación de datos con CutMix

A partir de los resultados obtenidos en el Experimento 1, se observó que el desbalance en la cantidad de edificios por clase afecta negativamente la capacidad del modelo para clasificar correctamente los daños. Por esta razón, en este experimento (EXP2) se evalúa una posible estrategia para balancear el conjunto de datos.

Dadas las características del dataset xBD, el proceso de balanceo presenta desafíos significativos. En este enfoque inicial, se propone generar imágenes sintéticas con el objetivo de equilibrar la cantidad de edificios por clase.

La técnica utilizada se inspira en el método CutMix, pero con una modificación clave: en lugar de insertar un parche aleatorio en otra imagen, el objetivo aquí es reemplazar edificios específicos según su nivel de daño. El proceso consiste en recortar edificios de una imagen y reemplazarlos en otra imagen que tenga edificios de una categoría similar, buscando preservar la coherencia del contexto.

4.2.1. Descripción del conjunto de entrenamiento

	Entrenamiento	**Validación**	**Prueba**	**Total**
Cantidad	615	19	22	692

Tabla 4.7: Cantidad de parches en cada subconjunto de datos del experimento 2.

Para este experimento se vuelve a utilizar el conjunto de parches de *mexico-eathquake* detallado en la Sección 4.1.1. Sin embargo, para el experimento 2, se han añadido nuevas imágenes sintetizadas, resultando en un total de **692** parches divididos en subconjuntos como se muestra en la Tabla 4.7.

En la Tabla 4.8 se muestra la cantidad de edificios resultante tras la aplicación de la técnica de balanceo. Se puede observar que, aunque la diferencia en la cantidad de edificios entre las clases minoritarias se ha reducido, la clase mayoritaria sigue siendo significativamente más numerosa en comparación con las demás. Esto se debe a que, en cada imagen sintetizada, no todos los edificios son reemplazados. Como resultado, con cada nueva imagen generada, inevitablemente se añaden más edificios de la clase mayoritaria.

	destroyed	*major-damage*	*minor-damage*	*no-damage*	*un-classified*	**Total**
mexico-earthquake	41 631	41 584	41 671	96 468	85	**221 439**

Tabla 4.8: Cantidad de edificios de cada clase presentes en los 692 parches del desastre *Mexico-earthquake* aumentados con la técnica inspirada en *CutMix*.

4.2.1.1. Asignación de pesos para entrenamiento

En este experimento se utilizan los pesos calculados en función de los parches del conjunto de entrenamiento que se detallan en la Tabla 4.9. Se puede destacar que el peso para la clase *building* tiene un peso menor que el de la clase *background* en la rama de segmentación. Esto se debe a que con el sobre muestreo de edificios la cantidad de píxeles pertenecientes a un edificio es mayor que la cantidad de píxeles pertenecientes al fondo.

Rama	Clase	Peso
Segmentación	*background*	3,01
	building	1,50
Clasificación	*background*	3,01
	no-damage	3,24
	minor-damage	6,72
	major-damage	8,71
	destroyed	10,43

Tabla 4.9: Pesos calculados en función del desbalance para cada clase en el experimento 2.

4.2.2. Búsqueda y Optimización de Hiperparámetros

Los hiperparámetros explorados en este experimento son los mismos que en el experimento 1 descritos en la Sección 4.1.2, en la Figura 4.2 y Figura 4.2.

4.2.2.1. Tiempos de Ejecución

Se puede apreciar en la Tabla 4.10 que el tiempo de ejecución creció considerablemente, en comparación con el experimento 1. Esto se debe principalmente a la cantidad de imágenes de parches de entrenamiento utilizados. La etapa de preprocesamiento también aumento considerablemente debido a que la cantidad de cómputo requerido para sintetizar las imágenes es mayor.

Ejecución	Tiempo de ejecución
Preproc.	00:52:33
C0	36:03:15
C1	35:35:24
C2	34:30:57
C3	33:33:52
C4	34:19:40
C5	32:34:09
Entr. EXP2_C1	07:21:34
Postproc.	00:02:02

Tabla 4.10: Tiempo de ejecución, en (hh:mm:ss), transcurrido para cada etapa del experimento 2.

4.2.2.2. Selección de la mejor configuración

La Tabla 4.11 resume la búsqueda de hiperparámetros de este experimento, donde el valor de HMF1 ha aumentado considerablemente respecto al experimento anterior.

4.2.3. Desempeño del modelo EXP2_C1

La configuración 1 fue la configuración seleccionada para realizar el entrenamiento del modelo EXP2_C1 donde el factor de aprendizaje es de 0,001 y el tamaño de lote es de 16 parches.

La Figura 4.6 muestra la evaluación de la función de perdida sobre el conjunto de entrenamiento y validación a lo largo de las 100 épocas de entrenamiento y se puede notar que la función de un comportamiento poco deseado. La función de perdida aumenta sobre el conjunto de validación desde la primera época.

Este comportamiento se debe a que el conjunto de validación no contiene las imágenes aumentadas y conserva la distribución original del conjunto de datos. Principalmente, esto hace inferir que el modelo está aprendiendo patrones en el conjunto de entrenamiento que no existen en el conjunto de validación.

Este comportamiento hace que la métrica HMF1 aumente considerablemente sobre el conjunto de entrenamiento, pero no aumente sobre el conjunto de validación, como se puede apreciar en la Figura 4.8.

La métrica F1 graficada en la Figura 4.8 nos muestra que el modelo es capaz de aprender a predecir píxeles de la clase *background* y *no-damage*, pero no las demás clases que fueron aumentadas con la técnica inspirada en CutMix en la fase de preprocesamiento.

4.2.3.1. Desempeño en la mejor época

En la Tabla 4.12 se muestra el desempeño del modelo en la época 7, época en la que se obtuvieron HMF1 más alta sobre el conjunto de validación. En esta época el valor de la métrica HMF1 es considerablemente pequeño, tanto que al redondearlo con 4 decimales, el valor se redondee a 0.

Conf.	*Fold*	Mejor época	Perdida en validación	HMF1 de Clasificación	F1 de Segmentación
C0	F0	8	0,1424	0,6927	0,8727
	F1	36	0,1289	0,7204	**0,9092**
	F2	72	0,1736	0,7000	0,9057
	F3	29	0,1168	0,7469	0,9026
	F4	56	0,1316	0,7092	0,9089
C1	F0	87	0,1287	0,7223	0,8934
	F1	51	0,1700	0,6469	0,8869
	F2	73	0,1140	**0,7522**	0,8977
	F3	92	0,1144	0,7457	0,8988
	F4	94	0,1222	0,7454	0,8979
C2	F0	70	0,0608	0,7068	0,9006
	F1	13	0,0607	0,6892	0,8968
	F2	35	0,0636	0,6711	0,9020
	F3	29	0,0572	0,7069	0,8994
	F4	5	0,0627	0,7068	0,8707
C3	F0	39	0,0637	0,6304	0,8827
	F1	42	0,0617	0,6485	0,8864
	F2	75	0,0624	0,6710	0,8912
	F3	92	0,0630	0,6810	0,8922
	F4	83	0,0736	0,5709	0,8886
C4	F0	75	0,0409	0,6935	0,9040
	F1	26	0,0346	0,6727	0,8878
	F2	30	0,0314	0,6657	0,8882
	F3	52	0,0330	0,6156	0,9009
	F4	55	0,0385	0,7181	0,9015
C5	F0	16	0,0300	0,6132	0,8770
	F1	87	0,0324	0,5631	0,8567
	F2	10	0,0314	0,5783	0,8654
	F3	44	**0,0300**	0,6125	0,8763
	F4	33	0,0309	0,6330	0,8755

Tabla 4.11: Resumen del rendimiento de cada configuración del experimento 2 en la mejor época lograda durante el entrenamiento de cada *fold*.

Observando la tabla podemos ver que el modelo tiene un rendimiento menor en la tarea de segmentación de edificios en comparación con el modelo obtenido en el experimento 1. El desempeño del modelo para clasificar las clases de daño minoritarias, es considerablemente pobre.

4.2.4. Discusión

Los resultados de este experimento han permitido obtener las siguientes conclusiones:

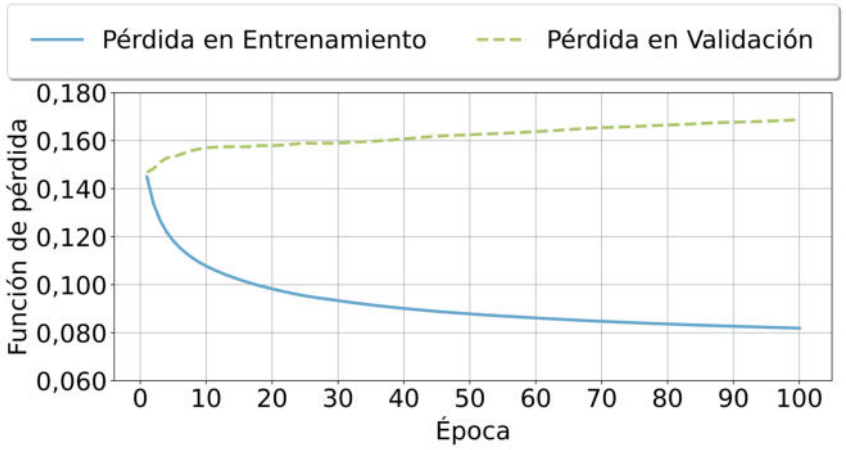


Figura 4.6: Evolución de la función de pérdida sobre el conjunto de entrenamiento y validación durante el experimento 2.

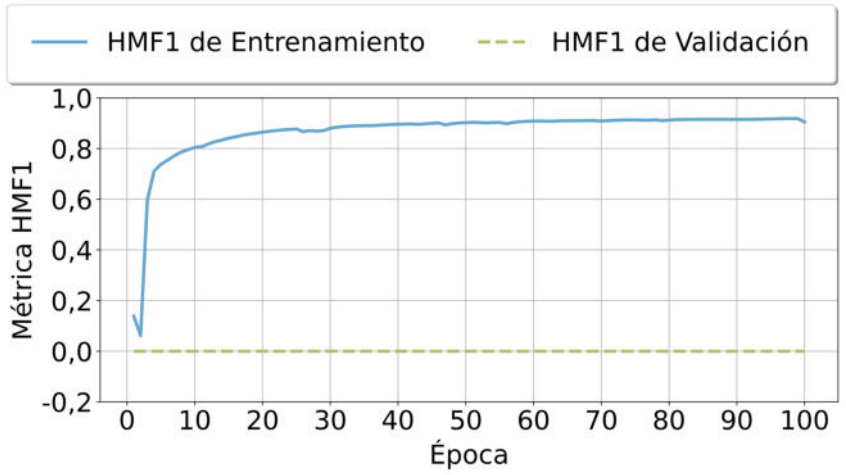


Figura 4.7: Evolución de la métrica *Harmonic Mean f1-score* sobre el conjunto de entrenamiento y validación en el experimento 2.

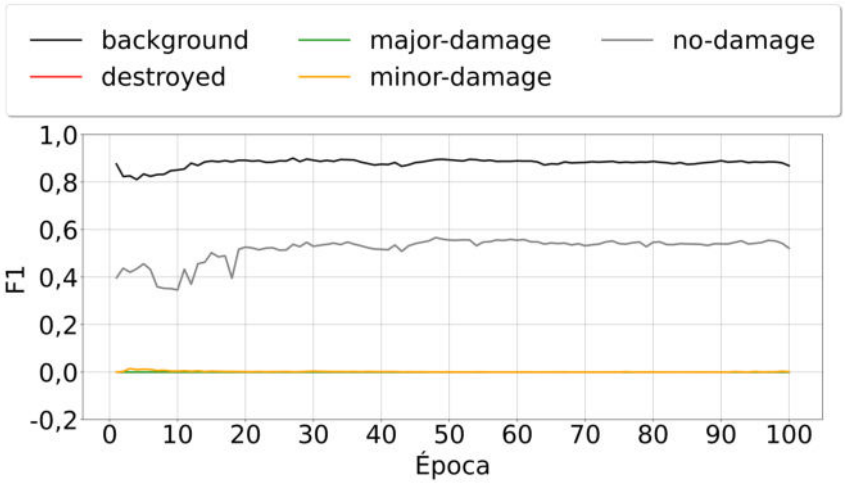


Figura 4.8: Evolución de la métrica *F1-score* para cada clase sobre el conjunto de validación durante el experimento 2.

