UNIVERSIDAD
DE LA REPUBLICA
URUGUAY

Aprendizaje profundo para la extracción de edificios en ciudades sin planificación urbana

Lucas González Petti

Tutores:

Sergio Nesmachnow

Jamal Toutouh

Programa de Grado en Ingeniería en Computación
Facultad de Ingeniería
Universidad de la República

Montevideo – Uruguay
Junio de 2022

RESUMEN

Este proyecto estudia el problema de extracción de edificios, como subtarea de segmentación de imágenes, sobre ciudades o poblados con poca o nula planificación urbana. Para abordar del problema se seleccionaron las arquitecturas de redes neuronales profundas FCN, ResNet (y sus variantes) y U-Net, en conjunto con una serie de hiperparámetros (función de pérdida, tasa de aprendizaje) con el fin de encontrar los modelos que mejor se desempeñan en la tarea. Para este propósito se diseñó un proceso de selección progresivo y eliminatorio dividido en tres etapas y con dos criterios distintos de selección. Se definieron dos etapas adicionales que consisten en la calibración de los modelos elegidos y en el aplicado de votación de modelos para la clasificación de cada píxel de la imagen.

Los resultados experimentales muestran un mejor rendimiento en general de las arquitecturas basadas en U-Net y la combinación de entropía cruzada con Tversky focalizado como función de pérdida. Los mejores desempeños alcanzaron un IoU de $0,91$ para el dataset de San José de las Matas, que fue generado exclusivamente con el propósito de validación de este trabajo.

La obtención de resultados aceptables en la labor de extracción de edificios es el punto de partida para otras como la estimación de habitantes o la generación de la propia planificación urbana. En ese sentido, se tomó un área de estudio perteneciente a Montevideo (Uruguay) de la cual se tienen datos demográficos y se experimentó estimando la población de la zona a partir de los edificios extraídos por los modelos. El producto del proceso de estimación aplicado tuvo un error de un $11,7\,\%$ respecto a la estimación más fiable para esa área en particular.

Palabras claves:
aprendizaje automático, aprendizaje profundo, segmentación de imágenes, extracción de edificios, redes neuronales convolucionales, FCN, ResNet, U-Net.

Tabla de contenidos

Lista de figuras

Lista de tablas

Capítulo 1

Introducción

El advenimiento de remote sensing (teledetección) ha permitido la extracción de información de imágenes de alta resolución y altamente detalladas sobre el entorno que representan. En particular, la extracción de edificios sobre un área poblada puede dar información valiosa, no solo sobre la ubicación de cada edificio respecto a los otros, sino además sobre la población que allí habita, su planificación urbana o hasta su modelado ambiental. El conocimiento de estas características es especialmente importante en ciudades donde la planificación urbana es inexistente o donde los censos sobre su población se encuentran desactualizados. El problema es aún más grave en zonas con una gran cantidad de asentamientos informales o donde hay cambios demográficos rápidos, como es el caso de las ciudades emergentes.

Uno de los mecanismos más utilizados para censar una población es el de entrevista presencial en cada una de las viviendas del área de interés. Este procedimiento tiene un alto costo desde el punto de vista logístico, puesto que no solo requiere de una gran cantidad de trabajadores de campo que lleven a cabo las entrevistas, sino que además es necesario que la población esté físicamente en las viviendas para responderlas en el momento coordinado. Puede darse además el caso donde no se cense una zona determinada por falta de garantías en materia de seguridad para el censista que deba trasladarse hasta allí a cumplir con su tarea.

Aún cuando ya se ha implementado y puesto en práctica el método de respuesta en línea a la entrevista censal, lo cual reduce el costo logístico si lo comparamos con las presenciales, no se puede garantizar que todas las personas puedan responder por esta vía, dado que es necesario tener acceso a una computadora y conectividad a Internet.

El gran esfuerzo requerido para obtener la información, sumado al posterior costo de unificarla y procesarla tienen como consecuencia que los procesos censales en general se realicen una vez cada varios años para una zona dada, por lo que en áreas donde hay grandes cambios demográficos la información tiende a desactualizarse rápidamente.

El campo de la Inteligencia Artificial (IA) ha demostrado ser una buena alternativa en general para llevar a cabo tareas en diversas disciplinas que incluyen visión, habla y diálogo, decisiones y planificación, resolución de problemas, robótica y otras aplicaciones que permiten el autoaprendizaje. Durante el último tiempo, los sistemas basados en aprendizaje profundo han superado repetidamente el estado del arte en áreas como el ajedrez, la traducción automática y los automóviles autónomos. Es por este motivo que parece razonable plantear el problema de extracción de edificios a partir de una imagen como uno de aprendizaje profundo, para lo cual es necesario estudiar los mejores acercamientos al problema hasta el momento, así como los datos disponibles.

Si bien la cantidad de conjuntos de datos para la extracción de edificios está acotada y en la mayoría de los casos contienen datos sobre grandes ciudades urbanizadas, se valora como posible el explorar y llegar a un modelo de redes neuronales profundas capaz de generalizar ciertas características de los edificios, presentes tanto en ciudades con planificación urbana como en las que no la tienen.

Este trabajo propone valorar la aptitud de diversas arquitecturas de redes neuronales profundas en la tarea de extracción de edificios sobre una ciudad con poca o nula planificación urbana. Aún más, se pretendió que las características de la ciudad sobre las que se validaría el trabajo fueran disímiles a las de las ciudades que estaban presentes en el conjunto de entrenamiento. Es por esta razón que se trazó como un objetivo más el generar un dataset anotado para una nueva ciudad. Tomar ese enfoque fue beneficioso para comprobar la capacidad de pluralización de los modelos construidos. En este proyecto, la localidad de estudio fue el municipio San José de las Matas, República Dominicana.

Una vez la labor de identificación de edificios culminó, se esperaba poder tomar acciones con la información obtenida, como por ejemplo la ya mencionada estimación de la población o la planificación del saneamiento de aguas residuales sobre una ciudad sobre la que se cuente con información demográfica.

Las principales contribuciones de la investigación reportadas en este proyecto incluyen: i) la exploración de distintas variantes de arquitecturas de redes neuronales profundas para la tarea de extracción de edificios; ii) la implementación de mecanismos de entrenamiento, calibración y votación de modelos fácilmente ejecutables con la configuración de unos pocos parámetros; iii) la validación de los mejores modelos propuestos en el conjunto de datos creado manualmente con ese fin y iv) validación de la estimación de la población en una zona acotada sobre la base de las segmentaciones generadas por los modelos.

Este documento consta de siete capítulos, a continuación se describe cómo se organizan. El capítulo 1 contiene las principales motivaciones y objetivos del proyecto. En el capítulo 2 se exponen los conceptos teóricos que fundamentan este trabajo. En el capítulo 3 se presentan los trabajos relacionados y datos utilizados. En el capítulo 4 se describen los métodos utilizados para el estudio. El capítulo 5 contiene las técnicas de procesamiento de datos, modelos y bibliotecas de software utilizadas. En el capítulo 6 se exponen los resultados obtenidos y su análisis. Finalmente, en el capítulo 7 se exponen las conclusiones y las consideraciones finales del proyecto, así como las lineas de trabajo futuro.

Capítulo 2

Marco Teórico

El presente capítulo expone los conceptos teóricos que fundamentan este trabajo.

2.1. Visión Artificial

La visión artificial (computer vision en inglés) es la rama de la ciencia de la computación cuyo objetivo es el de crear sistemas autónomos que realicen tareas que los humanos pueden ejecutar haciendo uso de, al menos, su sistema visual [29]. El concepto de visión artificial se basa en el desarrollo de técnicas que permitan a las computadoras procesar una imagen a nivel de píxel y comprenderla. Técnicamente, las máquinas intentan recuperar información visual, manejarla e interpretar los resultados a través de algoritmos de software para ese fin. La comprensión en este contexto significa la transformación de imágenes visuales en descripciones del mundo que tienen sentido para los procesos involucrados y pueden provocar la acción apropiada. La comprensión de la imagen puede verse como la separación de la información simbólica de los datos de la imagen utilizando modelos construidos con la ayuda de la geometría, la física, la estadística y la teoría del aprendizaje [18].

El procesamiento de una imagen por parte de una máquina consiste en leerla como una matriz de píxeles, siendo el píxel la unidad más pequeña en la que se puede dividir una imagen. Cada píxel tiene un conjunto de valores que representan la presencia y la intensidad de los tres colores primarios: rojo, verde y azul, si la imagen es a color. Para el caso de imágenes monocromáticas, cada píxel tiene un valor entre 0 y 255, donde cuanto mayor es este número,

mayor es el nivel de oscuridad del píxel. La imagen digital, por tanto, se convierte en una matriz, y la visión artificial en un estudio de matrices. Mientras que los algoritmos de visión artificial más simples usan álgebra lineal para manipular las matrices, las aplicaciones complejas involucran operaciones como convoluciones con núcleos que se pueden aprender y reducción de muestreo a través de la agrupación.

Una tarea de visión artificial que tiene como objetivo resolver un problema puede requerir que la máquina realice primero varias subtareas más pequeñas que abordan ciertas partes de la solución. Las principales subtareas dentro de visión artificial son:

- Clasificación de imágenes: las tareas de clasificación de imágenes identifican imágenes que contienen un determinado tipo o clase de objetos, personas, lugares u otras características mediante el estudio de sus contenidos visuales.
- Detección de objetos: esta tarea implica no solo la identificación de uno o más tipos de objetos en una imagen dada, sino también su ubicación. Su objetivo es determinar exactamente dónde están presentes los objetos en la imagen y generalmente se les asignan cuadros delimitadores para indicar su posición.
- Segmentación de imágenes: esta tarea puede verse como una versión más precisa de la detección de objetos que identifica los bordes exactos de los objetos. Realiza un análisis píxel por píxel de una imagen para determinar qué píxeles están ocupados por qué objetos o regiones particulares.
- Reconocimiento de video: es una extrapolación del reconocimiento de objetos basado en imágenes a datos de video. En el contexto de reconocimiento de video se puede utilizar el análisis de varios fotogramas consecutivos para obtener una mejor comprensión o contexto de lo que sucede en la imagen.
- Seguimiento de objetos: su objetivo es que la máquina no solo detecte diferentes tipos de objetos dentro de la imagen inicial (cuadro), sino que también rastree su presencia y trayectoria en los cuadros posteriores de un video.

Dentro de los ejemplos de aplicación más destacados de visión artificial se encuentran el reconocimiento facial, de gestos, objetos, estimación de distancia

y posición de un objeto y búsqueda en imágenes. Un factor clave para el crecimiento de las aplicaciones de visión artificial es la gran cantidad de información visual que fluye desde los teléfonos inteligentes, los sistemas de seguridad, las cámaras de tránsito y otros dispositivos con instrumentos visuales.