EXP2_C1 Mejor época: 7 *val_loss* = 0,1552						
Conjunto	**Clase**	**HMF1**	**P**	**R**	**F1**	**ACC**
Validación	*background*	0,5845	0,8833	0,7853	0,8315	0,7421
	building		0,3783	0,5574	0,4507	0,7421
	background	0,0000	0,8837	0,7850	0,8315	0,7420
	no-damage		0,3558	0,3627	0,3592	0,7585
	minor-damage		0,0035	0,0310	0,0063	0,9754
	major-damage		0,0000	0,0025	0,0000	0,9504
	destroyed		0,0000	0,0000	0,0000	0,9824

Tabla 4.12: Métricas resultado de la ejecución de la mejor época con la mejor configuración del experimento 2 sobre el conjunto de *validación*.

- La aumentación de datos utilizando la técnica de remplazo de edificios individuales no permite tratar con el problema de desbalance de clases. Esto se debe principalmente a que el modelo se centra en aprender patrones que no existen en la distribución real.
- Podemos ver que la mejor configuración es aquella donde el tamaño de lotes de parches es 16, al igual que en el experimento 1.

4.3. Experimento 3: Muestreo con optimización voraz

A partir de los resultados obtenidos en el experimento 1 y 2, queda claro que el desbalance de las clases de nivel de daño presentes en el conjunto de datos xBD es la principal dificultad que no permite al modelo aprender correctamente para realizar la tarea de clasificación.

En este experimento, se utiliza una técnica diseñada específicamente para abordar el problema del desbalance de clases, priorizando evitar un aumento drástico en la cantidad de imágenes de entrenamiento. Además, la técnica empleada busca utilizar únicamente imágenes presentes en el conjunto de datos, evitando así la necesidad de un aumento de datos, que, como se observó en el experimento anterior, no es trivial.

4.3.1. Descripción del conjunto de entrenamiento

Para lograr un conjunto de datos balanceado en este experimento, se seleccionaron 1253 parches del conjunto de datos xBD utilizando una **técnica de muestreo voraz** propuesta en esta tesis presentada en la Sección 3.5.2. Estos parches se dividen en los tres subconjuntos especificados en la Tabla 4.13.

	Entrenamiento	**Validación**	**Prueba**	**Total**
Cantidad	1 002	126	125	**1 253**

Tabla 4.13: Cantidad de parches en cada subconjunto de datos del experimento 3.

En la Tabla 4.14 se presenta la cantidad de edificios de cada clase en los 1253 parches, clasificados por desastre natural y tipo de daño. Se puede observar que el total de edificios no muestra un desbalance significativo, a diferencia de los conjuntos de datos utilizados en experimentos anteriores

Etiqueta \ Desastre	*no-damage*	*minor-damage*	*major-damage*	*destroyed*	*un-classified*	**Total**
guatemala-volcano	42	2	0	0	0	44
hurricane-florence	692	11	74	1	45	823
hurricane-harvey	919	2 169	**9 884**	388	173	**13 533**
hurricane-matthew	753	**8 880**	1 231	701	**327**	11 892
hurricane-michael	1 266	1 070	348	132	36	2 852
joplin-tornado	623	668	487	2 298	119	4 195
lower-puna-volcano	302	10	1	229	85	627
mexico-earthquake	47	0	0	0	17	64
midwest-flooding	627	32	20	19	34	732
moore-tornado	452	214	183	934	53	1 836
nepal-flooding	712	597	1 072	31	62	2 474
palu-tsunami	**1 715**	1	279	**4 350**	85	6 430
pinery-bushfire	902	13	19	51	79	1 064
portugal-wildfire	1 504	15	31	150	129	1 829
santa-rosa-wildfire	1 059	41	39	3 896	32	5 067
socal-fire	1 379	17	23	695	184	2 298
sunda-tsunami	132	0	1	2	245	380
tuscaloosa-tornado	537	341	50	208	84	1 220
woolsey-fire	979	32	22	521	41	1 595
Total	**14 642**	**14 113**	**13 764**	**14 606**	**1 830**	**58 955**

Tabla 4.14: Cantidad de edificios de cada clase presentes en las 1253 imágenes obtenidas utilizando un algoritmo voraz.

4.3.1.1. Asignación de pesos para Entrenamiento

En este experimento se vuelve a utilizar la técnica de entrenamiento con pesos para tratar el desbalance presente en la cantidad de píxeles en las imágenes, utilizando los pesos de la Tabla 4.15.

Rama	Clase	Peso
Segmentación	*background*	1,56
	building	2,77
Clasificación	*background*	1,56
	no-damage	6,76
	minor-damage	15,22
	major-damage	13,66
	destroyed	13,59

Tabla 4.15: Pesos calculados en función del desbalance de píxeles, para cada clase en segmentación y clasificación de daños en el experimento 3.

4.3.2. Búsqueda y Optimización de Hiperparámetros

Teniendo en cuenta los resultados de la búsqueda de hiperparámetros realizada en los experimentos 1 y 2, donde las configuraciones que alcanzaron un valor de HMF1 más alto utilizaron un tamaño de lote de 16, en este experimento se establece el dominio de hiperparámetros descrito en la Figura 4.9.

Como se puede ver en la Figura 4.10, las configuraciones exploradas se diferencian únicamente en el factor de aprendizaje.

Hiperparámetros	Valores
Tamaño de batch:	16
Factor de aprendizaje:	$\{10^{-1};\ldots;10^{-6}\}$
Total de épocas:	50

Figura 4.9: Hiperparámetros explorados en el experimento 3.

Conf. N°	BS	LR	Épocas Totales
C0	16	0,000001	50
C1	16	0,00001	50
C2	16	0,0001	50
C3	16	0,001	50
C4	16	0,01	50
C5	16	0,1	50

Figura 4.10: Configuraciones de entrenamiento del experimento 3.

4.3.2.1. Tiempos de Ejecución

Como se ve en la Tabla 4.16, el tiempo de ejecución para probar cada configuración es menor al del experimento 2. Esto se debe a que a pesar de haber aumentado el tamaño del conjunto de entrenamiento se disminuyo la cantidad de épocas de entrenamiento, al menos para la prueba de configuraciones.

Ejecución	Tiempo de ejecución
Preproces.	00:16:58
C0	27:02:57
C1	25:37:56
C2	26:47:10
C3	26:28:45
C4	26:56:43
C5	27:27:35
Entr. EXP3_C3	23:15:53
Postproces.	00:05:10

Tabla 4.16: Tiempo de ejecución, en (hh:mm:ss), transcurrido para cada etapa del experimento 3 en el cluster Mendieta.

4.3.2.2. Selección de la mejor configuración

En la Tabla 4.17 podemos ver los resultados de la búsqueda de hiperparámetros para este experimento. Se puede apreciar que la métrica HMF1 más alta es obtenida con la configuración 3 (EXP3_C3).

Conf.	*Fold*	Mejor época	Perdida en validación	HMF1 de Clasificación	F1 de Segmentación
C0	F0	48	0,1652	0,0853	0,6153
	F1	50	0,1268	0,2625	0,7529
	F2	50	0,1554	0,0660	0,6931
	F3	7	0,1889	0,0441	0,2573
	F4	31	0,2067	0,0832	0,3151
C1	F0	48	0,1113	0,5385	0,8386
	F1	48	0,1115	0,5276	0,8454
	F2	49	0,1179	0,5191	0,8313
	F3	45	0,1160	0,5222	0,8375
	F4	49	0,1180	0,5102	0,8220
C2	F0	45	0,1078	0,6108	0,8622
	F1	37	0,1046	0,6332	0,8713
	F2	38	0,1060	0,6280	0,8706
	F3	17	0,1083	0,6245	0,8683
	F4	43	0,1066	0,6245	0,8780
C3	F0	32	0,1035	0,6332	0,8759
	F1	42	0,1035	0,6406	0,8775
	F2	21	**0,1033**	**0,6498**	0,8809
	F3	23	0,1036	0,6440	**0,8830**
	F4	43	0,1051	0,6252	0,8815
C4	F0	29	0,1039	0,6100	0,8709
	F1	42	0,1058	0,6012	0,8762
	F2	26	0,1079	0,5531	0,8365
	F3	36	0,1048	0,6271	0,8723
	F4	44	0,1053	0,5999	0,8784
C5	F0	37	0,1166	0,0000	0,8156
	F1	48	0,1117	0,0000	0,8562
	F2	3	0,1188	0,0000	0,7272
	F3	25	0,1138	0,0000	0,8426
	F4	42	0,1163	0,0000	0,8487

Tabla 4.17: Resumen del rendimiento de cada configuración del experimento 3 en la mejor época lograda durante el entrenamiento de cada *fold*.

4.3.3. Reducción del factor de aprendizaje

Los resultados de la búsqueda de hiperparámetros mostraron que la mejor configuración fue la número 3 (EXP3_C3). En este experimento, con el objetivo de seguir explorando el factor de aprendizaje, se entrenaron varios modelos durante 100 épocas, variando la cantidad de épocas de paciencia utilizadas para el ROPL. Esto se hizo principalmente para analizar el comportamiento del modelo al modificar el número de épocas que se espera antes de reducir el factor de aprendizaje, es decir, la paciencia del ROPL. Después de identificar la mejor configuración de la

búsqueda de hiperparámetros, se decidió realizar entrenamientos de 100 épocas para cada valor de paciencia, con el fin de observar cómo afecta este parámetro al rendimiento del modelo. Además, dado que se detectó que hay gran dificultad para aprender sobre las clases *minor-damage* y *major-damage*, se modificaron los pesos como se ve en la Tabla 4.18.

Rama	Clase	Peso
Segmentación	*background*	1,56
	building	2,77
Clasificación	*background*	1,56
	no-damage	6,76
	minor-damage	10 + 15,22
	major-damage	10 + 13,66
	destroyed	13,59

Tabla 4.18: Modificación de los pesos para aprender las clases con más dificultad.

Con el fin de probar el método ROPL aplicada al entrenamiento de nuestro modelo, se realizaron varios entrenamientos utilizando la configuración (C3) y se varió la paciencia como se detalla en la Tabla 4.19. Se puede notar que el modelo que alcanzo el mejor desempeño con la métrica HMF1 ha sido aquel que no utilizaba la técnica ROPL que está marcado con el símbolo - en la tabla.

Conf.	Paciencia de ROPL	Mejor época	Perdida en validación	HMF1 de Clasificación	F1 de Segmentación
C3	-	71	0,1120	**0,6341**	**0,8824**
	10	40	0,1114	0,5939	0,8640
	20	32	0,1113	0,5754	0,8459
	30	28	**0,1094**	0,6002	0,8811

Tabla 4.19: Tabla de búsqueda de hiperparámetros paciencia de ROPL con la mejor configuración.

4.3.3.1. Evolución de la función de pérdida y métricas

El modelo EXP3_C3 fue entrenado durante 100 épocas, con un batch de 16, un factor de aprendizaje 0,001 y sin utilizar la técnica ROPL. A continuación se describe la evolución de cada curva durante su entrenamiento.

En la Figura 4.11, se observa que la función de pérdida del conjunto de validación disminuye hasta la época 20. Sin embargo, a partir de esa época, el modelo comienza a sobreajustarse al conjunto de entrenamiento. Esta tendencia se refleja en la Figura 4.12, donde el valor de la métrica HMF1 aumenta hasta alcanzar un valor de 0,6, el cual difícilmente es superado después de la época 20. Además, en la Figura 4.13 se puede notar que la métrica F1 de cada clase presenta pocos cambios a lo largo del entrenamiento tras la época 20.

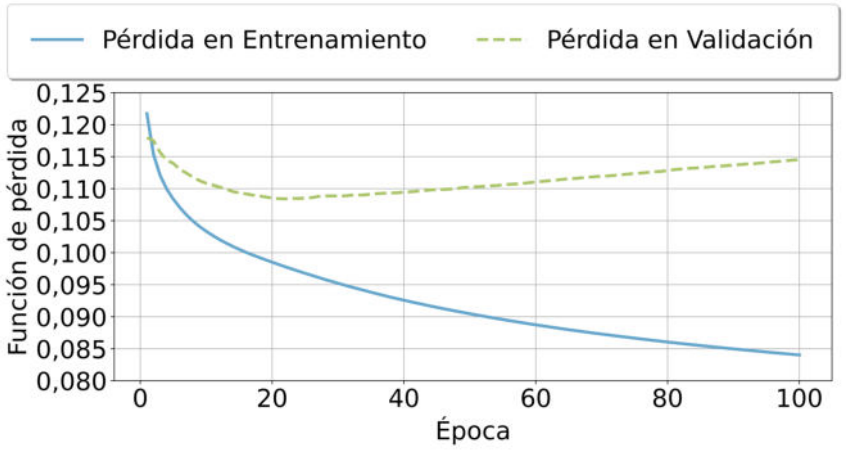


Figura 4.11: Evolución de la función de pérdida sobre el conjunto de entrenamiento y validación durante el experimento 3.

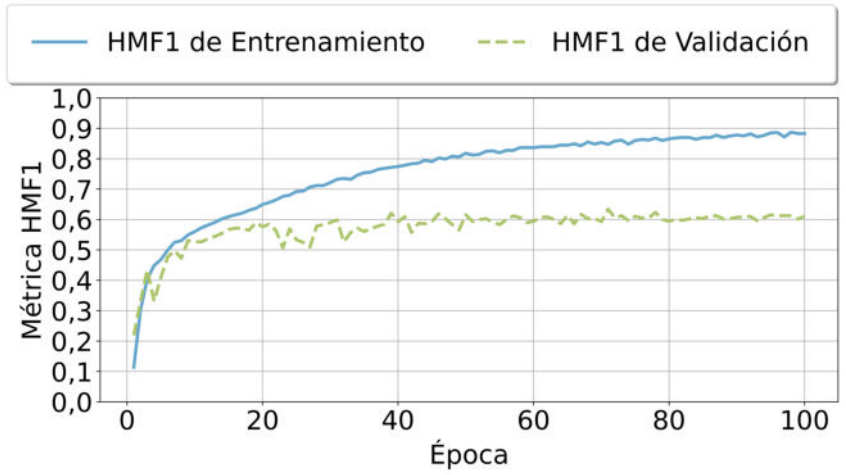


Figura 4.12: Evolución de la métrica *Harmonic Mean f1-score* sobre el conjunto de entrenamiento y validación en el experimento 3.

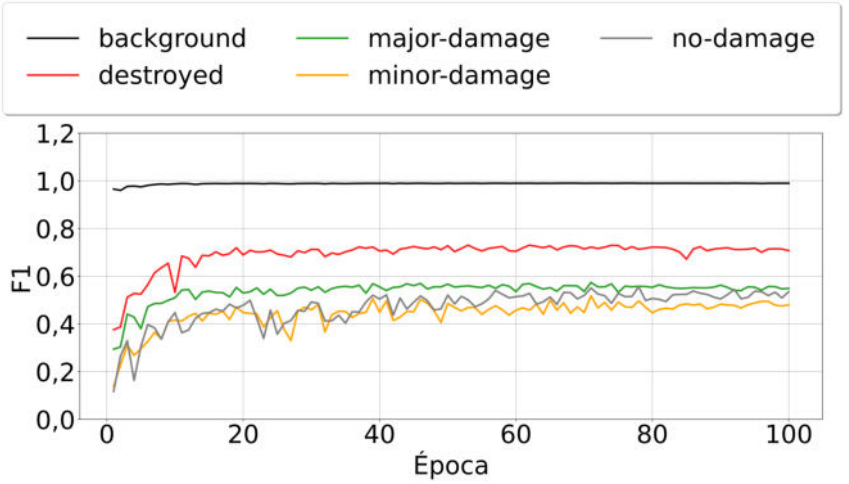


Figura 4.13: Evolución de la métrica *F1-score* para cada clase sobre el conjunto de validación durante el experimento 3.

4.3.4. Desempeño del modelo EXP3_C3

Observando la Tabla 4.20, se puede notar que el desempeño del modelo ha mejorado con respecto a los experimentos anteriores. El modelo ha demostrado una mayor capacidad para predecir las clases minoritarias de manera más precisa en comparación con los experimentos previos.

Mejor Configuración: EXP3_C3 Mejor época: 71 *val_loss* = 0,1120 *test_loss* = 0,1117						
Conjunto	**Clase**	**HMF1**	**P**	**R**	**F1**	**ACC**
Validación	*background*	0,8824	0,9895	0,9896	0,9895	0,9800
	building		0,7971	0,7954	0,7963	0,9800
	background	0,6341	0,9896	0,9896	0,9896	0,9803
	no-damage		0,6008	0,5078	0,5504	0,9860
	minor-damage		0,5135	0,5221	0,5178	0,9894
	major-damage		0,5442	0,6070	0,5739	0,9899
	destroyed		0,6839	0,7653	0,7223	0,9943
Prueba	*background*	0,8705	0,9932	0,9918	0,9925	0,9855
	building		0,7593	0,7917	0,7751	0,9855
	background	0,5571	0,9933	0,9918	0,9925	0,9856
	no-damage		0,5585	0,5022	0,5289	0,9913
	minor-damage		0,3320	0,3898	0,3586	0,9914
	major-damage		0,5200	0,5325	0,5262	0,9926
	destroyed		0,6701	0,7797	0,7208	0,9954

Tabla 4.20: Experimento 3. Métricas resultado de la ejecución de la mejor época con la mejor configuración 7 sobre el conjunto de *validación* y *prueba*.

La Tabla 4.21 es la tabla de predicciones sobre el conjunto de prueba que muestra el desempeño en la tarea de segmentación del modelo EXP3_C3. Particularmente se puede observar que el modelo tiene un muy buen desempeño para clasificar los píxeles de fondo. Debido al desbalance entren la cantidad de píxeles de fondo y edificios, podemos ver que un 21% de los píxeles correspondientes con edificios en el conjunto de prueba fueron clasificados como fondo.

		Predicción			
		Background	*Building*	**Total**	**Total%**
Ground Truth	*Background*	**99%**	1%	126 988 492	97%
	Building	21%	**79%**	4 083 508	3%
	Total	97%	3%	**131 072 000**	100%

Tabla 4.21: Matriz de predicciones de segmentación de cada píxel en el conjunto de Prueba realizadas por el modelo EXP3_C3 en el experimento 3.

Por otro lado, la Tabla 4.22 expone las predicciones realizadas por el modelo EXP3_C3 al clasificar los píxeles del conjunto de prueba. Se observa que la clase con

mayor dificultad de predicción es la *minor-damage*. La cual tiende a confundirse tanto con los edificios sin daños como con aquellos que presentan daños mayores. Esta confusión puede atribuirse a las características específicas del conjunto de datos. Sin embargo, el modelo demuestra un desempeño notable en la detección de los píxeles correspondientes a los edificios destruidos en las imágenes de prueba.

		Predicción					
		no-damage	*minor-damage*	*major-damage*	*destroyed*	**Total**	**Total %**
Ground Truth	*no-damage*	**69 %**	17 %	12 %	2 %	921 404	28 %
	minor-damage	13 %	**49 %**	34 %	4 %	616 507	19 %
	major-damage	8 %	20 %	**65 %**	7 %	860 316	27 %
	destroyed	0 %	3 %	6 %	**91 %**	838 824	26 %
	Total	27 %	20 %	26 %	27 %	**3 237 051**	100 %

Tabla 4.22: Matriz de predicciones de daño de cada píxel en el conjunto de Prueba realizadas por el modelo Exp3_C3 en el experimento 3.

4.3.4.1. Ejemplos de predicción

A continuación se presentan algunos ejemplos de los resultados del experimento 3. En la Figura 4.14 se muestra una predicción realizada por el modelo en una de las imágenes del conjunto de prueba. Como se puede observar, es posible distinguir los edificios del fondo, y clasifica correctamente edificios como destruidos.

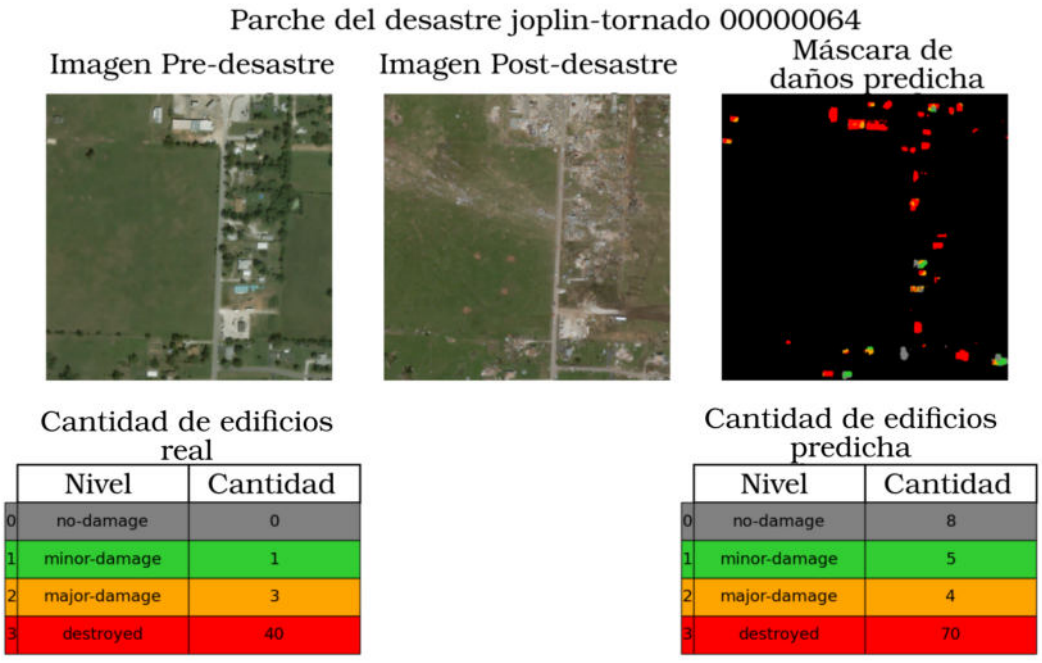


Figura 4.14: Ejemplo de predicción del modelo del experimento 3 con 100 épocas sobre una imagen del conjunto de prueba.

4.3.5. Discusión

De este experimento podemos sacar las siguientes conclusiones.

- Las clases más difíciles de aprender para el modelo son la clase *minor-damage* y *major-damage*, las cuales suelen ser confundidas entre sí o clasificadas como edificios sin daños.
- El desempeño del modelo para clasificar píxeles como fondo y píxeles como *destroyed* es bastante bueno.
- La estrategia de *muestreo voraz* junto con el entrenamiento con pesos fueron beneficiosos y permitieron que el modelo lograra un mejor desempeño que en los experimentos anteriores.
- El modelo sobreajusta el conjunto de entrenamiento muy rápidamente, durante las primeras épocas y sus métricas sobre el conjunto de validación encuentran un techo que difícilmente pueden superar.
- Modificar el factor de aprendizaje durante el entrenamiento con la técnica ROPL realmente no presenta mejoras en el desempeño del modelo.
- Debido a que el modelo tiene un enfoque basado en segmentación semántica multiclase, presenta dificultad para crear polígonos que puedan ser asociados con cada edificio presente en la imagen. Esto conduce a un mal desempeño en su rendimiento para clasificar edificios.

4.4. Experimento 4: Voraz aumento del número de imágenes

En este experimento, el modelo se entrena con un conjunto más amplio de parches seleccionados con el algoritmo voraz, considerando las limitaciones identificadas en los experimentos previos, con el objetivo de mejorar su desempeño en la tarea de generalización.

4.4.1. Descripción del conjunto de entrenamiento

Utilizando el enfoque de selección de muestreo voraz que se demostró que funciona en el experimento anterior, se seleccionan 3001 parches que se dividen en el conjunto de entrenamiento, validación y prueba correspondientes, como se ve en la Tabla 4.23.

La Tabla 4.30 muestra los edificios presentes por cada clase en los 3001 parches utilizados en este experimento. Es posible notar que la cantidad de edificios de cada clase es notablemente más que en el experimento 3, pero también presenta un mayor desbalance con respecto a la cantidad total de edificios de tipo *no-damage*.

4.4.2. Configuración

Para este experimento se utiliza la mejor configuración del experimento 3 resumidas en la Tabla 4.24.

Etiqueta \ Desastre	*no-damage*	*minor-damage*	*major-damage*	*destroyed*	*un-classified*	**Total**
guatemala-volcano	197	11	3	1	1	213
hurricane-florence	3 403	53	377	38	387	4 258
hurricane-harvey	1 954	2 993	**11 134**	439	297	**16 817**
hurricane-matthew	1 213	**9 801**	1 623	1 368	449	14 454
hurricane-michael	5 640	2 750	1 024	445	122	9 981
joplin-tornado	1 099	801	594	2 598	201	5 293
lower-puna-volcano	1 800	28	20	399	351	2 598
mexico-earthquake	350	4	1	0	43	398
midwest-flooding	2 479	100	79	39	89	2 786
moore-tornado	1 634	294	213	1 054	119	3 314
nepal-flooding	3 099	1 236	1 829	91	433	6 688
palu-tsunami	1 956	1	313	**4 469**	87	6 826
pinery-bushfire	2 708	19	43	98	267	3 135
portugal-wildfire	**7 258**	59	108	389	427	8 241
santa-rosa-wildfire	1 684	48	46	4 448	40	6 266
socal-fire	5 096	27	39	1 160	353	6 675
sunda-tsunami	518	0	13	2	**562**	1 095
tuscaloosa-tornado	1 815	582	96	403	238	3 134
woolsey-fire	2 700	69	49	991	96	3 905
Total	**46 603**	**18 876**	**17 604**	**18 432**	**4 562**	**106 077**

	Entrenamiento	**Validación**	**Prueba**	**Total**
Cantidad	2401	300	300	3001

Tabla 4.23: Cantidad de parches en cada subconjunto de datos del experimento 4.