2.2. Segmentación de imágenes

En términos generales, la segmentación de imágenes tiene como objetivo capturar los límites de los objetos de interés en una imagen. En una definición más formal [43], la segmentación de imágenes busca particionar una imagen en un conjunto de regiones disjuntas, de manera que:

1. Los píxeles que comparten regiones en la clasificación son homogéneos con respecto a cierta característica.
2. Los píxeles de regiones distintas son significativamente diferentes en referencia a esa misma característica.

Las tareas de segmentación de imágenes se pueden dividir en dos grandes categorías: segmentación semántica y segmentación de instancias. En la segmentación semántica, cada píxel pertenece a una clase particular, por ejemplo: "perro", "árbol", "automóvil". Cualquier píxel que pertenezca a cualquier automóvil se asigna a la misma clase de "automóvil". La segmentación de instancias va un paso más allá y separa objetos distintos que pertenecen a la misma clase. Concretamente, si hubiera dos autos en una imagen dada, a cada auto se le asignaría la etiqueta "automóvil", pero se le daría una anotación distinta porque son instancias diferentes de la clase.

En algunos escenarios, el localizar los píxeles que delimitan un objeto (segmentación) puede ser también una subtarea de otras más complejas, como es la clasificación en una subcategoría de esos objetos encontrados. Por ejemplo, si se tuviera la intención de identificar en una imagen a perros de determinada raza, una de sus subtareas podría ser identificar los límites del animal dentro de la imagen, y otra consideraría más en profundidad otros atributos como el color o tamaño para determinar la raza en concreto.

2.3. Aprendizaje automático

Una definición de aprendizaje automático es: "se dice que un programa aprende de una experiencia E con respecto a alguna clase de tareas T y medida de desempeño P, si su desempeño en las tareas de T, medido por P, mejora con la experiencia E" [53].

Uno de los tipos de aprendizaje automático es el supervisado, donde los datos de entrenamiento son una secuencia finita $S = ((x_1, y_1), ..., (x_n, y_n))$ de pares pertenecientes a $X \times Y$, donde y_i se denomina la etiqueta correspondiente a x_i. La salida del algoritmo es una función $h : X \to Y$, llamada hipótesis, cuyo objetivo es predecir $y \in Y$ para un $x \in X$ arbitrario. Para conseguirlo, se busca encontrar el modelo que mejor se ajuste a la función h, con el menor error posible según una función de pérdida $L : Y \times Y \to \mathbb{R}$ que mide cuán lejos está $h(x)$ de su respectivo y. Cuanto menor sea el valor de pérdida promedio, mejor será h (hipótesis).

Los valores del dominio de Y pueden ser continuos o discretos, lo que determinará si el problema de aprendizaje es de regresión o clasificación respectivamente. En el marco de este proyecto, la tarea de identificar edificios a partir de imágenes satelitales es un problema de aprendizaje automático de segmentación semántica de imágenes donde:

- E = Imágenes satelitales.
- T = Identificar los límites de los edificios en una imagen satelital.
- P = Métrica que representa qué tan buena fue la predicción. Un ejemplo es la métrica intersection over union, explicada en la sección 2.6.

Al tener cada imagen su contraparte anotada con los edificios ya identificados, se puede decir que el tipo de aprendizaje aplicado es supervisado. Cada píxel de cada imagen de entrada puede tomar conceptualmente dos valores en Y: si pertenece a un edificio o no. Al ser estos valores discretos entonces podemos afirmar que es un problema de clasificación.

2.4. Teorema "No hay almuerzo gratis"

El Teorema No Free Lunch (NFL por sus siglas en inglés) indica que "el rendimiento esperado de cualquier par de algoritmos de optimización en todos los problemas posibles es idéntico" [88]. Dicho de otra forma, no existe ningún

algoritmo que supere a los demás en todo el dominio de los problemas. Esto significa que la elección del algoritmo más apropiado depende del problema específico que se está abordando y, si un algoritmo tiene un buen desempeño para cierta tarea, puede no tenerlo para otras.

En el contexto de aprendizaje automático el teorema NFL también aplica: no hay un modelo que funcione mejor para todos los escenarios y muchas veces parte del problema de aprendizaje automático es explorar los distintos algoritmos (y ajustar sus hiperparámetros) hasta encontrar aquel que mejor se desempeñe para la tarea en cuestión.

En el presente proyecto se experimentó con distintos modelos, donde basándose en las arquitecturas FCN y U-Net, se variaron los encoders entre ResNet34, ResNet50, ResNet101 y ResNext50 para obtener siete modelos iniciales con los que experimentar. Además, se exploró más de una función de costo, tasa de aprendizaje y proporción de los datasets originales a considerar durante la etapa de entrenamiento. En el capítulo 6 se desarrollan los criterios y elecciones acerca de los hiperparámetros considerados en cada etapa.

2.5. Redes neuronales artificiales

Esta sección presenta un repaso histórico sobre las redes neuronales, los conceptos más importantes para entender su funcionamiento y la forma en que aprenden.

2.5.1. Repaso histórico

Las redes neuronales artificiales o artificial neural networks (ANN) son un sistema utilizado para tareas de aprendizaje automático. Su nombre se debe a que modelan vagamente las neuronas del cerebro, como se muestra en la figura 2.1. En 1943 Warren S. McCulloch y Walter Pitts realizaron una publicación [51] que buscaba comprender cómo el cerebro humano podría producir patrones complejos a través de células cerebrales conectadas o neuronas. Una de las principales ideas que surgieron de este trabajo fue la comparación de neuronas con un umbral binario con la lógica booleana (es decir, 0/1 o declaraciones verdadera/falsas). Reforzando el concepto de neuronas y cómo trabajan, D. Hebb publicó The Organization of Behavior [27] en 1949 donde señaló que las vías neuronales se fortalecen cada vez que se utilizan, lo que se

denominó el aprendizaje Hebbiano. A Frank Rosenblatt se le atribuye el desarrollo del perceptrón en 1958 [64]. Rosenblatt llevó el trabajo de McCulloch y Pitt un paso más allá al introducir pesos en la ecuación, logrando hacer que una computadora aprendiera a distinguir las tarjetas marcadas a la izquierda de las tarjetas marcadas a la derecha. Desafortunadamente, el perceptrón es limitado y así se demostró en el libro Perceptrons de 1969 de Marvin Minsky y Seymour Papert [52]. La demostración de dicha limitación, sumado a que los recursos de esa época no eran suficientes para soportar redes neuronales muy grandes, generó la pérdida de interés y disminución de investigación en este campo. Este período se denominó invierno de la inteligencia artificial. Sin embargo, el avance que finalmente puso fin al invierno de la inteligencia artificial en 1986 fue el algoritmo de backpropagation [65], que podría usarse para entrenar redes multicapa, superando las limitaciones de la red de una sola capa (perceptron). El desarrollo de backpropagation llevó a un renovado interés en la posible aplicación de redes neuronales.

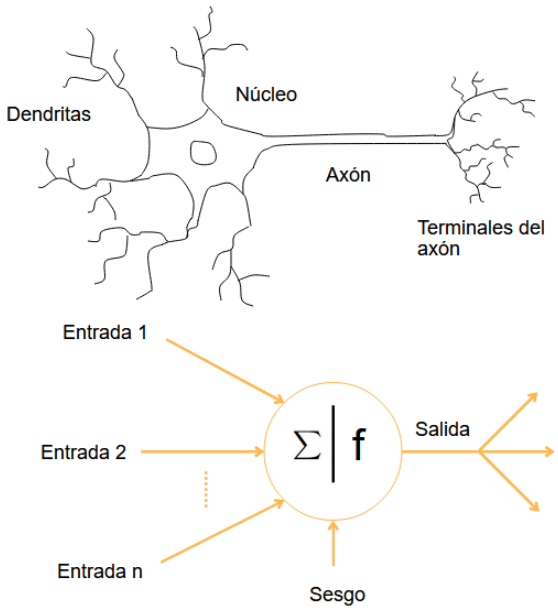


Figura 2.1: Comparación de la estructura de una neurona real con una artificial

2.5.2. Conceptos generales

Una neurona es la unidad básica de una red neuronal y su función es la de recibir información de entrada, ya sea del exterior de la red o de otras neuronas, procesarla de alguna manera, y producir una salida. Cada neurona tiene

asociado un conjunto de pesos que son utilizados durante el procesamiento de la información de entrada. Las neuronas usualmente son agrupadas en capas, siendo la capa de entrada la que recibe la información de una fuente externa, la capa de salida la que retorna el resultado de procesamiento de esa información por parte de la red, y las capas ocultas las que se encuentran entre esas dos.

La arquitectura más básica que se puede modelar es la del perceptrón, la cual consta de una capa de entrada y un único valor de salida. Dado un vector de entrada $x^T = (x_1, x_2, ..., x_n)$ el perceptrón se compone por una primera capa de neuronas con los atributos de entrada (capa de entrada), que denotaremos como $a^{(1)}$, y una segunda capa compuesta por una sola neurona, que calcula la combinación lineal de las entradas y le aplica la llamada función de activación para obtener una salida real. La arquitectura del perceptrón se muestra en la figura 2.2.

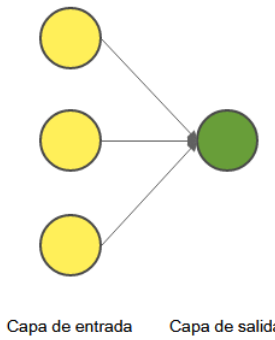


Figura 2.2: perceptrón

En la Ec. 2.1 se representa vectorialmente la capa de entrada del perceptrón, donde el supraíndice 1 denota que es la primer capa de la red y $x_0 = 1$ es el llamado sesgo (o bias), usualmente es inicializado en 1 y su objetivo es agregar un grado más de libertad a la función.

$$x \in \mathbb{R}^n = a^{(1)} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix} \tag{2.1}$$

Se define $\theta^{(1)} = [\theta_0^{(1)} ... \theta_n^{(1)}]$ como el conjunto de parámetros (o pesos) que son considerados tal cual se expresa en la Ec. 2.2 para el cálculo de la combinación lineal por parte de la capa de salida, siendo $g(x)$ la función de activación, cuya

tarea es recibir la señal de salida de la capa anterior y convertirla en alguna forma que pueda tomarse como entrada para la siguiente capa. El principal objetivo de incorporar funciones de activación a la red neuronal es que logre aprender patrones complejos en los datos. Algunos ejemplos de funciones de activación se presentan en la figura 2.3.