Conf.	**BS**	**LR**	**Épocas Totales**
C0	16	0,001	50

Tabla 4.24: Configuraciones de entrenamiento del experimento 4.

4.4.2.1. Asignación de pesos para Entrenamiento

Los pesos utilizados en la función de perdida se detallan en la Tabla 4.25. Podemos notar que el peso para la clase *no-damage* es mucho menor que la de las demás clases y esto se debe a que ha aumentado su presencia en el conjunto de datos utilizado para entrenamiento.

Rama	**Clase**	**Peso**
Segmentación	*background*	1,57
	building	2,76
Clasificación	*background*	1,57
	no-damage	4,71
	minor-damage	19,02
	major-damage	20,42
	destroyed	20,78

Tabla 4.25: Pesos calculados en función del desbalance de píxeles, para cada clase en segmentación y clasificación de daños en el experimento 4.

La Tabla 4.26 exhibe los tiempos de ejecución de cada etapa de este experimento. El tiempo de entrenamiento EXP4_C0 ha disminuido con respecto al experimento 3. Esto se debe a que en este experimento únicamente se entrena durante 50 épocas y no 100 como en el experimento 3.

Ejecución	**Tiempo de ejecución**
Preprocesamiento	00:46:54
Entrenamiento EXP4_C0	15:21:30
Postprocesamiento	00:11:35

Tabla 4.26: Tiempo de ejecución, en (hh:mm:ss), transcurrido para cada etapa del experimento 4.

4.4.2.2. Evolución de las métricas de desempeño durante entrenamiento

La Figura 4.15 muestra que la función de pérdida sobre el conjunto de validación alcanza su punto mínimo en la época 25. A partir de esta época, la función de pérdida en el conjunto de validación comienza a aumentar, mientras que la función de pérdida sobre el conjunto de entrenamiento continúa disminuyendo, lo que sugiere que el modelo ha comenzado a sobreajustar los datos. Este comportamiento se confirma al analizar la Figura 4.16, donde se observa que la métrica HMF1 deja de mejorar a partir de la época 25. Por otro lado, la Figura 4.17 indica que la métrica F1 para la clase *no-damage* alcanza el mayor valor, mientras que la clase *minor-damage* registra el menor, siguiendo un patrón similar al observado en el experimento 3.

4.4.3. Desempeño del modelo EXP4_C0

En la Tabla 4.27 se muestran las métricas de desempeño alcanzada por el modelo en la época 40, aquella en la cual se alcanzó el valor HMF1 más alto sobre el conjunto de validación. De esta tabla se puede destacar que el valor de la métrica HMF1 sobre el conjunto de prueba es más alto que en el experimento 3, dando a entender que al aumentar la cantidad de imágenes de entrenamiento, el modelo es capaz de mejorar su capacidad de generalización.

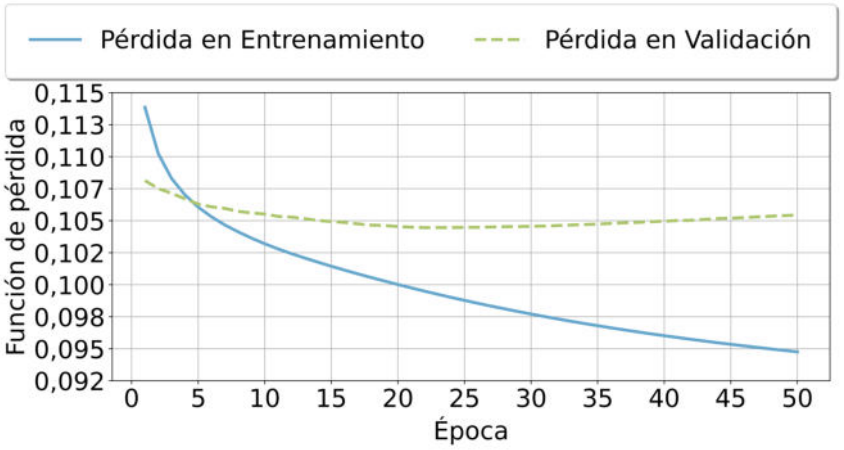


Figura 4.15: Evolución de la función de pérdida sobre el conjunto de entrenamiento y validación durante el experimento 4.

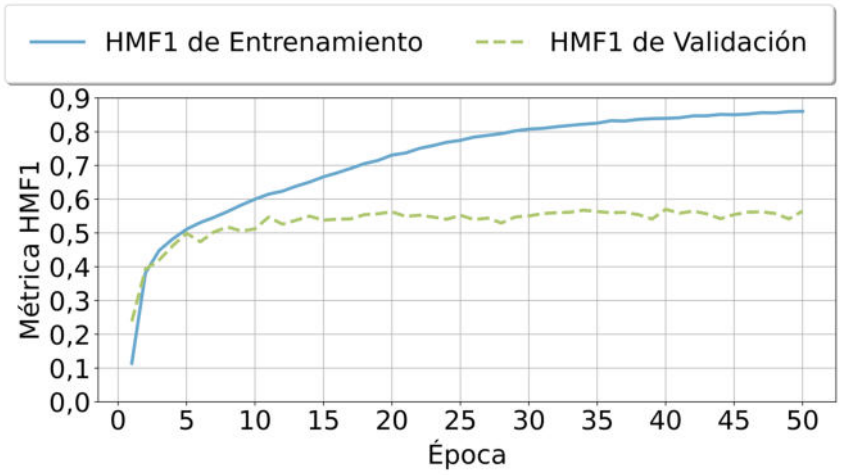


Figura 4.16: Evolución de la métrica *Harmonic Mean f1-score* sobre el conjunto de entrenamiento y validación en el experimento 4.

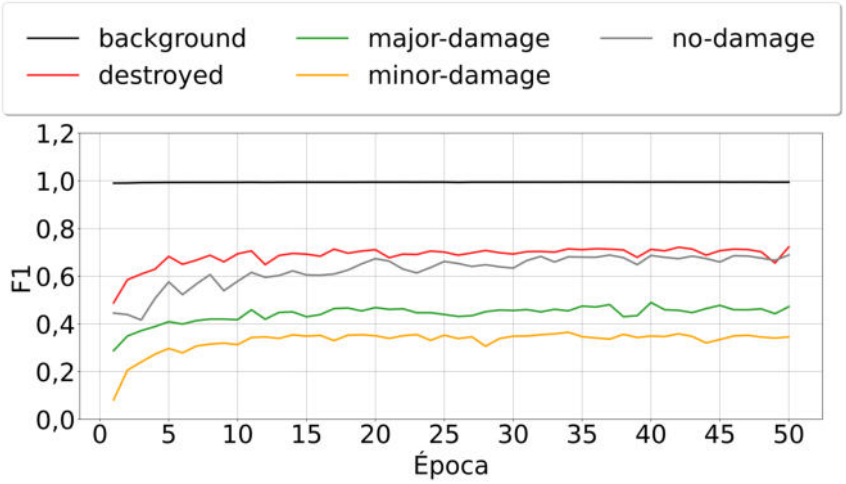


Figura 4.17: Evolución de la métrica *F1-score* para cada clase sobre el conjunto de validación durante el experimento 4.

Mejor época: 40	*val_loss* = 0,1050	*test_loss* = 0,1071				
Conjunto	**Clase**	**HMF1**	**P**	**R**	**F1**	**ACC**
Validación	*background*	0,8870	0,9934	0,9953	0,9944	0,9890
	building		0,8272	0,7754	0,8005	0,9890
	background	0,5701	0,9936	0,9953	0,9945	0,9892
	no-damage		0,7196	0,6577	0,6872	0,9908
	minor-damage		0,3459	0,3523	0,3491	0,9953
	major-damage		0,4927	0,4870	0,4898	0,9956
	destroyed		0,7435	0,6843	0,7127	0,9975
Prueba	*background*	0,8880	0,9916	0,9945	0,9930	0,9866
	building		0,8375	0,7712	0,8030	0,9866
	background	0,5776	0,9919	0,9944	0,9932	0,9868
	no-damage		0,7217	0,6401	0,6785	0,9884
	minor-damage		0,3867	0,3150	0,3472	0,9940
	major-damage		0,4927	0,5558	0,5224	0,9943
	destroyed		0,7283	0,7192	0,7237	0,9973

Tabla 4.27: Experimento 4. Métricas resultado de la ejecución de la mejor época con la mejor configuración 0 sobre el conjunto de *validación* y *prueba*.

4.4.3.1. Predicciones del modelo EXP4_C0

A continuación, en esta sección se presentan las tablas de predicción de píxeles del modelo EXP4_C0 en la época 40 sobre el conjunto de prueba.

La Tabla 4.28 muestra que el modelo ha sido capaz de predecir correctamente el 99 % de los píxeles que pertenecen al fondo y el 78 % de los píxeles que corresponden a los edificios. Estos valores indican un buen desempeño en la tarea de segmentación, aunque no presentan cambios significativos con respecto al experimento 3, a pesar del aumento en la cantidad de imágenes utilizadas para entrenamiento.

		Predicción			
		Background	*Building*	**Total**	**Total %**
Ground Truth	*Background*	**99 %**	1 %	303 625 852	97 %
	Building	22 %	**78 %**	10 946 948	3 %
	Total	97 %	3 %	**314 572 800**	100 %

Tabla 4.28: Matriz de predicciones de segmentación de cada píxel en el conjunto de Prueba realizadas por el modelo EXP4_C0 en el experimento 4.

A partir de Tabla 4.29 podemos interpretar que el modelo ha tenido una mejora en su capacidad para predecir la clase *no-damage* con respecto al experimento 3, pero sigue teniendo problemas significativos para predecir la clase *minor-damage* la cual tiende a categorizarse como *no-damage* y *major-damage*. El mejor desempeño del modelo se da para la clasificación de la clase *destroyed*.

		Predicción					
		no-damage	*minor-damage*	*major-damage*	*destroyed*	**Total**	**Total%**
Ground Truth	*no-damage*	**84%**	8%	8%	1%	4 567 570	54%
	minor-damage	34%	**42%**	22%	2%	1 194 436	14%
	major-damage	13%	13%	**68%**	6%	1 451 853	17%
	destroyed	2%	1%	8%	**89%**	1 254 993	15%
	Total	53%	12%	20%	15%	**8 489 852**	100%

Tabla 4.29: Matriz de predicciones de daño de cada píxel en el conjunto de Prueba realizadas por el modelo EXP4_C0 en el experimento 4.

4.4.3.2. Ejemplos de Predicción

La Figura 4.18 muestra una predicción realizada por el modelo en una de las imágenes del conjunto de prueba. Mostrando que en aspectos generales logra distinguir los edificios del fondo, y clasifica correctamente edificios como destruidos.

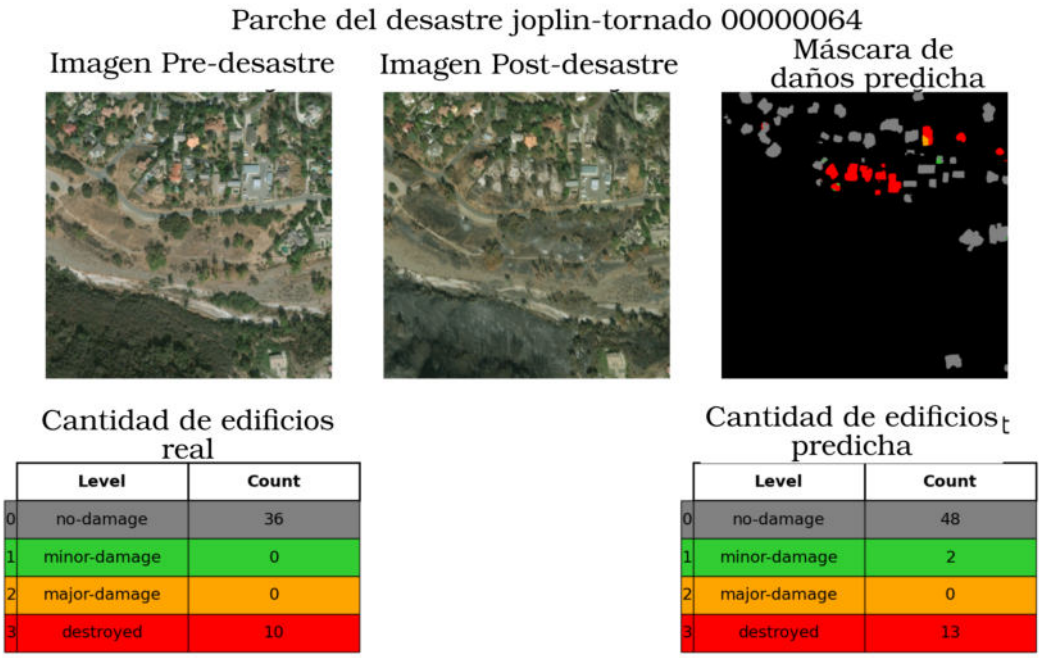


Figura 4.18: Ejemplo de predicción del modelo del experimento 4 sobre una imagen del conjunto de prueba.

4.4.4. Discusión

De este último experimento se pueden obtener las siguientes conclusiones:

- **Aumento de desbalance en el conjunto de entrenamiento**: El aumento en la cantidad de imágenes ha presentado un considerable aumento en el desbalance entre las clases de daño, pero ha permitido una mejora en la capacidad de inferencia y generalización del modelo para la tarea de clasificación de daños con respecto a su desempeño en experimentos anteriores.

- **Sobreajuste del conjunto de entrenamiento**: El modelo continúa presentando sobreajuste sobre el conjunto de entrenamiento durante las primeras épocas de entrenamiento.
- **Desempeño en la tarea de segmentación**: El modelo se desempeña considerablemente bien en la tarea de detección y segmentación de edificios, logrando capturar en gran parte los edificios en las imágenes de xBD.
- **Desempeño Consistente en la Clase *Background***: La clase *background* sigue siendo clasificada con alta precisión, lo que indica que el modelo prioriza esta clase, posiblemente debido a su mayor representatividad en las imágenes de entrenamiento.
- **Conjunto de prueba representativo**: Al incrementar el tamaño del conjunto de prueba podemos notar que el desempeño sobre el conjunto de validación para tarea de clasificación es similar a su desempeño sobre le conjunto de prueba lo que sugiere que el conjunto de prueba ha sido representativo.

Desastre \ Etiqueta	*no-damage*	*minor-damage*	*major-damage*	*destroyed*	*un-classified*	**Total**
guatemala-volcano	731	26	23	33	177	990
hurricane-florence	8 466	232	1 949	81	660	11 388
hurricane-harvey	18 638	4 510	**13 378**	848	502	37 876
hurricane-matthew	4 058	**12 331**	2 717	3 524	1 307	23 937
hurricane-michael	22 692	8 292	2 919	1 225	370	35 498
joplin-tornado	8 225	2 192	1 005	3 274	656	15 352
lower-puna-volcano	2 277	49	26	504	507	3 363
mexico-earthquake	51 084	221	54	3	107	51 469
midwest-flooding	12 819	246	193	165	468	13 891
moore-tornado	19 453	886	449	1 584	582	22 954
nepal-flooding	31 225	5 134	4 721	502	1 665	43 247
palu-tsunami	**46 796**	1	1 178	**7 203**	611	**55 789**
pinery-bushfire	5 027	82	99	229	453	5 890
portugal-wildfire	20 787	176	296	1 090	1 015	23 364
santa-rosa-wildfire	15 843	121	95	5 810	86	21 955
socal-fire	15 697	136	110	2 333	597	18 873
sunda-tsunami	14 078	0	100	179	**1 876**	16 233
tuscaloosa-tornado	10 499	2 036	466	1 097	879	14 977
woolsey-fire	4 638	189	126	1 876	175	7 004
Total	**313 033**	**36 860**	**29 904**	**31 560**	**14 011**	**425 368**

Tabla 4.30: Cantidad de edificios presentes en los 7000 parches con edificios clasificados del conjunto de datos xBD.

4.5. Experimento 5: Todo el conjunto de datos xBD

En este último experimento, se utilizaron un total de 7000 parches, seleccionados bajo el criterio de incluir únicamente aquellos que contenían edificios clasificados, excluyendo parches sin edificios o con edificios no clasificados del conjunto de datos xBD. Este experimento tiene el fin de verificar si se mejora la generalización del modelo al aumentar la cantidad de imágenes de entrenamiento.

4.5.1. Descripción del conjunto de entrenamiento

El conjunto xBD cuenta con 6983 parches que cumplen estas características, por lo que se añadieron 17 imágenes vacías adicionales para completar los 7000 parches. La Tabla 4.30 presenta la cantidad de edificios de cada clase incluidos en este experimento

Utilizando el enfoque de selección de muestreo voraz que se demostró que funciona en el experimento anterior, se seleccionan 7000 parches que se dividen en el conjunto de entrenamiento, validación y prueba correspondientes, como se ve en la Tabla 4.31.

	Entrenamiento	**Validación**	**Prueba**	**Total**
Cantidad	5600	700	700	7000

Tabla 4.31: Cantidad de parches en cada subconjunto de datos del experimento 5.

4.5.2. Configuración

Para este experimento, se utiliza la configuración presentada en la Tabla 4.32, donde destaca que la cantidad de épocas es considerablemente menor en comparación con los experimentos anteriores. Esto se debe al incremento en el tiempo de cómputo necesario al aumentar la cantidad de imágenes, así como al comportamiento observado del modelo en experimentos previos. Dado que el modelo tiende a sobreajustarse al conjunto de entrenamiento durante las primeras épocas, no se requiere un gran número de épocas de entrenamiento.

Conf.	**BS**	**LR**	**Épocas Totales**
C0	16	0,001	35

Tabla 4.32: Configuraciones de entrenamiento del experimento 5.

Rama	**Clase**	**Peso**
Segmentación	*background*	1,49
	building	3,06
Clasificación	*background*	1,49
	no-damage	4,90
	minor-damage	19,89
	major-damage	24,49
	destroyed	31,62

Tabla 4.33: Pesos calculados en función del desbalance de píxeles, para cada clase en segmentación y clasificación de daños en el experimento 5.

4.5.2.1. Asignación de pesos para Entrenamiento

Los pesos utilizados en la función de perdida se detallan en la Tabla 4.25. Podemos notar que el peso para la clase *no-damage* es mucho menor que la de las demás clases y esto se debe a que ha aumentado su presencia en el conjunto de datos utilizado para entrenamiento.

La Tabla 4.34 muestra los tiempos de ejecución de cada etapa de este experimento. El incremento en la cantidad de imágenes de entrenamiento ha producido un aumento considerable en el tiempo de preprocesamiento, de entrenamiento y de postprocesamiento.

Ejecución	**Tiempo de ejecución**
Preprocesamiento	01:51:55
Entrenamiento EXP5_C0	36:50:18
Postprocesamiento	00:31:11

Tabla 4.34: Tiempo de ejecución, en (hh:mm:ss), transcurrido para cada etapa del experimento 5.

4.5.2.2. Evolución de las métricas de desempeño durante entrenamiento

Observando la Figura 4.19, la Figura 4.20 y Figura 4.21, se puede inferir que, a pesar del considerable aumento en la cantidad de imágenes de entrenamiento, el modelo se sobreajusta al conjunto de entrenamiento rápidamente durante las primeras épocas, lo que impide mejorar su capacidad de generalización. Al igual que en los experimentos anteriores, la capacidad del modelo para clasificar la clase fondo sigue siendo superior. Sin embargo, se percibe una clara dificultad para clasificar la clase *minor-damage*.

4.5.3. Desempeño del modelo EXP5_C0

La Tabla 4.35 muestra que el modelo alcanza su mejor desempeño en la época 24, logrando una *val_loss* de 0,1028 y una *test_loss* de 0,1035. A pesar de haber incrementado más del doble la cantidad de parches en comparación con los experimentos anteriores, el modelo no logra un incremento significativo en la métrica HMF1. Sin embargo, se observa una ligera mejora en la tarea de segmentación respecto al experimento 4, aunque esta mejora no resulta considerable en términos prácticos.

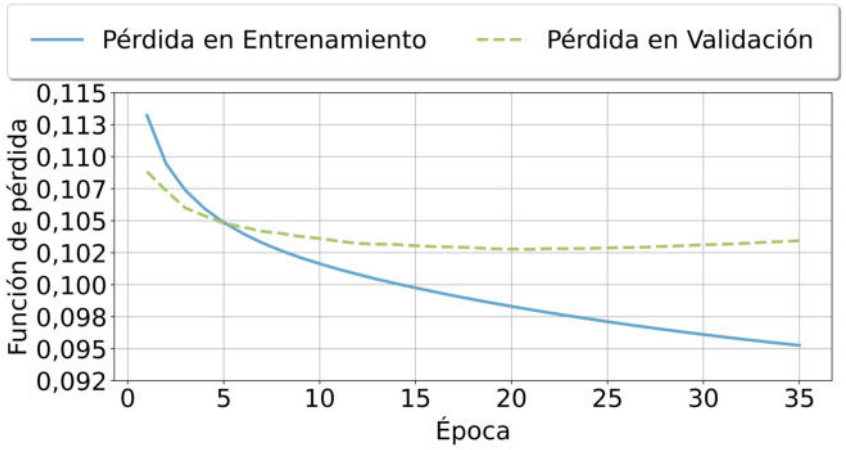


Figura 4.19: Evolución de la función de pérdida sobre el conjunto de entrenamiento y validación durante el experimento 5.

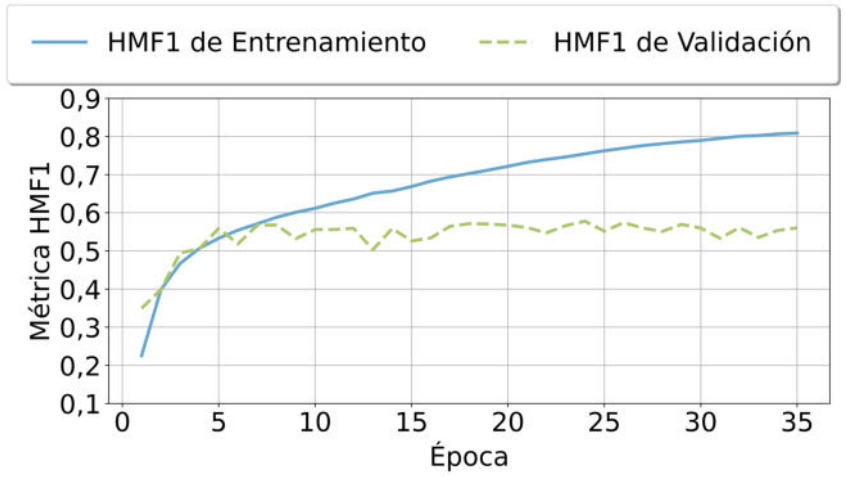


Figura 4.20: Evolución de la métrica *Harmonic Mean f1-score* sobre el conjunto de entrenamiento y validación en el experimento 5.

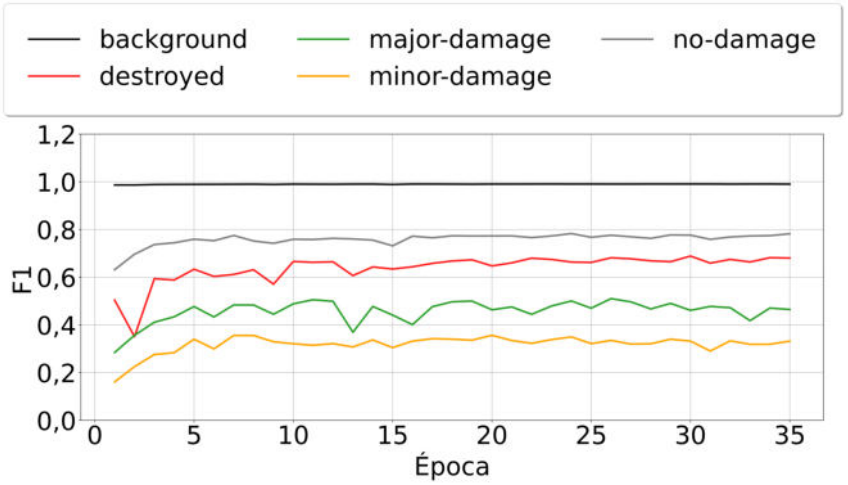


Figura 4.21: Evolución de la métrica *F1-score* para cada clase sobre el conjunto de validación durante el experimento 5.