$$h_\theta(x) = a^{(2)} \in \mathbb{R} = g(\theta^{(1)} \cdot x) \tag{2.2}$$

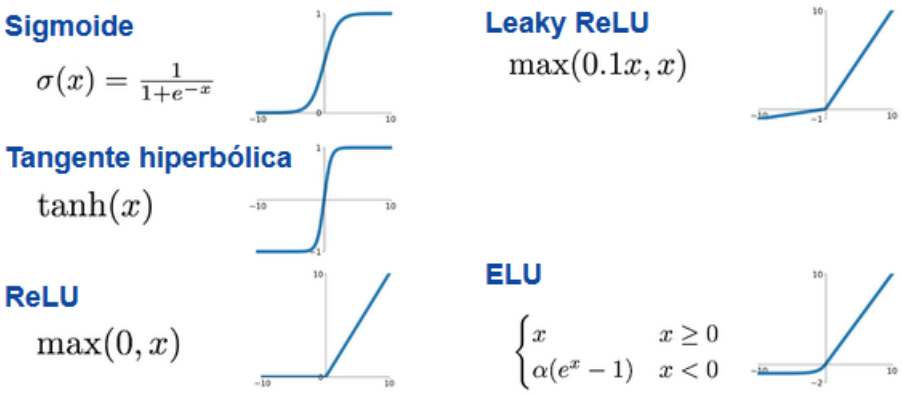


Figura 2.3: Funciones de activación usualmente utilizadas

Las redes neuronales son una generalización del perceptrón: en cada una de las capas puede haber más de una neurona, que recibe como entrada los resultados de la capa anterior, y pueden introducirse capas intermedias (ocultas). Generalizando, se tiene que la representación de la capa j es la Ec. 2.3, donde S_j es la cantidad de neuronas de la capa j y $\theta^{(j)}$ la matriz de pesos utilizados para pasar de la capa j a la capa $j+1$. La matriz $\theta^{(j)}$, como se muestra en la Ec. 2.4, contiene en sus filas los pesos asociados a la combinación lineal de las entradas de la unidad i de la capa $j+1$, que son los resultados de la activación de las unidades en la capa j.

$$a^{(j)} \in \mathbb{R}^{S_j} = \begin{bmatrix} a_0^{(j)} \\ a_1^{(j)} \\ \vdots \\ a_{S_j}^{(j)} \end{bmatrix} = g(\theta^{(j-1)} \cdot a^{(j-1)}) \tag{2.3}$$

$$\theta^{(j)} = \begin{pmatrix} \theta_{10}^{(j)} & \theta_{11}^{(j)} & \dots & \theta_{1S_j}^{(j)} \\ \theta_{20}^{(j)} & \theta_{21}^{(j)} & \dots & \theta_{2S_j}^{(j)} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{S_{j+1}0}^{(j)} & \theta_{S_{j+1}1}^{(j)} & \dots & \theta_{S_{j+1}S_j}^{(j)} \end{pmatrix} \tag{2.4}$$

Es posible observar que $\theta^{(j)} \in \mathbb{R}^{S_{j+1}} \times \mathbb{R}^{S_j+1}$. Es decir, $\theta^{(j)}$ tiene tantas filas como neuronas hay en la capa $j+1$, y tantas columnas como neuronas hay en la capa j, adicionando el sesgo de cada capa. Cada valor $\theta_{ik}^{(j)}$ es el peso asociado a la i-ésima neurona de la capa $j+1$, correspondiente a la entrada proveniente de la k-ésima neurona de la capa j. Una representación de un ejemplo de red neuronal con dos capas ocultas puede verse en la figura 2.4. A aquellas capas en donde en todas sus neuronas hay un peso conectado a cada salida de la capa anterior se las denominan capas completamente conexas.

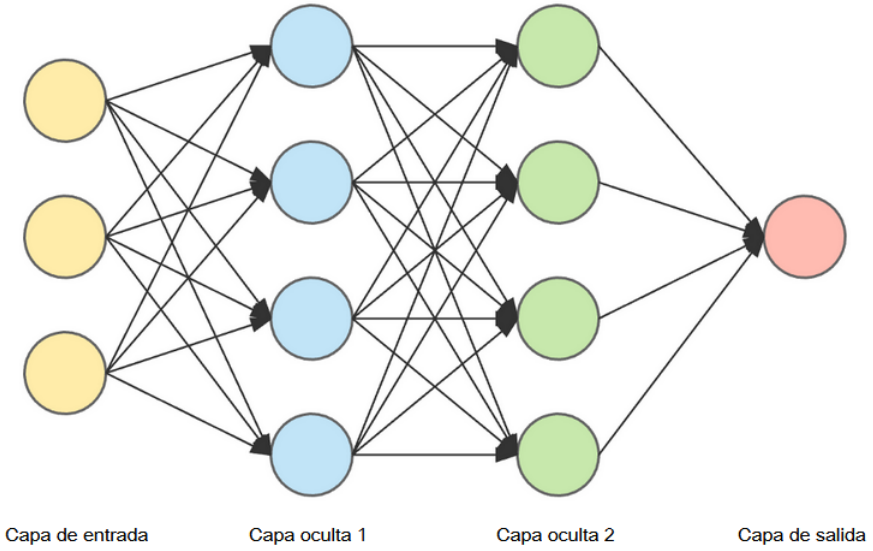


Figura 2.4: Red Neuronal multicapa

2.5.3. Aprendizaje en Redes Neuronales

En esta sección se introducen los algoritmos descenso por gradiente y back-propagation utilizados para el aprendizaje en redes neuronales, además del rol de la tasa de aprendizaje.

2.5.3.1. Algoritmos de aprendizaje

En el contexto de redes neuronales, lo que se intenta aprender son las matrices de pesos $\theta^{(j)}$ de cada capa de la red. Para que el aprendizaje ocurra, se utilizan los datos de entrenamiento, una función de pérdida f y una forma de buscar minimizarla.

El primer método para minimizar la función de pérdida a mencionar es el descenso por gradiente, el cual es un algoritmo de optimización que retorna un mínimo local de una función diferenciable f. Para hacerlo, se parte de un punto aleatorio x y se calcula el gradiente respecto a las variables de la función, luego se actualiza x conforme una tasa de aprendizaje α, según la Ec. 2.5.

$$x \to x' = x - \alpha \nabla \frac{\partial f}{\partial x} \tag{2.5}$$

En el ámbito de redes neuronales, f es la función de pérdida y las variables de la función cada uno de los pesos θ^j de la red tal cual se muestra en la Ec. 2.6. De esta forma se puede minimizar la función de pérdida ajustando los valores de los pesos, lo que típicamente es el aprendizaje de la red. Si bien el descenso por gradiente plantea que este sea calculado luego de haber procesado todos los datos de entrada, existe su variable estocástica (descenso por gradiente estocástico) que lo calcula y ajusta los pesos cada cierta cantidad de datos procesados, subconjunto del total, que se denominan lotes.

$$\theta^{(j)} \to \theta^{(j)\prime} = \theta^{(j)} - \alpha \nabla \frac{\partial f}{\partial \theta^{(j)}} \tag{2.6}$$

Otro algoritmo importante de minimización es backpropagation, que permite calcular el gradiente de la función de costo de forma eficiente haciendo uso de la regla de la cadena. Sea el conjunto de entrenamiento $(x^{(1)}, y^{(1)}), ..., (x^{(m)}, y^{(m)})$, para cada ejemplo $x = x^{(i)}$:

1. Se define la capa 1 de entrada: $a^{(1)} := x$
2. Para cada capa $l = 2, 3, ..., L$ se calcula $z^{(l)} = \theta^{(l-1)} a^{(l-1)}$ y $a^{(l)} = g(z^{(l)})$
3. Se calcula $\delta^{(L)} = (a^{(L)} - y) \odot g'(z^{(L)})$ donde $\odot$ es el producto de Hadamard.
4. Se propaga el error hacia atrás de la siguiente forma: para cada $l = L-1, L-2, ..., 2$ se calcula $\delta^{(l)} = (\theta^{(l)})^T \delta^{(l+1)} \odot g'(z^{(l)})$

5. El gradiente de la función de costo f se calcula según la Ec. 2.7.

$$\frac{\partial f}{\partial \theta_{jk}^{(l)}} = a_k^{(l)} \delta_j^{(l+1)} \tag{2.7}$$

6. Finalmente, luego de calculadas las derivadas para cada uno de los parámetros y para cada uno de los ejemplos de entrenamiento (o lotes en el caso estocástico), se utiliza descenso por gradiente para actualizar los pesos de las capas $l = L, L-1, L-2, ..., 2$ tal cual se expresa en la Ec 2.8, donde $\delta^{x,l}$ es el valor de δ^l para la instancia x.

$$\theta^{(l)} = \theta^{(l)} - \alpha \sum_x \delta^{x,(l+1)} \cdot (a^{(x,l)})^T \tag{2.8}$$

La combinación de descenso por gradiente y backpropagation no garantiza hallar el mínimo global de la función, pero sí uno local. Este resultado es en general suficiente en la práctica.

2.5.3.2. Tasa de aprendizaje

En la sección anterior se presentó el descenso por gradiente estocástico, un algoritmo de optimización que estima el gradiente de error para el estado actual del modelo usando ejemplos del conjunto de datos de entrenamiento, y que luego actualiza los pesos del modelo usando el algoritmo de backpropagation. Mientras que la dirección del descenso se determina a partir del gradiente de la función de pérdida, la tasa de aprendizaje determina qué tan grande se da un paso en esa dirección. La tasa de aprendizaje es un hiperparámetro configurable utilizado en el entrenamiento de redes neuronales que tiene un pequeño valor positivo, a menudo en el rango entre 0 y 1.

La tasa de aprendizaje controla la rapidez con la que el modelo se adapta a la tarea. Las tasas de aprendizaje de menor valor requieren más épocas de entrenamiento dados los cambios más pequeños realizados en los pesos en cada actualización, mientras que las tasas de aprendizaje de valores más cercanos a 1 dan como resultado cambios más bruscos y requieren menos épocas de entrenamiento. Sin embargo, una tasa demasiado grande puede generar que el aprendizaje se saltee los mínimos de la función u oscile alrededor de uno (figura 2.5b), mientras que una tasa demasiado chica puede hacer que el proceso tarde demasiado en converger o lo haga a un mínimo local no deseado (figura 2.5a).

De modo que se busca una tasa de aprendizaje que sea suficientemente baja como para que la red converja en algo útil, pero lo suficientemente alta como para que pueda entrenarse en un período de tiempo razonable. Si bien el uso de una tasa de aprendizaje fija puede proporcionar resultados decentes, a menudo se suele optar por utilizar técnicas que manejan una tasa variable dependiendo de la etapa de entrenamiento. El objetivo es establecer un valor óptimo según ciertas condiciones.