Mejor época: 24		*val_loss* = 0,1028		*test_loss* = 0,1035		
Conjunto	**Clase**	**HMF1**	**P**	**R**	**F1**	**ACC**
Validación	*background*	0,9042	0,9899	0,9910	0,9904	0,9819
	building		0,8406	0,8232	0,8318	0,9819
	background	0,5780	0,9899	0,9912	0,9905	0,9821
	no-damage		0,7925	0,7736	0,7830	0,9819
	minor-damage		0,3430	0,3567	0,3497	0,9947
	major-damage		0,5331	0,4717	0,5005	0,9956
	destroyed		0,6458	0,6824	0,6636	0,9981
Prueba	*background*	0,9063	0,9916	0,9945	0,9930	0,9866
	building		0,8429	0,8274	0,8351	0,9827
	background	0,5536	0,9903	0,9914	0,9909	0,9827
	no-damage		0,7886	0,7695	0,7790	0,9821
	minor-damage		0,3295	0,3074	0,3181	0,9940
	major-damage		0,4663	0,4735	0,4699	0,9965
	destroyed		0,6616	0,7027	0,6815	0,9980

Tabla 4.35: Experimento 5. Métricas resultado de la ejecución de la mejor época con la mejor configuración 0 sobre el conjunto de *validación* y *prueba*.

4.5.3.1. Predicciones del modelo EXP5_C0

A continuación, en esta sección se presentan las tablas de predicción de píxeles del modelo EXP4_C0 en la época 40 sobre el conjunto de prueba.

La Tabla 4.28 muestra que el modelo ha sido capaz de predecir correctamente el 99 % de los píxeles que pertenecen al fondo y el 78 % de los píxeles que corresponden a los edificios. Estos valores indican un buen desempeño en la tarea de segmentación, aunque no presentan cambios significativos con respecto al experimento 3, a pesar del aumento en la cantidad de imágenes utilizadas para entrenamiento.

		Predicción			
		Background	*Building*	**Total**	**Total %**
Ground Truth	*Background*	**99 %**	1 %	695 962 566	95 %
	Building	18 %	**82 %**	38 040 634	5 %
	Total	95 %	5 %	**734 003 200**	100 %

Tabla 4.36: Matriz de predicciones de segmentación de cada píxel en el conjunto de Prueba realizadas por el modelo EXP5_C0 en el experimento 5.

A partir de Tabla 4.37 podemos interpretar que el modelo ha tenido una mejora en su capacidad para predecir la clase *no-damage* con respecto al experimento 3, pero sigue teniendo problemas significativos para predecir la clase *minor-damage* la cual tiende a categorizarse como *no-damage* y *major-damage*. El mejor desempeño del modelo se da para la clasificación de la clase *destroyed*.

		Predicción					
		no-damage	*minor-damage*	*major-damage*	*destroyed*	**Total**	**Total%**
Ground Truth	*no-damage*	**93%**	5%	2%	1%	24 926 846	80%
	minor-damage	45%	**40%**	14%	2%	2 577 437	8%
	major-damage	19%	17%	**57%**	7%	1 993 165	6%
	destroyed	2%	1%	7%	**89%**	1 801 445	6%
	Total	79%	8%7%	6%	**31 298 893**	100%	

Tabla 4.37: Matriz de predicciones de daño de cada píxel en el conjunto de Prueba realizadas por el modelo EXP5_C0 en el experimento 5.

4.5.3.2. Ejemplos de Predicción

La Figura 4.22 muestra una predicción realizada por el modelo en una de las imágenes del conjunto de prueba. Mostrando que en aspectos generales logra distinguir los edificios del fondo y clasifica correctamente los edificios como destruidos.

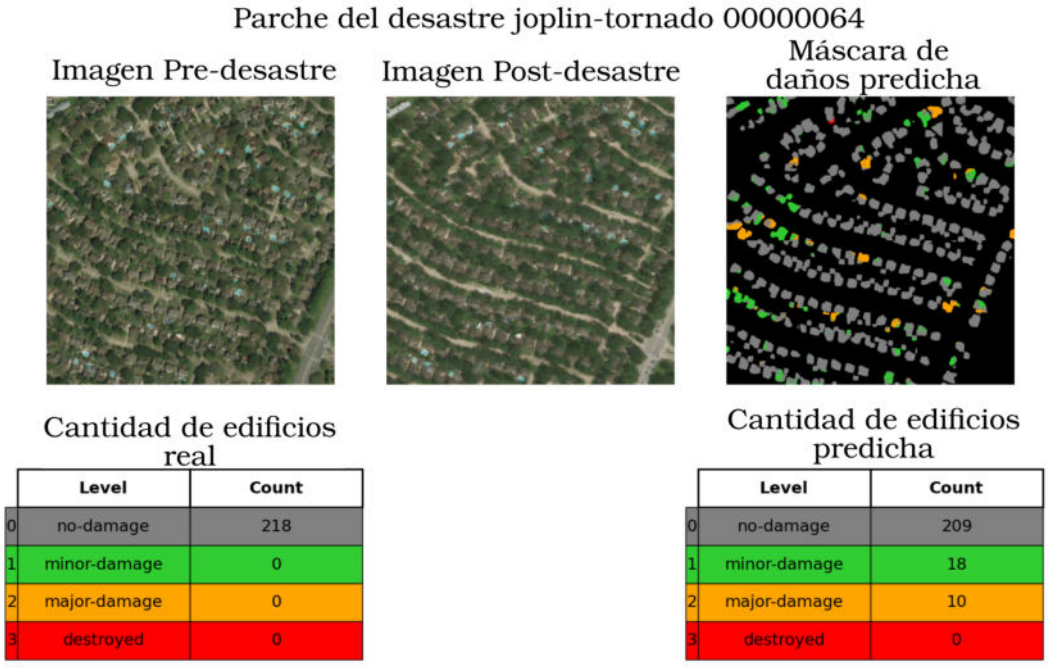


Figura 4.22: Ejemplo de predicción del modelo del experimento 5 con 50 épocas sobre una imagen del conjunto de prueba.

4.5.4. Discusión

De este último experimento, se derivan las siguientes conclusiones principales:

- **Límite en la capacidad de inferencia de la arquitectura**: En este último experimento se han utilizado todas las imágenes que contenían edificios del conjunto de datos xBD. A pesar del incremento considerable en la cantidad de imágenes utilizadas para entrenamiento la capacidad de generalización del modelo no ha incrementado en comparación con el entrenamiento realizado

en el experimento 4. Este comportamiento nos permite concluir que el desempeño en la tarea de clasificación de daños de la arquitectura no mejorará con el incremento en la cantidad de datos de entrenamiento.

- **Clase Minor Damage**: El modelo continúa mostrando un desempeño del modelo en la tarea de clasificación de la clase *minor-damage*. Podemos concluir que esto se debe a que la clase presenta características que se solapan con la clase *no-damage* principalmente. Esto se acentúa al incrementar la cantidad de imágenes del conjunto de entrenamiento.
- **Persistencia del sobreajuste**: A pesar de incorporar un conjunto de datos significativamente más amplio, el modelo sigue sobreajustando rápidamente al conjunto de entrenamiento durante las primeras épocas de entrenamiento, lo que impide una mejora sustancial en el desempeño sobre los conjuntos de validación y prueba.
- **Buen desempeño para clases *no-damage* y *destroyed***: La arquitectura ha permitido que el modelo entrenado presente un desempeño aceptable en la tarea de detección y clasificación de daños especialmente en las clases *no-damage* y *destroyed*.
- **Desempeño considerablemente bueno en la tarea de segmentación**: Con el incremento en la cantidad de imágenes de entrenamiento, y habiendo utilizado todo el conjunto de datos xBD podemos ver que el desempeñó de la arquitectura para la tarea de detección y segmentación de edificios en la imagen satelitales es considerablemente bueno y ha tenido un pequeño incremento con respecto al experimento 4 hasta llegar a un 90% tanto en validación como sobre el conjunto de prueba.

4.6. Comparación con el estado del arte

La Tabla 4.38 muestra cuál es el desempeño del modelo entrenado en cada experimento sobre el conjunto de validación y prueba. Se puede observar que de manera secuencial el desempeño del modelo mejoró después de cada experimento, llegando hasta el experimento 4 donde obtenemos el modelo con mejor desempeño.

		Conjunto de Validación		Conjunto de Prueba	
Experimentos	**Mejor época**	**HMF1 de Segmentación**	**HMF1 de Clasificación**	**HMF1 de Segmentación**	**HMF1 de Clasificación**
EXP1_C0	98	0,7754	0,0000	-	-
EXP2_C1	7	0,5845	0,0000	-	-
EXP3_C3	71	0,8824	**0,6341**	0,8705	0,5571
EXP4_C0	40	0,8870	0,5701	0,8880	**0,5776**
EXP5_C0	24	**0,9042**	0,5780	**0,9063**	0,5536

Tabla 4.38: Resumen comparativo del desempeño del modelo obtenido de cada experimento.

La Tabla 4.39 nos permite comparar el desempeño de nuestro modelo en comparación con los demás trabajos en el estado del arte. Es importante destacar

que cada trabajo se hace uso del conjunto de datos xBD, pero se utilizan diferentes arquitecturas, muestras y técnicas para tratar el desbalance entre las clases.

A pesar de que la arquitectura utilizada en esta tesis no cuenta con mecanismos de atención, el rendimiento del modelo entrenado es equiparable al propuesto en el trabajo Hao et al. [9]. Los trabajos comparables con el desarrollado en esta tesina EXP4_C0 de grado [9] y [55] obtienen valores similares a los reportados aquí, con lo que se demuestra la capacidad del modelo desarrollado para obtener buenas soluciones

A continuación, se describen los factores que limitan el desempeño del modelo en comparación con los trabajos más avanzados en el estado del arte:

Cantidad de imágenes los trabajos relacionados utilizan un número de imágenes significativamente mayor (cuadriplican el número de imágenes) [10, 23]. Incluso, los últimos trabajos del estado del arte, hacen uso de todo el conjunto de datos xBD y otros conjuntos para validación y prueba como xFDB, IDA-BD, imágenes de Turquía y demás. Además de utilizar aumentación de datos para mejorar la capacidad de generalización. [57–60].

Arquitectura El modelo utilizado en esta tesis no posee mecanismos de atención. Para alcanzar un rendimiento comparable con el estado del arte, sería necesario modificar la arquitectura para agregar mecanismos de atención [9], un módulo de múltiple escala [53] o realizar la clasificación de daños utilizando Transformers [58–60]. La implementación de estos cambios para mejorar la capacidad de generalización del modelo presentan una gran complejidad técnica y requieren de gran cantidad de poder de cómputo que no se encuentra disponible debido a su alto costo.

Tiempo de ejecución La implementación actual únicamente corre en una sola Unidad de Procesamiento Gráfico (GPU, *Graphics Processing Unit*). Modificar la implementación para aprovechar el paralelismo de varias GPU's, permitiría disminuir el tiempo de entrenamiento necesario y aumentar la cantidad de imágenes de entrenamiento, pero aumentado la complejidad de implementación en consecuencia.

Esquema de entrenamiento multi-tarea Los últimos trabajos han demostrado que un enfoque de entrenamiento en dos fases permite un entrenamiento más controlado para la tarea de segmentación y clasificación, permitiendo una mejora del desempeño del modelo Shen et al. [10], Chen et al. [60].

Tratamiento del desbalance con poca escalabilidad La técnica de muestreo voraz funciona bien cuando el objetivo es la selección de pocas imágenes del conjunto xBD. A medida que la cantidad de imágenes a seleccionar crece, la efectividad disminuye, resultando en muestras del conjunto de datos xBD cada vez con mayor desbalance. Es por esto que, con el fin se aumenta la cantidad de imágenes de entrenamiento, es necesario cambiar a otras técnicas como las que se proponen en [10, 24, 58, 60] o agregar nuevas imágenes que tengan mayor cantidad de edificios con diferentes tipos de daño.

Sobre ajuste El último experimento demuestra que el modelo sobreajusta muy rápidamente en las primeras 50 épocas de entrenamiento, lo cual resulta contraproducente para el aprendizaje de la tarea de clasificación de daños. Para esto existen dos alternativas, se podría realizar aprendizaje en dos etapas, o bien, implementar métodos de regularización para evitar el sobreajuste en la arquitectura actual. Siempre pensando en una gran cantidad de recursos computacionales necesarios para ejecutar estas etapas.

Nombre	**Cantidad de parches**	**HMF1 de clasificación**	**F1 de segmentación**	**F1 de clases**			
				no-damage	*minor-damage*	*major-damage*	*destroyed*
Gupta et al. [22]	9 168	0,0342	-	0,6631	0,1435	0,0094	0,4557
Hao et al. [9]	2 799	0,5500	0,7400	-	-	-	-
Zhao y Zhang [55]	12 030	0,5732	-	0,7140	0,3955	0,6037	0,7181
Shen et al. [10]	9 168+	0,7820	0,8640	0,9250	0,6160	0,7880	0,8760
May et al. [23]	9 168	0,5977	0,8130	0,7900	0,4120	0,6350	0,7020
Wang et al. [24]	385	0,8102	-	0,9977	0,7507	0,7407	
Kaur et al. [58]	9 168	0,7960	0,8250	0,978	0,7110	0,7650	0,7720
Dong y Zhao [59]	9 168	0.7160	0.8190	0,9450	0,5290	0,7190	0,8010
Risso et al. [57]	9 168	0,4400	-	0,5300	0,3500	0,4800	
Chen et al. [60]	9 168	0,7800	0,8600	0,9280	0,6470	0,7720	0,8490
EXP4_C0	2 401	0,5776	0,8880	0,5289	0,3586	0,5262	0,7208

Tabla 4.39: Comparación de resultados con el estado del arte.

Conclusión de la sección de resultados: Se logró un incremento en el desempeño del modelo a lo largo del experimento, alcanzando niveles comparables con los trabajos presentados en la literatura. Este avance se logró a pesar de enfrentar desafíos significativos, como el notable desbalance en el conjunto de datos xBD, la complejidad técnica de implementar soluciones para el problema de CDEN, y la limitación de recursos de cómputo disponibles. Estos factores dificultan la implementación de mecanismos avanzados, como los utilizados en los trabajos más recientes. Además, la aplicabilidad de una solución para el CDEN en Argentina depende de la creación de un conjunto de datos nuevo más representativo de las características de las ciudades de la región.

Capítulo 5

Conclusiones, desafíos y perspectivas futuras

Este capítulo contiene las conclusiones principales del presente Trabajo de Fin de Grado. Se resumen los objetivos cumplidos (Sección 5.1), los problemas y desafíos encontrados (Sección 5.2) y los posibles trabajos futuros (Sección 5.3).

5.1. Conclusiones

El objetivo general de esta tesina final de carrera fue evaluar algoritmos *machine learning* y, en particular, de *Deep Learning* para la segmentación de imágenes satelitales a fin de detectar daños producidos por un evento natural (sismos, incendios forestales, inundaciones, huracanes, temporales, etc.) que afectan y/o destruyen edificios y casas en las ciudades. Para esto se implementó una arquitectura de red neuronal profunda siamesa y se evaluó su desempeño utilizando el dataset xBD del estado del arte.

En relación con los objetivos planteados al principio de este documento podemos concluir:

- Con respecto a *entender, aplicar y evaluar diferentes modelos algorítmicos existentes en la literatura para la detección y clasificación de daños en imágenes satelitales*, se ha realizado una exploración del estado del arte sobre el uso de modelos de DL para dar solución al problema de CDEN. Se han analizado los aspectos que caracterizan los diferentes enfoques y arquitecturas, permitiendo la implementación de técnicas para el entrenamiento del modelo y para el procesamiento de los datos, adaptados a las limitaciones del contexto en el que se ha desarrollado el proyecto.
- En relación con *entender, aplicar y evaluar el impacto de diferentes técnicas en la fase de entrenamiento y cómo afectan la capacidad de inferencia del modelo*, no fue posible realizar una comparativa justa y completa de rendimiento entre el modelo entrenado en este proyecto y los del estado del arte, debido a la imposibilidad de utilizar el conjunto de datos xBD en su totalidad y los

recursos computacionales disponibles. Sin embargo, a través de diversos experimentos y la exploración de hiperparámetros, se logró evaluar el impacto de las distintas técnicas implementadas en el desempeño del modelo. Igualmente, los resultados del modelo propuesto fueron comparados con el estado del arte con características similares a las del presente trabajo obteniendo resultados similares.

- En cuanto a *la aplicación de algoritmos en un prototipo de sistema que permite ingresar dos imágenes satelitales (pre y post desastre) de una zona urbana y clasificar el nivel de daño de las estructuras edilicias*, el desarrollo del prototipo ha sido fluido y ha demostrado la efectividad de las predicciones del modelo, así como los resultados del trabajo de postprocesamiento. Este prototipo presenta un enfoque viable para abordar el problema de CDEN y demuestra la posibilidad de desarrollar otras aplicaciones utilizando un enfoque similar.

5.2. Dificultades encontradas

La implementación de este proyecto ha superado satisfactoriamente diversas dificultades surgidas durante su desarrollo. Algunas de estas dificultades son inherentes al problema en cuestión, mientras que otras están relacionadas con el contexto en el que se ha llevado a cabo. A continuación, se enumeran algunas de las principales complicaciones encontradas:

- *Limitación en el poder de cómputo disponible*: La Facultad de Ingeniería de la Universidad Nacional de Cuyo no dispone de un clúster adaptado a las necesidades de los estudiantes e investigadores. El entrenamiento de modelos de AI, especialmente aquellos basados en DL, demanda recursos de hardware especializados, principalmente por la intensidad computacional y el tiempo requerido. El uso de GPUs es fundamental para procesar datos y entrenar redes neuronales en un tiempo razonable. Para resolver esta limitación, desde el inicio del proyecto se solicitó acceso al clúster *TOKO* de la Facultad de Ciencias Exactas de la misma universidad. Aunque no contaba con GPUs, sí ofrecía una considerable cantidad de procesadores, lo que resultó clave para realizar los primeros ensayos. Como se observó en el Experimento 1, aunque es posible entrenar redes sin hardware especializado, el tiempo requerido es significativamente mayor. Dado que se evidenció la necesidad de aprovechar las capacidades de paralelismo que brinda el entrenamiento en GPU, se gestionó el acceso al clúster Mendieta, propiedad del Centro de Computación de Alto Desempeño de la Universidad Nacional de Córdoba. Una vez obtenido dicho acceso, fue posible continuar con el desarrollo de los experimentos de manera más eficiente.

- *Técnicas para tratar el desbalance de datos*: Uno de los desafíos más complejos e inherentes a las características del CDEN fue el fuerte desbalance del conjunto de datos xBD en varias dimensiones. En los trabajos del estado del arte, se utilizan enfoques diversos, a menudo sin tener en cuenta el tamaño final del conjunto de entrenamiento. El tiempo de ejecución necesario para el

entrenamiento de modelos de AI está estrechamente ligado al tamaño del conjunto de datos. Para abordar este problema, en este proyecto se plantearon dos estrategias basadas en métodos observados en el estado del arte, adaptadas para mantener una cantidad razonable de imágenes. Por ello, a lo largo de los experimentos se empleó inicialmente un conjunto de datos reducido, utilizando aumentación de datos, y posteriormente un subconjunto específico del conjunto total. Finalmente, se diseñó un algoritmo voraz que construye un subconjunto de imágenes balanceadas lo mejor posible para ser utilizadas por cualquier modelo para su entrenamiento y evaluación. Más aún este algoritmo voraz puede ser utilizado en otros problemas con conjuntos de datos desbalanceados relacionados con imágenes digitales.

- *Complejidad técnica*: La implementación del pipeline de entrenamiento de un modelo de DL implicó un alto nivel de complejidad técnica, logrado tras una extensa inversión de tiempo en un proceso iterativo de prueba y error. Dado que no se contaba con capacidad de procesamiento local y se dependía de recursos remotos, cada prueba de funcionamiento requirió tiempos de espera prolongados, que oscilaron entre 1 y 5 días.

5.3. Trabajos futuros

El trabajo realizado en este proyecto abre diversas líneas de investigación que podrían mejorar y expandir los resultados obtenidos, así como aplicar las soluciones desarrolladas a otros problemas. A continuación, se presentan algunas de las direcciones que serían interesantes explorar en futuros trabajos:

- **Paralelización del entrenamiento con múltiples GPUs**: Para mejorar el rendimiento y reducir los tiempos de entrenamiento, se propone la implementación de paralelismo utilizando múltiples GPUs en paralelo. Esta optimización permitiría una mayor eficiencia en la capacitación de modelos más complejos y el manejo de conjuntos de datos más grandes.

- **Uso de imágenes satelitales específicas para la región**: Un paso importante sería la utilización de conjuntos de datos que contengan imágenes satelitales con características representativas de las ciudades en Argentina. Esto podría mejorar la generalización del modelo y hacer que sea más aplicable a la realidad local, adaptándose a las particularidades geográficas y urbanísticas de la región.

- **Exploración de arquitecturas basadas en Transformers**: En la búsqueda de mejorar la precisión y eficiencia de las soluciones, se podría investigar la implementación de arquitecturas basadas en *Transformers*, similares a las que se han utilizado en los últimos trabajos del estado del arte. Estas arquitecturas, conocidas por su capacidad de manejar grandes cantidades de datos y su rendimiento en tareas de visión por computadora, podrían aportar mejoras significativas al modelo.

- **Análisis de la clase *minor-damage***: A pesar de los avances logrados, la clasificación de la clase *minor-damage* sigue siendo un desafío. Es importante investigar las características de esta clase que dificultan su correcta clasificación incluso utilizando otras arquitecturas. Comprender la causa de este problema permitiría desarrollar soluciones específicas para mejorar el rendimiento en esta categoría.

- **Enfoques avanzados en detección de objetos**: Otra línea de investigación prometedora sería la implementación de enfoques avanzados de detección de objetos, como *Mask R-CNN* o *YOLO*. Estas técnicas permiten la detección y segmentación simultánea de objetos, lo que resulta especialmente adecuado para tareas como la segmentación de daños en edificios. Incorporar estos métodos podría ofrecer una evaluación más precisa del desempeño del modelo para resolver problemas del CDEN en situaciones del mundo real.

- **Aplicaciones en otros problemas de comparación de imágenes**: La arquitectura utilizada en este proyecto podría extenderse a otros problemas relacionados con la comparación de imágenes a lo largo del tiempo. Ejemplos de aplicaciones futuras incluyen la detección de deshielo en glaciares o el monitoreo de cambios en el caudal de ríos. Con un conjunto de datos adecuado, estas extensiones podrían abrir nuevas oportunidades de investigación en campos relacionados con la observación y monitoreo de cambios ambientales.

- **Conjunto de datos de urbes en Argentina**: El conjunto de datos xBD se compone de imágenes satelitales de ciudades y zonas ubicadas en América del Norte, lo que podría introducir un sesgo en los modelos entrenados, limitando su desempeño en la tarea de CDEN cuando se aplican a imágenes de nuestra región. Esto se debe, en particular, a las características distintivas en la disposición de viviendas y edificios en ciudades como Mendoza. Por ello, un posible trabajo futuro sería la creación de un conjunto de datos basado en imágenes satelitales de desastres naturales específicamente orientado a Argentina.

En resumen, existen diversas oportunidades para expandir este trabajo, ya sea mejorando la precisión y eficiencia de los modelos mediante nuevas técnicas o aplicando las soluciones desarrolladas a otros problemas de relevancia social y científica.

Durante el desarrollo de esta tesina final de grado se logró aplicar los conocimientos adquiridos durante los cinco años de la carrera y abordar un nuevo problema de relevancia social y económica. Por otro lado, se diseñó una nueva técnica voraz para extraer del conjunto de datos imágenes representativas para el problema abordado. Todo esto me ha ayudado a amalgamar los conceptos vistos durante la carrera, enriqueciendo mi perfil profesional y preparándome para mi futuro como Licenciado en Ciencias de la Computación.

Apéndice A

Anexos

En este capítulo se presenta un conjunto de documentos y materiales adicionales que complementan y respaldan los resultados y hallazgos expuestos a lo largo del trabajo. Estos anexos incluyen, pero no se limitan a, tablas, gráficos, códigos, y descripciones detalladas de los experimentos realizados. La información contenida en este apartado es fundamental para una comprensión más profunda del análisis llevado a cabo y para la replicación de los experimentos en futuros estudios.

A.1. Repositorio de código

Todo el código utilizado para el entrenamiento del modelo y el desarrollo de la página web se encuentra disponible en el repositorio de GitHub, al cual se puede acceder a través del siguiente enlace: https://gitfront.io/r/Mrtc101/k4gzPrpN1AZB/Thesis-DL-for-BDA/.