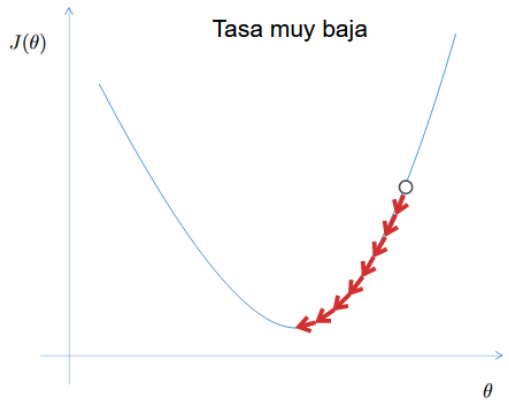
Un ejemplo de la utilización de tasas optimizadas son las adaptables. En este enfoque, la tasa de aprendizaje aumenta o disminuye según el valor del gradiente de la función de costo. Para un valor de gradiente más alto, la tasa será menor y para un valor de gradiente más bajo, será mayor. Por lo tanto, el aprendizaje se desacelera y acelera, respectivamente, en las partes más empinadas y menos profundas de la curva de la función de costo (figura 2.5c). La fórmula utilizada en las tasas optimizadas se muestra en la Ec 2.9. La variable m_t es denominada "impulso" y se calcula según la Ec. 2.10, donde m_t y m_{t-1} son los agregados de gradientes en los tiempos t y $t-1$ respectivamente (inicialmente, $m_t = 0$), W_t y W_{t+1} los pesos en los tiempos t y $t+1$, α_t y ∂W_t la tasa de aprendizaje y derivada de los pesos en el tiempo en t correspondientemente, ∂L la derivada de la función de costo y β el parámetro de movimiento promedio (constante).

$$w_{t+1} = w_t - \alpha m_t \tag{2.9}$$

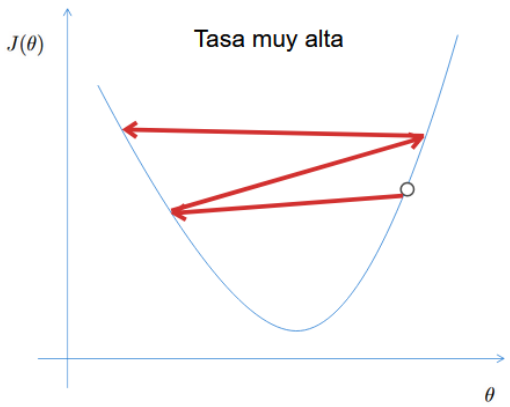
$$m_t = \beta m_{t-1} + (1-\beta)\left[\frac{\partial L}{\partial w_t}\right] \tag{2.10}$$

2.5.3.3. Optimizador Adam

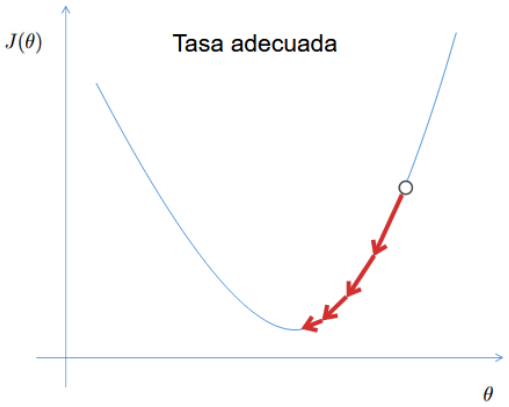
El algoritmo de optimización de estimación de momento adaptativo (Adam) es una extensión del descenso de gradiente estocástico que recientemente ha visto una adopción más amplia para aplicaciones de aprendizaje profundo en visión artificial y procesamiento de lenguaje natural [38]. Adam permite actualizar iterativamente los pesos de la red en función de los datos de entrenamiento y su principal diferencia con el descenso por gradiente estocástico es que este último mantiene una única tasa de aprendizaje (denominada α) para todas las actualizaciones de peso que no cambia durante el



(a) Minimización para una tasa muy baja.



(b) Minimización para una tasa muy alta.



(c) Minimización para una tasa adecuada (y variable).

Figura 2.5: Representaciones del proceso de minimización de la función θ para una tasa muy baja (a), muy alta (b) y una adecuada (c).

entrenamiento, mientras que Adam sostiene una tasa de aprendizaje para cada peso de red (parámetro) y se adapta por separado a medida que se desarrolla el aprendizaje.

Los autores describen a Adam como una combinación de las ventajas de otras dos extensiones del descenso de gradiente estocástico: algoritmo de gradiente adaptativo (AdaGrad) y propagación de raíz cuadrática media (RMSProp). En lugar de adaptar las tasas de aprendizaje en función solo de los impulsos promedio de primer orden (Ec. 2.11) como en RMSProp, Adam también utiliza los impulsos promedio de los gradientes de segundo orden (Ec. 2.12).

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.11}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.12}$$

Debido a que el impulso de primer y segundo orden son muy pequeños durante el entrenamiento inicial porque los valores β_1 y β_2 son cercanos a uno (0.9 y 0.999 respectivamente), se agrega un paso más conocido como corrección del sesgo. Las fórmulas finales que consideran la corrección del sesgo son las expuestas en las ecuaciones 2.13 y 2.14, donde t representa la t-ésima potencia. Lo que significa que al comienzo del entrenamiento la tasa de aprendizaje se corrige dividiendo entre $(1 - \beta)$ y, a medida que suceden las iteraciones de entrenamiento, el denominador se acerca cada vez más a 1.

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.13}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.14}$$

2.5.4. Funciones de pérdida

En esta sección se presentan funciones de pérdida utilizadas en un conjunto de trabajos que implementan segmentación de imágenes.

2.5.4.1. Introducción

Las funciones de pérdida juegan un papel importante en cualquier modelo de aprendizaje automático: los parámetros aprendidos por el modelo se determinan minimizando la función de pérdida elegida y pueden eventualmente definir el objetivo con el que se evalúa el rendimiento del modelo. Se utilizan para medir qué tan lejos está un valor estimado de su valor real y se asignan decisiones en relación a sus costos asociados. Como se cumple generalmente en aprendizaje automático, no hay una función de pérdida que sea mejor que otra en todos los casos, y se deberá utilizar la que mejor funcione o se adapte a la tarea a desarrollar.

2.5.4.2. Entropía Cruzada

La entropía cruzada es una medida de la diferencia entre dos distribuciones de probabilidad para una determinada variable aleatoria o un conjunto de eventos. En el campo de teoría de la información, la entropía es el número de bits

necesarios para transmitir un evento seleccionado al azar de una distribución de probabilidad. Una distribución sesgada tiene una entropía baja, mientras que una distribución en la que los eventos tienen la misma probabilidad tiene una entropía mayor.

La idea de entropía cruzada puede ser útil para optimizar modelos de clasificación si consideramos que cada muestra tiene una etiqueta de clase conocida con una probabilidad de 1 y una probabilidad de 0 para todas las demás etiquetas. Un modelo puede estimar la probabilidad de que una muestra pertenezca a cada etiqueta de clase y la entropía cruzada se puede utilizar para calcular la diferencia entre las dos distribuciones de probabilidad. La ecuación 2.15 muestra el cálculo de la entropía cruzada, donde $\hat{y}_i$ es la predicción de la muestra i, y y_i representa su valor real. C es la cantidad de clases total. Si $C = 2$ entonces se calcula la versión binaria de entropía cruzada (BCE) según la ecuación 2.16. Para el caso particular de BCE existen dos clases: $C1$ y $C2$, $y_2 = (1 - y_1)$ y $\hat{y}_2 = 1 - \hat{y}_1$.

$$CE = -\sum_{i=1}^{C} y_i \cdot \log \hat{y}_i \tag{2.15}$$

$$BCE = -\sum_{i=1}^{C=2} y_i \cdot \log \hat{y}_i = -y_1 \cdot \log \hat{y}_1 - (1 - y_1) \cdot \log(1 - \hat{y}_1) \tag{2.16}$$

2.5.4.3. Dice Loss

Dice loss tiene su origen en el coeficiente Sørensen-Dice (DSC), utilizado estadísticamente para medir la similitud entre dos muestras cuantificando su superposición. DSC varía de 0 a 1, donde un valor de 1 denota una superposición perfecta y completa. DSC se desarrolló originalmente para datos binarios y se puede calcular según la Ec. 2.17, donde $|X \cap Y|$ representa los elementos comunes entre los conjuntos X y Y, y $|X|$ representa el número de elementos en el conjunto A (análogo para B).

$$DSC = 2 \times \frac{|X \cap Y|}{|X| \cup |Y|} \tag{2.17}$$

Como función de pérdida en el entrenamiento de redes neuronales, el numerador representa las activaciones comunes entre la predicción y la máscara objetivo, mientras que el denominador refiere a la cantidad de activaciones en

cada máscara por separado. Esto tiene el efecto de normalizar la pérdida de acuerdo con el tamaño de la máscara de destino, de esta manera DSC mitiga las dificultades para aprender de clases con menor representación espacial en una imagen. Como valores más grandes de DSC connotan un mejor desempeño, la función que se minimiza es $1 - DSC$. En la sección 2.6 se menciona DSC como una métrica de desempeño y se realiza un análisis comparativo acerca de su uso.

2.5.4.4. Tversky

El índice de Tversky (TI) es una medida de similitud asimétrica que es una generalización de DSC utilizada ya en otros estudios sobre segmentación de imágenes con buenos resultados [66]. TI se define según la Ec. 2.18, donde α y β funcionan como un mecanismo de ponderación sobre los falsos negativos (FN) y falsos positivos (FP) respectivamente. TP son las predicciones verdaderas positivas. Si se cumple que $\alpha > \beta$ entonces se está penalizando los falsos negativos. Observar que DSC es el caso particular de TI cuando $\alpha = \beta = 0.5$ y, al igual que el primero, la función de pérdida a considerar es $1 - TI$.

$$TI = \frac{TP}{TP + \alpha FN + \beta FP} \tag{2.18}$$

Abraham y Khan [1] propusieron una variante focalizada de Tversky: Focal Tversky Loss (FTL). Los autores abordaron la tarea de segmentación sobre imágenes médicas y notaron que uno de los problemas que tienen otras funciones como DSC es que tienen dificultades para segmentar lo que denominan regiones de interés o regions of interest (ROI). Las ROI son aquellas regiones usualmente pequeñas y con poca representatividad respecto a la clase negativa donde se ubican los casos positivos y que en general no contribuyen a la pérdida de manera significativa.