Bibliografía

[1] T. Spencer, "How helene became the near-perfect storm to bring widespread destruction across the south," 29 de Septiebre 2024, accedido: 04-10-2024. Disponible en: https://apnews.com/article/hurricane-helene-florida-georgia-carolina-268ba170519c52c2bc1abcbc0b093e53

[2] R. RMM, "Japón se prepara para otro "gran terremoto", tras el impacto de uno de magnitud 7,1," 14 de agosto de 2024, accedido: 04-10-2024. Disponible en: https://reddemediosmisiones.com.ar/contenido/54549/japon-se-prepara-para-otro-gran-terremoto-tras-el-impacto-de-uno-de-magnitud-71

[3] DW, "Bosnia: Death toll mounts after flooding and landslides," 4 de Octubre 2024, accedido: 04-10-2024. Disponible en: https://www.dw.com/en/bosnia-death-toll-mounts-after-flooding-and-landslides/a-70408421

[4] M. P. Gallardo, "Suramérica bajo fuego: alerta en varios países por récord de incendios," 14 de septiembre 2024, accedido: 04-10-2024. Disponible en: https://www.france24.com/es/am%C3%A9rica-latina/20240914-suram%C3%A9rica-bajo-fuego-alerta-en-varios-pa%C3%ADses-por-r%C3%A9cord-de-incendios

[5] W. H. Organization *et al.*, "Protocolos de evaluación sanitaria rápida en situaciones de emergencia," Organización Mundial de la Salud, Informe técnico, 1999.

[6] M. Markus, F. Fiedrich, F. Gehbauer, y S. Hirschberger, "Strong earthquakes, rapid damage assessment and rescue planning," en *Proc. of the 7th Annual Conference of the International Emergency Management Society:"Contingency, Emergency, Crisis, and Disaster Management: Emergency Management in the Third Millennium", edited by Kowalski, KM and Trevits, MA, Orlando, Florida*, 2000, pp. 369–378.

[7] V. Poggi, C. Scaini, L. Moratto, G. Peressi, P. Comelli, P. L. Bragato, y S. Parolai, "Rapid damage scenario assessment for earthquake emergency management," *Seismological Research Letters*, vol. 92, n.º 4, pp. 2513–2530, 2021.

[8] P. Ge, H. Gokon, y K. Meguro, "A review on synthetic aperture radar-based building damage assessment in disasters," *Remote Sensing of Environment*, vol. 240, p. 111693, 2020.

[9] H. Hao, S. Baireddy, E. R. Bartusiak, L. Konz, K. LaTourette, M. Gribbons, M. Chan, E. J. Delp, y M. L. Comer, "An attention-based system for damage assessment using satellite imagery," en *2021 IEEE International Geoscience and Remote Sensing Symposium IGARSS*. IEEE, 2021, pp. 4396–4399.

[10] Y. Shen, S. Zhu, T. Yang, C. Chen, D. Pan, J. Chen, L. Xiao, y Q. Du, "Bdanet: Multiscale convolutional neural network with cross-directional attention for building damage assessment from satellite images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2021.

[11] D. S. O. ELEKS, "Deep learning for damage detection using satellite images," visitado en junio 2024. Disponible en: https://eleks.com/research/deep-learning-for-damage-detection-using-satellite-images/

[12] S. May, A. Dupuis, A. Lagrange, F. D. Vieilleville, y C. Fernandez-Martin, "Building damage assessment with deep learning," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, vol. 43, pp. 1133–1138, 5 2022.

[13] C. G. Rafael y E. W. Richard, *Digital image processing*. Pearson education, 2018.

[14] E. Chuvieco, *Fundamentals of satellite remote sensing: An environmental approach*. CRC press, 2020.

[15] "Well-known text representation of coordinate," accedido: 04-10-2024. Disponible en: https://www.ogc.org/standard/wkt-crs/

[16] El Código ASCII, "El código ascii - tabla de caracteres y símbolos," 2024, accedido: 09-oct-2024. Disponible en: https://elcodigoascii.com.ar/

[17] "Introducing json," accedido: 04-10-2024. Disponible en: https://www.json.org/json-en.html

[18] W. Li, H. Huang, L. Ma, y H. Wei, "Object detection in remote sensing imagery using deep learning: A survey," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 58, n.º 5, pp. 3549–3568, 2020.

[19] R. Szeliski, *Computer vision: algorithms and applications*. Springer Nature, 2022.

[20] L. Sun, H. Zou, J. Wei, X. Cao, S. He, M. Li, y S. Liu, "Semantic segmentation of high-resolution remote sensing images based on sparse self-attention and feature alignment," *Remote Sensing*, vol. 15, n.º 6, 2023. Disponible en: https://www.mdpi.com/2072-4292/15/6/1598

[21] T. Wu, Y. Hu, L. Peng, y R. Chen, "Improved anchor-free instance segmentation for building extraction from high-resolution remote sensing images," *Remote Sensing*, vol. 12, n.º 18, 2020. Disponible en: https://www.mdpi.com/2072-4292/12/18/2910

[22] R. Gupta, B. Goodman, N. Patel, R. Hosfelt, S. Sajeev, E. Heim, J. Doshi, K. Lucas, H. Choset, y M. Gaston, "Creating xbd: A dataset for assessing building damage from satellite imagery," en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2019, pp. 10–17.

[23] S. May, A. Dupuis, A. Lagrange, F. De Vieilleville, y C. Fernandez-Martin, "Building damage assessment with deep learning," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 43, pp. 1133–1138, 2022.

[24] Y. Wang, A. W. Z. Chew, y L. Zhang, "Building damage detection from satellite images after natural disasters on extremely imbalanced datasets," *Automation in construction*, vol. 140, p. 104328, 2022.

[25] D. L. P. A. K. Mackworth, *Artificial Intelligence: Foundations of Computational Agents, 3rd Edition*, 3r ed. Cambridge University Press, 2024.

[26] I. Goodfellow, Y. Bengio, y A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[27] G. James, D. Witten, T. Hastie, R. Tibshirani *et al.*, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[28] H. Dong, Z. Ding, y S. Zhang, *Deep Reinforcement Learning: Fundamentals, Research and Applications*. Springer Nature Singapore, 2020. Disponible en: https://books.google.com.ar/books?id=31juDwAAQBAJ

[29] S. H. Park, J. M. Goo, y C.-H. Jo, "Receiver operating characteristic (roc) curve: practical review for radiologists," *Korean journal of radiology*, vol. 5, n.º 1, pp. 11–18, 2004.

[30] R. Rifkin y A. Klautau, "In defense of one-vs-all classification," *Journal of Machine Learning Research*, vol. 5, pp. 101–141, 2004.

[31] X. Ying, "An overview of overfitting and its solutions," en *Journal of physics: Conference series*, vol. 1168. IOP Publishing, 2019, p. 022022.

[32] *A survey of cross-validation procedures for model selection*, 2010.

[33] L. Prechelt, "Early stopping-but when?" en *Neural Networks: Tricks of the trade*. Springer, 2002, pp. 55–69.

[34] P. Zhang, "Model Selection Via Multifold Cross Validation," *The Annals of Statistics*, vol. 21, n.º 1, pp. 299 – 313, 1993. Disponible en: https://doi.org/10.1214/aos/1176349027

[35] C. Arnold, L. Biedebach, A. Küpfer, y M. Neunhoeffer, "The role of hyperparameters in machine learning models and how to tune them," *Political Science Research and Methods*, pp. 1–8, 2023.

[36] K. Abhishek y M. Abdelaziz, *Machine Learning for Imbalanced Data: Tackle imbalanced datasets using machine learning and deep learning techniques*. Packt Publishing, 2023. Disponible en: https://books.google.com.ar/books?id=xx_nEAAAQBAJ

[37] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, y Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," en *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6023–6032.

[38] A. L. Jaimes, S. Z. Martınez, C. A. C. Coello *et al.*, "An introduction to multiobjective optimization techniques," *Optimization in Polymer Processing*, vol. 1, p. 29, 2009.

[39] W. Li, T. Zhang, R. Wang, S. Huang, y J. Liang, "Multimodal multi-objective optimization: Comparative study of the state-of-the-art," *Swarm and Evolutionary Computation*, vol. 77, p. 101253, 2023.

[40] T. H. Cormen, C. E. Leiserson, R. L. Rivest, y C. Stein, *Introduction to algorithms*. MIT press, 2022.

[41] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, y N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, n.º 7, pp. 579–588, 2009.

[42] C. M. Bishop y H. Bishop, *Deep learning: Foundations and concepts*. Springer Nature, 2023.

[43] A. Zhang, Z. C. Lipton, M. Li, y A. J. Smola, *Dive into deep learning*. Cambridge University Press, 2023.

[44] B. Hanin, "Which neural net architectures give rise to exploding and vanishing gradients?" *Advances in neural information processing systems*, vol. 31, 2018.

[45] Caleb Robinson , "building-damage-assessment-cnn-siamese," visitado en junio de 2024. Disponible en: https://github.com/microsoft/building-damage-assessment-cnn-siamese

[46] L. E. Sucar y G. Gómez, "Visión computacional," *Instituto Nacional de Astrofísica, Óptica y Electrónica. México*, 2011.

[47] F. G. Palomares, J. A. Monsoriu, y E. Alemany, "Aplicación de la convolución de matrices al filtrado de imágenes," *Modelling in Science Education and Learning*, vol. 9, n.º 1, pp. 97–108, 2016.

[48] N. Siddique, S. Paheding, C. P. Elkin, y V. Devabhaktuni, "U-net and its variants for medical image segmentation: A review of theory and applications," *IEEE access*, vol. 9, pp. 82 031–82 057, 2021.

[49] P. Ulmas y I. Liiv, "Segmentation of satellite imagery using u-net models for land cover classification," *arXiv preprint arXiv:2003.02899*, 2020.

[50] Z. Shaukat, Q. u. A. Farooq, S. Tu, C. Xiao, y S. Ali, "A state-of-the-art technique to perform cloud-based semantic segmentation using deep learning 3d u-net architecture," *BMC bioinformatics*, vol. 23, n.º 1, p. 251, 2022.

[51] M. Zhang, Z. Liu, J. Feng, L. Liu, y L. Jiao, "Remote sensing image change detection based on deep multi-scale multi-attention siamese transformer network," *Remote Sensing*, vol. 15, n.º 3, p. 842, 2023.

[52] B. Fang, L. Pan, y R. Kou, "Dual learning-based siamese framework for change detection using bi-temporal vhr optical remote sensing images," *Remote Sensing*, vol. 11, n.º 11, p. 1292, 2019.

[53] H. Chen, C. Wu, B. Du, y L. Zhang, "Change detection in multi-temporal vhr images based on deep siamese multi-scale convolutional networks," *arXiv preprint arXiv:1906.11479*, 2019.

[54] S. Ji, Y. Shen, M. Lu, y Y. Zhang, "Building instance change detection from large-scale aerial images using convolutional neural networks and simulated samples," *Remote Sensing*, vol. 11, n.º 11, p. 1343, 2019.

[55] F. Zhao y C. Zhang, "Building damage evaluation from satellite imagery using deep learning," en *2020 IEEE 21st International Conference on Information Reuse and Integration for Data Science (IRI)*. IEEE, 2020, pp. 82–89.

[56] M. Papadomanolaki, M. Vakalopoulou, y K. Karantzalos, "A deep multitask learning framework coupling semantic segmentation and fully convolutional lstm networks for urban change detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 59, n.º 9, pp. 7651–7668, 2021.

[57] M. Risso, A. Goffi, B. A. Motetti, A. Burrello, J. B. Bove, E. Macii, M. Poncino, D. J. Pagliari, y G. Maffeis, "Building damage assessment in conflict zones: A deep learning approach using geospatial sub-meter resolution data," *arXiv preprint arXiv:2410.04802*, 2024.

[58] N. Kaur, C.-C. Lee, A. Mostafavi, y A. Mahdavi-Amiri, "Large-scale building damage assessment using a novel hierarchical transformer architecture on satellite images," *Computer-Aided Civil and Infrastructure Engineering*, vol. 38, n.º 15, pp. 2072–2091, 2023.

[59] C. Dong y X. Zhao, "Bi-daunet: Leveraging biformer in a unet-like architecture for building damage assessment," en *Journal of Physics: Conference Series*, vol. 2833, n.º 1. IOP Publishing, 2024, p. 012015.

[60] F. Chen, Y. Sun, L. Wang, N. Wang, H. Zhao, y B. Yu, "Hrtbda: a network for post-disaster building damage assessment based on remote sensing images," *International Journal of Digital Earth*, vol. 17, n.º 1, p. 2418880, 2024.

[61] H. Xia, J. Wu, J. Yao, H. Zhu, A. Gong, J. Yang, L. Hu, y F. Mo, "A deep learning application for building damage assessment using ultra-high-resolution

remote sensing imagery in turkey earthquake," *International Journal of Disaster Risk Science*, vol. 14, n.º 6, pp. 947–962, 2023.

[62] xView, "xview2 web page," visitado en junio de 2024. Disponible en: https://xview2.org/

[63] "Github web page," accedido: 04-10-2024. Disponible en: https://github.com/