Para abordar la dificultad de segmentar las ROI los autores agregaron un nuevo parámetro γ cuyo objetivo es regular, según su valor, el aprendizaje sobre los casos negativos "fáciles" y las áreas de interés, típicamente "difíciles". Si bien los autores presentan FTL con la Ec. 2.19, para mayor facilidad en la explicación de la función de γ, en este trabajo se consideró el cálculo de FTL de la clase C según la Ec. 2.20. En la Ec. 2.20 se puede observar que si $\gamma < 1$, el gradiente de FCL es mayor para los ejemplos en los que $TI > 0,5$, lo que obliga al modelo a centrarse en tales ejemplos ("fáciles"). Por el contrario,

si $\gamma > 1$ se tiene como resultado un gradiente más alto para los ejemplos en los que $TI < 0,5$. Esto obliga al modelo a centrarse en ejemplos más difíciles, especialmente en segmentaciones a pequeña escala que suelen recibir puntuaciones de TI bajas.

$$FTL_C = \sum_c (1 - TI_c)^{\frac{1}{\gamma}} \tag{2.19}$$

$$FTL_C = \sum_c (1 - TI_c)^{\gamma} \tag{2.20}$$

2.5.5. Aprendizaje profundo

Goodfellow [20] llama aprendizaje profundo a los enfoques de aprendizaje que hacen uso de redes neuronales de múltiples capas. Como el total de las capas es la cantidad de niveles de composición de operaciones no lineales en la función aprendida, este tipo de redes tienden a aprender funciones más complejas. Dicho de otra forma, los métodos de aprendizaje profundo apuntan a aprender las jerarquías de los distintos atributos o características partiendo de sus niveles más altos. El aprendizaje de los atributos en múltiples niveles de abstracción, representados por cada capa de la arquitectura, permite que un sistema aprenda funciones complejas mapeando la entrada a la salida directamente a partir de los datos. La necesidad de contar con redes de mayor profundidad se da porque algunas funciones no pueden ser representadas eficientemente por arquitecturas "llanas", lo que tiene como consecuencia que no sean las indicadas para aprender la tarea de interés.

Bengio [6] estableció que una arquitectura con profundidad insuficiente puede requerir muchos más elementos computacionales (potencialmente exponencialmente mayor con respecto al tamaño de entrada), que arquitecturas cuya profundidad es acorde a la tarea. Además afirmó que una profundidad insuficiente puede ser perjudicial para el aprendizaje y que, si una solución a la tarea se representa con una arquitectura muy grande pero poco profunda, es posible que se necesiten muchos ejemplos de entrenamiento para ajustar cada uno de estos elementos.

2.6. Métricas de desempeño

La presente sección desarrolla los conceptos detrás de un conjunto de métricas utilizadas en aprendizaje automático.

2.6.1. Introducción

En la literatura existen diversas métricas para evaluar el desempeño de un modelo de aprendizaje automático. En las secciones siguientes se presentan las métricas más utilizadas, comenzando por las más simples o aplicables de forma más general a un problema de aprendizaje, y luego se sigue con las más habituales en el contexto de procesamiento y clasificación de imágenes. Para esto se definen los siguientes términos, donde las clases refieren a Edificio y No Edificio:

- Verdadero positivo/true positive (TP): clasificación correcta de la clase positiva (Edificio).
- Falso positivo/false positive (FP): clasificación incorrecta de la clase positiva (Edificio).
- Verdadero negativo/true negative (TN): clasificación correcta de la clase negativa (No Edificio).
- Falso negativo/false negative (FN): clasificación incorrecta de la clase negativa (No Edificio).

2.6.2. Métricas en aprendizaje automático

Exactitud (accuracy en inglés) es probablemente la métrica más simple, calculada como el número de predicciones correctas dividido por el total de predicciones (Ec. 2.21).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.21}$$

Como es conocido, accuracy no es un buen indicador de desempeño en muchos escenarios. Un ejemplo de esto es cuando existe un claro desbalanceo en la distribución de las clases. Por ejemplo, si se tuviera un conjunto de datos cuyas instancias tuvieran una presencia proporcional mucho mayor de una clase C_1 sobre otra C_2. Si el modelo predijera siempre a C_1, podría tener

un buen valor de accuracy pero no estaría aprendiendo a identificar a los de la clase C_2. En segmentación de imágenes accuracy es denominada pixel accuracy y, como su nombre lo indica, las instancias de clasificación son cada píxel de la imagen. Mostrando un ejemplo más concreto dados los datos de ejemplo de la tabla 2.1, obtendríamos un pixel accuracy por encima de $0.90 = (10 + 940)/(10 + 940 + 90 + 60)$ cuando claramente la clase Edificio no es correctamente clasificada en la gran mayoría de los casos.

		Predicción	
		E	NoE
Real	E	10	10
	NoE	90	940

Tabla 2.1: Ejemplo hipotético de clasificación de 1000 píxeles, donde E se corresponde con la clase Edificio, y NoE con No Edificio.

Precisión (precision), por otro lado, es una métrica que cubre la necesidad de considerar el desempeño para cada clase. Si C es la clase de interés, su precision se calcula según la Ec. 2.22. Para el caso de ejemplo visto anteriormente, el valor de precision de cada clase sería calculado como $precision_E = (10)/(10 + 90) = 0.10$ para la clase Edificio y $precision_{NoE} = (940)/(940 + 10) = 0.99$ para No Edificio.

$$precision_C = \frac{TP}{TP + FP} \tag{2.22}$$

El mecanismo por el cual se calcula precision da una noción más certera acerca del mal desempeño para la clase Edificio. La idea detrás de precision es la de medir la proporción de las instancias que nuestro modelo dice que existían en la clase, y que eran realmente de esa clase.

Otra métrica frecuentemente utilizada es exhaustividad (recall), definida según la Ec. 2.23, y expresa la capacidad de recuperar todas las instancias relevantes de una clase en un conjunto de datos. El cálculo para el caso de ejemplo es $recall_E = (10)(10 + 10) = 0.50$ para la clase Edificio y $recall_{NoE} = (940)(940 + 90) = 0.91$ para No Edificio.

$$recall_C = \frac{TP}{TP + FN} \tag{2.23}$$

Mientras que la precision toma más relevancia cuando nos interesa minimizar los falsos positivos, el recall lo hace cuando se intentan minimizar los falsos negativos.

Existe otra métrica, F_1-score, que combina precision y recall calculando el promedio armónico de ambos según la Ec. 2.24. F_1-score es útil si se busca un equilibrio entre precision y recall y hay una distribución de clases desigual.

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \tag{2.24}$$

Una versión más general de F_1-score, denominada F_β (Ec.2.25), agrega la variable β que puede manipularse para ponderar la precision o el recall: β se elige de manera que recall se considere β veces tan importante como precision. F1-score es el caso particular de F_β donde $\beta = 1$.

$$F_\beta = (1 + \beta^2) \times \frac{precision \times recall}{(\beta^2 \times precision) + recall} \tag{2.25}$$

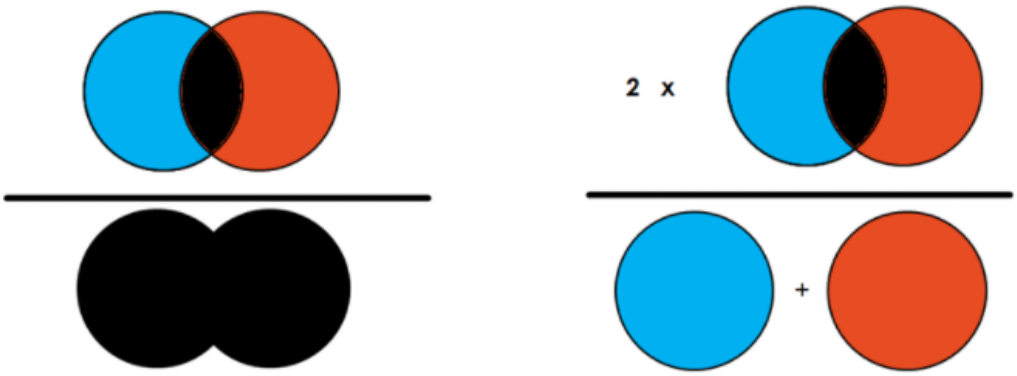
En segmentación de imágenes el valor de F_1-score es también denominado como el coeficiente Sørensen-Dice (DSC), utilizado para comparar la similitud de dos muestras. Si bien el cálculo es el mismo, la formulación conceptual se adecúa más a los resultados de una predicción donde el concepto de localización está incluido, como es en el caso de segmentación de imágenes. Sea X el mapa de clases real de una imagen e Y el mapa predicho por el modelo, entonces el DSC se define por la ya presentada Ec. 2.17.

Por último, se define el índice de Jaccard o intersection over union (IoU) como la división de la intersección entre X e Y, dividido por su unión (Ec. 2.26).

$$IoU = \frac{|X \cap Y|}{|X \cup Y|} \tag{2.26}$$

2.6.3. IoU vs. DSC

En la sección anterior fueron presentadas las métricas evaluadas para el desempeño de los modelos implementados. Las dos últimas, IoU y DSC, tienen una aplicación directa en la medición de similaridad de dos muestras y se adecúan muy bien al contexto en donde se comparan un conjunto de instancias (figura 2.6), como es el caso de las imágenes (formadas por un conjunto de píxeles). En esta sección se hace una comparativa entre ambas con el fin de concluir cuál de ellas es la mejor para la evaluación de los modelos implementados.



(a) IoU **(b)** DSC

Figura 2.6: En las figuras se representan a los conjuntos de píxeles de X e Y como los círculos anaranjado y azul respectivamente. (a) y (b) muestran el cálculo de cada métrica en dicha representación.

De las definiciones de ambas métricas se puede ver que se cumple la desigualdad expresada en Ec. 2.27. Además, en los casos de borde donde hay una coincidencia total o nula entre ambas muestras, ambas métricas toman los mismos valores (1 y 0 respectivamente).

$$\frac{DSC}{2} \leqslant IoU \leqslant DSC \tag{2.27}$$

De una forma más explícita, se formula la igualdad de Ec. 2.28, donde se aprecia cuál es la relación entre ellas. En la Ec. 2.28 queda evidenciado que la relación tiende a $1/2$ cuando IoU y DSC tienden a 0. También se puede aseverar que tanto para una métrica como para la otra se cumple que si un clasificador A es mejor que B en una de ellas, también es mejor que el clasificador B en la otra. Sin embargo, hay una distinción entre ambos que se manifiesta cuando se toma la puntuación media de un conjunto de predicciones. La diferencia surge al cuantificar cuánto peor es el clasificador B que A para cualquier caso dado. En general, IoU tiende a penalizar las instancias individuales de mala clasificación más que la puntuación DSC cuantitativamente, incluso cuando ambos pueden estar de acuerdo en que esta instancia es mala. Esto es porque IoU tiende a tener un efecto de "elevar al cuadrado" en los errores comparativamente con el DSC. Por lo tanto, DSC tiende a medir algo más cercano al desempeño promedio, mientras que el IoU mide algo más cercano al desempeño en el peor de los casos. Por ejemplo, si se tuviera que la gran mayoría de las predicciones son moderadamente mejores con el clasificador A que con B, pero algunas de

ellas son significativamente peores con el clasificador A, puede darse el caso de que DSC favorezca al clasificador A mientras que IoU favorezca al clasificador B.

$$\frac{IoU}{DSC} = \frac{1}{2} + \frac{IoU}{2} \tag{2.28}$$

Dado que se considera más importante ponderar las peores clasificaciones de cada modelo por sobre el promedio, se decide considerar IoU como la medida de desempeño a considerar para evaluar los modelos de este proyecto. Además, IoU es una métrica ampliamente utilizada en la literatura y trabajo similares consultados [2, 24, 45, 61, 63, 79].

2.7. Data augmentation

La actual sección introduce los principales conceptos sobre aumento de datos (data augmentation).

2.7.1. Introducción

La precisión de la predicción de los modelos de aprendizaje profundo depende en gran medida de la cantidad y diversidad de datos disponibles durante el entrenamiento. En términos simples, la cantidad de datos necesarios es proporcional al número de parámetros que se pueden aprender en el modelo. Como en el contexto de aprendizaje profundo los parámetros a entrenar están en el orden de millones, los datos necesarios para hacerlo deben ser también numerosos. Sin embargo, en muchos escenarios no se cuenta con la cantidad deseada de datos para llevar adelante un correcto entrenamiento, por lo que se aplican técnicas de aumento de datos (data augmentation) para generarlos.

Data augmentation es una técnica que se utiliza para aumentar artificialmente el tamaño del conjunto de datos. El proceso consta de tomar una muestra del conjunto, modificarla de alguna manera y luego agregarla al conjunto de datos original. Data augmentation ayuda no solo a superar el problema de la escasez de datos, sino que además evita el sobreajuste y hace que en general el modelo funcione mejor ante ejemplos no vistos. También es de utilidad cuando en una tarea de clasificación hay un desbalance significativo entre ejemplos de cada clase. Los beneficios se obtienen sin el esfuerzo que implica recoger nuevos datos y etiquetarlos, donde en algunos contextos puede ser costoso o inviable.

2.7.2. Data augmentation aplicado a imágenes

En el escenario donde los datos son imágenes, es útil enmarcar el contexto del problema y considerar qué hace que el reconocimiento de imágenes sea una tarea tan difícil en primer lugar. En los ejemplos clásicos de discriminación entre gatos y perros, el software de reconocimiento de imágenes debe superar problemas de punto de vista, iluminación, fondo, escala, entre otros. La tarea del aumento de datos es integrar las invariancias en el conjunto de datos de modo que los modelos resultantes funcionen bien a pesar de estos desafíos [70].

Una de las primeras aplicaciones de aumento de imágenes por medio de su edición se puede encontrar en Le-Net [41], la cual utiliza CNNs. Otra arquitectura muy conocida, AlexNet [39] utilizó data augmentation para aumentar el tamaño del conjunto de datos en una magnitud de 2048. El enfoque abordado ayudó a reducir el sobreajuste al entrenar, los autores afirman que sus aumentos redujeron la tasa de error del modelo en más del 1 %. Otro caso de éxito es en el ámbito de las imágenes médicas, un ejemplo de esto son los resultados obtenidos en la detección de cáncer de piel presentada por Esteva et al. [17] en 2017.

A continuación se exponen algunas transformaciones que frecuentemente se realizan sobre imágenes con el fin de aumentar los datos, aunque es normal que se apliquen combinaciones de dos o más transformaciones sobre una imagen de entrenamiento. Ejemplos individuales de cada una pueden apreciarse en la figura 2.7:

- Inversión: la imagen se invierte sobre el eje horizontal o vertical.
- Rotación: se realizan girando la imagen en sentido horario o antihorario en un eje entre 1° y 359°.
- Recorte: se realiza una segmentación de la imagen menor a su dimensión original. El recorte puede estar centrado en el mismo centro de la imagen o en algún otro punto aleatorio.
- Traslado: se hace un corrimiento de la imagen en algún sentido. A medida que la imagen original se traslada en una dirección, el espacio restante se puede rellenar con un valor constante como 0 o 255.
- Adición de ruido: se logra ennegreciendo o emblanqueciendo píxeles aleatorios, agregando ruido Gaussiano o incluso eliminando una región.
- Transformaciones en colores: las transformaciones de color más simples consisten en aislar un solo canal de color como rojo (R), verde (G) o azul

(B). Una imagen se puede convertir rápidamente en su representación en un canal de color aislando esa matriz y agregando dos matrices cero de los otros canales de color. Además, los valores RGB se pueden manipular fácilmente con operaciones matriciales simples para aumentar o disminuir el brillo o contraste de la imagen.

Figura 2.7: Algunas transformaciones habituales aplicadas para el aumento de datos.

2.8. Calibración de modelos

Esta sección presenta el concepto de calibración y un conjunto de métricas y métodos para medirla y mejorarla.

2.8.1. Introducción

Cuando se trabaja en un problema de clasificación puede ser conveniente que el modelo prediga las probabilidades de que los datos pertenezcan a cada clase posible además de la mera determinación de una clase u otra. Obtener acceso a las probabilidades es útil para una interpretación más extensiva de las respuestas, analizando las deficiencias del modelo o presentando la incertidumbre a los usuarios finales. La aplicación de funciones de activación como softmax que transforman la salida de los clasificadores a valores entre 0 y 1 pueden llevar a pensar que la confianza con la que se predice una clase es ese valor numérico, cuando esto no se cumple a menos que el modelo esté calibrado.

Formalmente, un modelo está perfectamente calibrado si para cualquier valor de probabilidad p, una predicción de una clase con confianza p es correcta el $100 \times p$ por ciento de las veces. La calibración del modelo se refiere al proceso en el que tomamos un modelo que ya está entrenado y le aplicamos ciertas operaciones, lo que mejora su estimación de probabilidad.

2.8.2. Métricas

En esta sección se introducen distintos métodos estadísticos diseñados para medir empíricamente la calibración de un modelo.

2.8.2.1. Diagramas de confianza

La forma más directa de evaluar la calibración de un modelo es generar un gráfico de calibración. Los gráficos muestran las posibles diferencias entre las probabilidades predichas por el modelo y las observadas en los datos. Si el modelo estuviera calibrado perfectamente, las probabilidades estimadas siempre serían las reales, por lo que se debería ver una línea recta correspondiente a la función identidad. Por este motivo, cuanto más calibrado esté el modelo, más cerca estará la curva del gráfico de la función identidad. Diagramas de confianza de ejemplos para modelos no calibrado y calibrado se muestran en la figura 2.8.

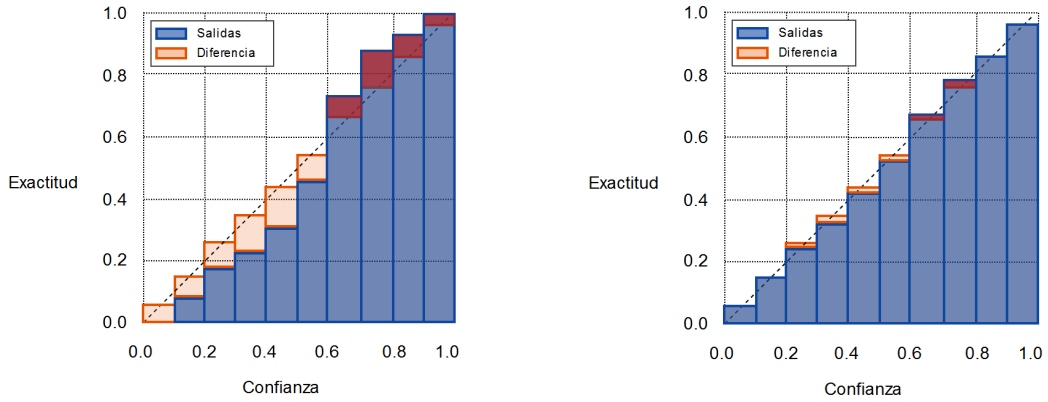


Figura 2.8: Ejemplo comparativo de un posible diagrama de confianza de modelo no calibrado (a) frente a uno que presenta una mejor calibración (b)

2.8.2.2. Error de calibración estimado

El error de calibración estimado (ECE) mide la diferencia en la precisión esperada y la confianza esperada dando como resultado un escalar. En la práctica se calcula como el promedio ponderado de la diferencia entre precisión y confianza de los intervalos según la Ec. 2.29, donde n es el número total de muestras en todos los contenedores. La calibración perfecta se logra cuando $ECE = 0$, es decir, $acc(B_m) = conf(B_m)$ para todos los contenedores m.

$$ECE = \sum_{m=1}^{M} \frac{B_m}{n} |acc(B_m) - conf(B_m)| \tag{2.29}$$

2.8.2.3. Entropía cruzada

La entropía cruzada no es una medida de calibración como tal, sino una medida estándar de la calidad de un modelo que se usa a menudo en la literatura de aprendizaje profundo y que ya fue presentada en la sección 2.5.4.2. Se menciona también aquí porque puede ser utilizada para calibrar un modelo a través de su minimización. La entropía cruzada también se conoce como probabilidad logarítmica negativa o negative log likelihood (NLL).

2.8.3. Métodos de calibración

Guo et al. [23] mencionaron en su trabajo varios métodos de calibración, en este informe dos son presentados: escalado de Platt y escalado de tempe-

ratura. Cada uno de ellos toma la forma de un modelo que podría verse como una extensión de uno anterior, al cual le corrige el error de calibración. La generación del modelo calibrado se realiza después de que el modelo original está completamente entrenado. Además, ambos métodos hacen uso de un conjunto de verificación (disociado del conjunto de entrenamiento) para obtener el modelo calibrado.

2.8.3.1. Escalado de Platt

El escalado de Platt aplica a la clasificación binaria y utiliza un modelo de regresión logística para asignar las probabilidades de salida de la red original a una probabilidad calibrada y reescalada. Sean $a, b \in \mathbb{R}$ escalares, el modelo reescalado se define según la Ec. 2.30, donde z_i es el logit (la salida de preactivación de la red original) del dato i y σ es la función sigmoide. El objetivo es optimizar a y b utilizando una pérdida como puede ser NLL sobre el conjunto de verificación.

$$\hat{q}_i = \sigma(az_i + b) \tag{2.30}$$

2.8.3.2. Escalado de Temperatura

El escalado de temperatura es aplicable a tareas de clasificación para K clases reescalando el logit vector z_i de la forma expuesta por la Ec. 2.31. En este caso, z_i^k es el logit asociado con el dato i para la clase k, T es el parámetro temperatura a ajustar y σ refiere a la función softmax. Al igual que el escalado de Platt, T se entrena con una función de pérdida sobre los datos de verificación.

$$\hat{q}_i = max_k \sigma(\frac{z_i^k}{T}) \tag{2.31}$$

Capítulo 3

Trabajos relacionados y datos

En el presente capítulo se presentan los trabajos relacionados y datos utilizados en este proyecto.

3.1. Trabajos relacionados

La segmentación sobre imágenes satelitales o aéreas obtenidas por remote sensing ha sido aplicada para la identificación de carreteras, vehículos o de edificios, siendo la última la que probablemente implique una mayor criticidad.

La información precisa y actualizada sobre la red de carreteras es importante para varias aplicaciones urbanas, como el mantenimiento de la navegación y la infraestructura. El advenimiento de remote sensing ha permitido la extracción de información de imágenes de alta resolución y altamente detalladas de las carreteras para actualizar las redes de carreteras urbanas. Una de las técnicas presentes en la extracción de carreteras implica tratar la extracción como un problema de segmentación de imágenes (dividido en carretera y no carretera). Jiuxiang, Anshuman et al. [28] presentaron un enfoque de dos pasos basado en la clasificación de la huella vial para extraer automáticamente redes de carreteras y detectar intersecciones partiendo de imágenes aéreas. Sukhendu et al. [15] entrenaron máquinas de vectores de soporte, o support vector machines (SVM), haciendo uso de dos características de las carreteras que denominaron como "sobresalientes": contraste espectral visible y trayectoria local lineal. Cheng et al. [13] también propusieron la utilización de SVM, combinándola con recortes de gráficos [7], que convierten la salida de las zonas de las carreteras de la SVM para hacerla más suave. Se observó que el método

de segmentación de imágenes para esta tarea se ve fácilmente afectado por oclusión de vegetación o sombras grandes, lo que conduce a la degradación en las tasas de reconocimiento. Además, debido a la introducción de algoritmos de postprocesamiento, otras características son confundidas fácilmente con carreteras [12].

La tarea de detección de vehículos guarda su trascendencia en aplicaciones de detección remota, incluida la gestión del tráfico, la planificación urbana, la utilización del espacio de estacionamiento, la vigilancia y la búsqueda y el rescate. Sin embargo, detectar vehículos en imágenes aéreas con precisión y rapidez no es trivial. Como las imágenes aéreas se toman desde la altura con una vista de arriba hacia abajo, los vehículos aparecen relativamente pequeños y una sola imagen puede contener muchos. Para el caso de imágenes satelitales, donde la distancia desde donde se genera la imagen aumenta, el desafío es aún mayor. Los primeros acercamientos al problema de detección de vehículos utilizaron métodos basados en el movimiento, donde a partir de la extracción del fondo de la imagen vehículos en una autopista pueden ser detectados [14, 55]. Otro tipo de método para abordar el problema son los basados en características definidas manualmente [11, 57, 76], pero este acercamiento al problema no logró alcanzar un equilibrio óptimo entre la capacidad de discriminar y la robustez, ya que debido a la falta de información semántica de alto nivel en términos de tipos de vehículos, todos los objetos detectados se tratan como vehículos equivalentes [10]. La técnica que supera a las anteriores en desempeño es la de aprender características del conjunto de entrenamiento automáticamente, que es lo que típicamente ofrecen las perspectivas basadas en redes neuronales profundas. Tang et al. [79] fueron unos de los autores que optaron por este enfoque, introduciendo un método basado en Faster R-CNN [61] que utiliza una red de "hiper región" con una combinación de funciones jerárquicas para mejorar la detección de vehículos pequeños. J. Zhu et al. [92] presentaron un método con detector SSD [47] de un solo paso que implementaba una red ResNet [26] para la extracción de características. Audebert et al. [2], por otro lado, utilizaron una red totalmente convolucional para la segmentación y detección de vehículos, seguida de una CNN para la clasificación de vehículos. Uno de los escenarios que dificultó la tarea de segmentación para estos enfoques es cuando existió una alta densidad de vehículos, con muchos automóviles acumulados en una zona pequeña. Esta situación tuvo como resultado que sea más difícil extraer vehículos individuales, dado que los modelos

tendían a generar máscaras de vehículos gruesas que generaban una superposición entre instancias de la misma clase por estar estas muy cerca en el espacio.

Con relación al reconocimiento de edificios, es una tarea que se utiliza para monitorear cambios en las áreas urbanas, para planificarlas, o para estimar la población que en estas vive. En general, los enfoques de extracción de edificios, al igual que para el caso de extracción de vehículos, se pueden dividir en dos categorías. Por un lado, los diseñados con las características tradicionales hechas a mano, donde la forma en que se construye el vector de características que contiene las diferentes características de cada clase es de gran importancia y, en particular, determina la precisión del resultado final. La mayoría de los trabajos que optaron por el planteamiento de características tradicionales generaron el vector de características a partir de filtros morfológicos [44], textura [85], descriptores de puntos [46], orientación de gradiente [4], entre otros. El otro enfoque se basa en las redes neuronales convolucionales profundas (CNN) [3, 39], que habían logrado los desempeños considerados como el estado del arte en muchas aplicaciones de visión artificial. Vakalopoulou et al. [84] propusieron un procedimiento de extracción de edificios supervisado basado en el marco ImageNet [16] con campo aleatorio de Markov. Emplearon combinaciones de bandas multiespectrales en el proceso de entrenamiento y mostraron resultados prometedores con validación cuantitativa. Huang et al. [30] entrenaron redes neuronales de deconvolución profunda (DeconvNet) de manera supervisada para extraer edificios de imágenes de remote sensing de muy alta resolución (VHRS), teniendo como resultado la extracción efectiva de edificios con diferentes escalas y varias apariencias topológicas. Otros autores [24, 45] optaron por modelos que siguen la arquitectura U-Net, obteniendo resultados superiores a otras arquitecturas de redes profundas previamente exploradas (CNN, FCN-8s, SegNet [86]) para el dataset de Massachusetts [62], también utilizado en este proyecto.

En lo que respecta a los conjuntos de datos presentes en este trabajo, el Massachusetts Buildings Dataset (MBD) [62] fue construido y aplicado en la tarea de desarrollar métodos para extraer automáticamente la ubicación de objetos como carreteras, edificios y árboles directamente a partir de las imágenes aéreas [54]. El conjunto Inria Aerial Image Labeling Dataset (IAD) [34] fue usado para probar la capacidad de generalización de métodos de clasificación de píxeles en el escenario donde las imágenes de entrenamiento pertenecen a

ciudades diferentes a las que están contenidas en las utilizadas en el conjunto de testeo [50]. Al mismo tiempo, las imágenes abarcan tipos de urbanizaciones distintas, que van desde áreas densamente pobladas (por ejemplo, el distrito financiero de San Francisco) hasta ciudades alpinas (por ejemplo, Lienz en el Tirol austríaco). Con fines similares a los dos anteriores, WHU Building Dataset (WHU) y Satellite dataset II East Asia (EAII) [87] se emplearon para evaluar la arquitectura SiU-Net, variante de U-Net, en la tarea de segmentación de imágenes [71]. Los autores obtuvieron un mejor desempeño empleando este mismo modelo para WHU, en comparación con el del obtenido utilizando los conjuntos MBD e IAD. Evaluando el modelo U-Net concretamente, los valores alcanzados para WHU en las métricas IoU, precision y recall fueron superiores en todos los casos a los conseguidos por los otros dos conjuntos.

Una dificultad que presenta la labor de reconocimiento de edificios es que ciertos elementos naturales (árboles, sombras, lagos) y artificiales (piscinas, contenedores, canchas), son similares a los objetos buscados, lo que hace que la tarea se complejice. Por otra parte, si bien una imagen de mayor calidad brinda más detalle sobre la escena, la cantidad excesiva de estos puede ser vista como un ruido con el que es necesario lidiar. Esto tiene un efecto adverso, ya que hace escalar los requerimientos de los modelos para clasificar correctamente [91].

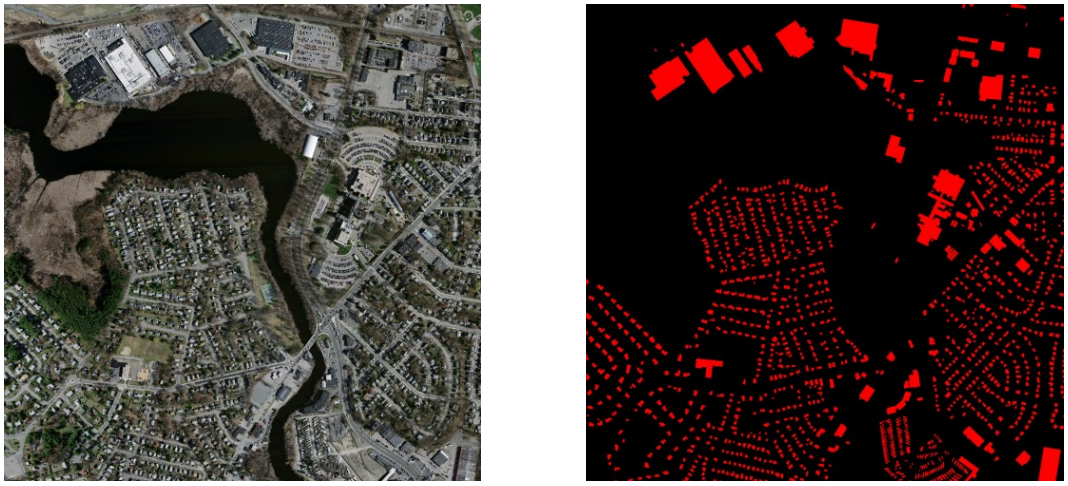
3.2. Datos

En esta sección se presentan los conjuntos de datos utilizados, así como también las características de cada uno. A partir de esta información se crearon los conjuntos de entrenamiento, validación y testeo empleados por los modelos implementados.

3.2.1. Massachusetts Buildings Dataset

El Massachusetts Buildings Dataset (MBD) [62] consiste de 151 imágenes aéreas del área de Boston (EEUU), cada una de las imágenes tiene un tamaño de 1500×1500 píxeles para un área de 2,25 kilómetros cuadrados. Por lo tanto, todo el conjunto de datos cubre aproximadamente 340 kilómetros cuadrados. Los datos están divididos aleatoriamente en un conjunto de entrenamiento de 137 imágenes, uno de test de 10 imágenes y uno de validación de 4 imágenes.

El conjunto de datos cubre principalmente áreas urbanas y suburbanas y edificios de diversos tamaños, incluidas casas individuales y garajes. Las imágenes anotadas cuentan con los tres canales RGB, teniendo los valores (255, 0, 0) para la clase Edificio y (0, 0, 0) para la clase No Edificio, lo que se puede observar en la figura 3.1.



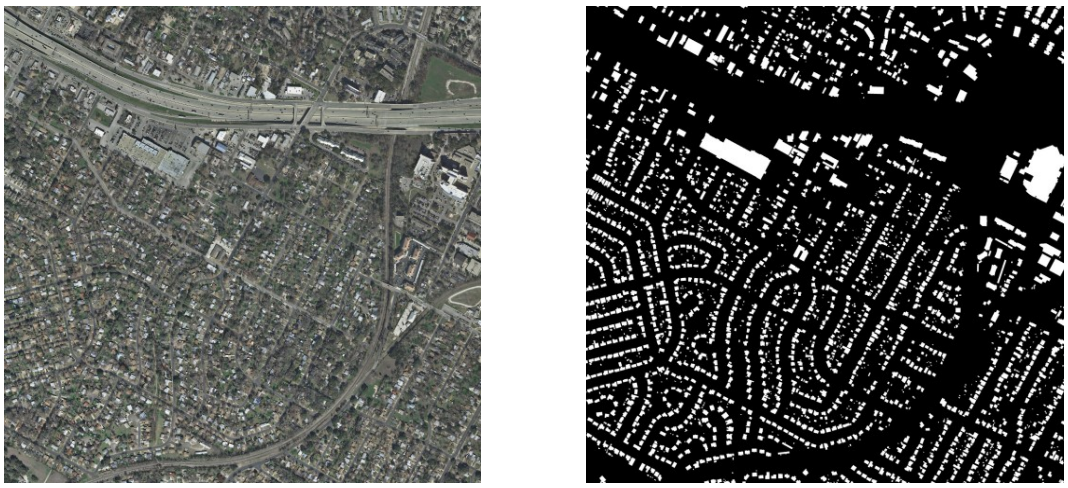
(a) **(b)**

Figura 3.1: (a) Imagen original del dataset de MBD. (b) Mapa anotado de la imagen original.

3.2.2. Inria Aerial Image Labeling Dataset

Las imágenes del conjunto Inria Aerial Image Labeling Dataset (IAD) [34] abarcan tipos de urbanizaciones distintas, que van desde áreas densamente pobladas (por ejemplo, el distrito financiero de San Francisco) hasta ciudades alpinas (por ejemplo, Lienz en el Tirol austríaco). El conjunto de entrenamiento consta de 180 imágenes de tamaño 5000 x 5000 píxeles donde cada una cubre una superficie de 1500m x 1500m. Se tienen 36 imágenes contiguas de tres canales (a color) de cada una de las siguientes regiones: Austin (EEUU), Chicago (EEUU), Condado de Kitsap (EEUU), Tirol Occidental (Austria) y Viena (Austria). Las imágenes anotadas son de un solo canal con valor 255 para la clase Edificio y 0 para la clase No Edificio, como se puede ver en la figura 3.2. El conjunto de test provisto no cuenta con su contraparte anotada y las imágenes allí contenidas pertenecen a regiones de Bellingham (EEUU), Bloomington (EEUU), San Francisco (EEUU), Innsbruck (Austria) y Tirol Oriental (Austria).

A fin de conseguir un conjunto de test anotado, se tomaron 10 imágenes del conjunto de entrenamiento original (dos de cada ciudad) para generarlo. De esta manera los conjuntos de entrenamiento, validación y test finales constan de 170, 20, y 10 imágenes respectivamente.



(a) **(b)**

Figura 3.2: (a) Imagen original del dataset IAD. (b) Mapa anotado de la imagen original.

3.2.3. WHU Building Dataset - Christchurch

El conjunto WHU Building Dataset (WHU) [87] de datos consta de más de 187.000 edificios independientes extraídos de imágenes aéreas con una resolución espacial de 0,3 de Christchurch, Nueva Zelanda. WHU está formado por 8.188 imágenes de 512 x 512 píxeles, donde el conjunto de entrenamiento posee 4.736 (130.500 edificios), el de validación 1.036 (14.500 edificios) y el de testeo 2.416 (42.000 edificios). Las imágenes anotadas son de un solo canal con valor 255 para la clase Edificio y 0 para la clase No Edificio, como se puede ver en la figura 3.3.

(a) **(b)**

Figura 3.3: (a) Imagen original del dataset WHU. (b) Mapa anotado de la imagen original.

3.2.4. Satellite dataset II East Asia

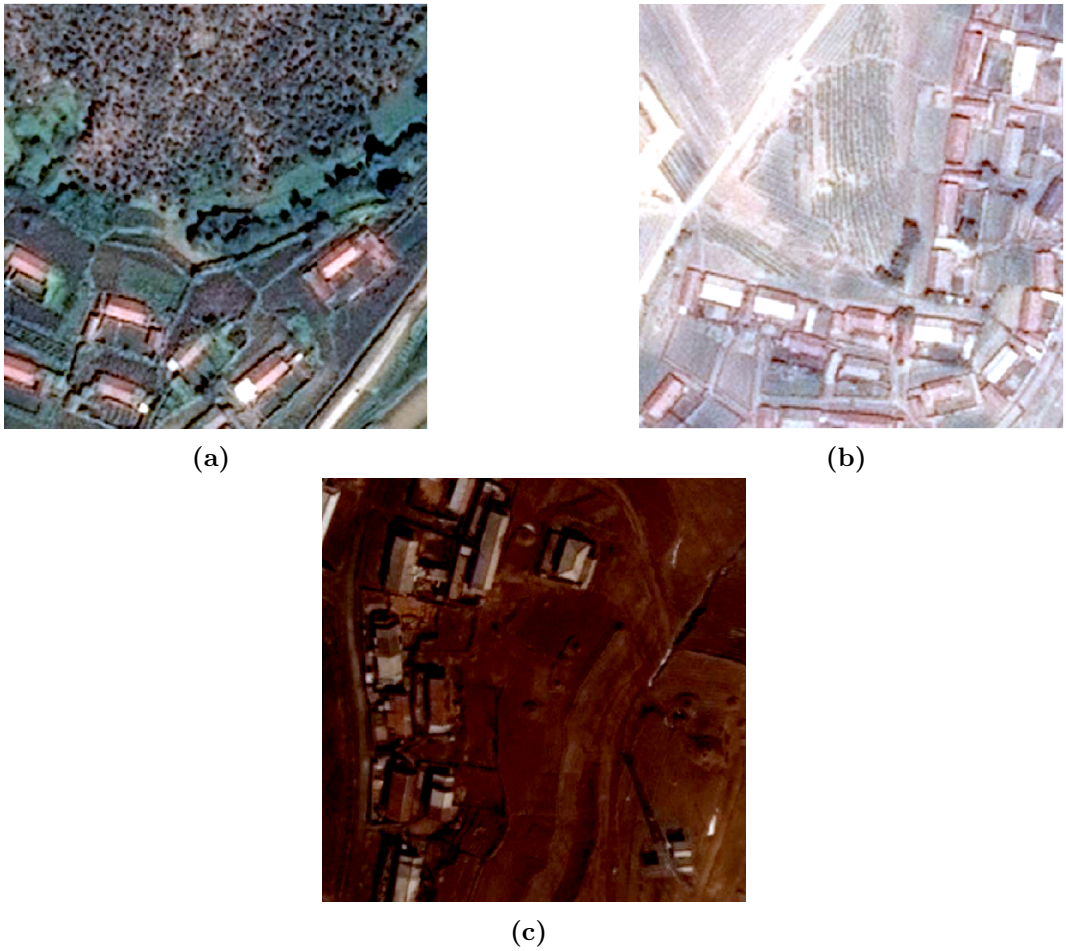
Satellite dataset II East Asia (EAII) fue publicado por los mismos autores de WHU [71]. Consta de seis imágenes de satélite contiguas que cubren 550 km^2 en el este de Asia con una resolución terrestre de 2,7 m. EAII fue diseñado principalmente para evaluar y desarrollar la capacidad de generalización de un método de aprendizaje profundo de diferentes fuentes de datos pero con estilos de construcción similares en la misma área geográfica. La imagen completa está dividida en 17.388 mosaicos de 512 $\times$ 512 px divididos entre conjunto de entrenamiento y test. El primero de los conjuntos cuenta con 21.556 edificios mientras que el otro con 7.529. Originalmente, ambos conjuntos de entrenamiento y test están divididos entre imágenes que tienen edificios y las que no. Para el entrenamiento de modelos en este proyecto solamente se consideraron los que tienen al menos un caso positivo en la imagen. De esta forma, el conjunto de entrenamiento se reduce a 3.135 imágenes y el de test a 903.

A diferencia de los anteriores conjuntos, EAII consta de imágenes satelitales (no aéreas), y abundan los casos de imágenes donde hay edificaciones escasas y localizadas de forma irregular entre ellas enmarcadas en terrenos no edificados y también irregulares en cuanto al color y forma.

De los ya presentados, WHU es el conjunto donde más zonas rurales hay, pero los territorios no edificados que allí figuran en la mayoría de los casos están cubiertos con árboles y pastos verdes de una tonalidad muy similar y terreno regular. En EAII, en cambio, existe una variedad mucho mayor: desde

caminos y plantaciones que convierten el territorio cubierto por vegetación en formas irregulares, hasta apariencias áridas y oscuras cuyo aspecto aparenta ser causa de las condiciones y calidad en que las imágenes fueron tomadas, y no solo como resultado de los elementos que conforman el terreno.

Si bien hay muchas imágenes de edificaciones residenciales que lo conforman, en la mayoría de los casos las edificaciones presentan una distribución irregular, acompañando caminos sinuosos o estando en zonas aisladas que no se ven en ninguno de los conjuntos anteriores, por lo que se estima que la planificación urbana para EAII fue mucho menor que para la de los casos anteriores. Ejemplos de EAII y las situaciones descritas anteriormente se observan en la figura 3.4.



(a) **(b)** **(c)**

Figura 3.4: (a) Las edificaciones se disponen de forma irregular, así como también lo es el color y relieve del terreno. (b) Se aprecia una sobre-saturación en la imagen, mostrando un tono de los elementos seguramente mucho más claro del real. (c) En contraste con (b), los elementos aparecen más oscuros y con una carga mayor del color rojo.

3.2.5. San José de las Matas

Con el fin de someter a evaluación a los distintos modelos estudiados en una ciudad donde probablemente no hubo planificación urbana para su desarrollo, se creó el dataset de San José de las Matas (SJM), municipio ubicado en la provincia de Santiago, República Dominicana. La superficie total abarca 3,41 km^2 aproximadamente y está dividida en imágenes de 640×640 px en su mayoría.

El conjunto fue generado a través de imágenes satelitales de Google Maps, y fue anotado utilizando la herramienta Labelbox [40], que permite dibujar segmentos sobre las imágenes originales. Para la tarea de identificación de los edificios a partir de los datos en bruto no se utilizó ninguna herramienta adicional, sino que las anotaciones fueron todas generadas de forma manual.

En San José de las Matas se aprecian edificaciones típicamente bajas, en su gran mayoría residenciales, dispuestas en diferentes orientaciones entre sí. Esto contrasta con imágenes que los datasets de MBD e IAD, por ejemplo, donde se ve que en muchos casos las paredes de las edificaciones están alineadas (ya sea en relación de perpendicularidad o en paralelo) con las calles que rodean la manzana donde se sitúan. Otra característica de SJM es que hay una presencia no menor de árboles, cuyas copas, en muchos casos superan la altura de los edificios, lo que genera que desde la perspectiva satelital la forma de sus techos, generalmente rectangular, se vea alterada (ejemplo en figura 3.5).

(a) **(b)**

Figura 3.5: (a) Imagen original del dataset SJM. (b) Mapa anotado de la imagen original.

Capítulo 4

Métodos considerados para el estudio

En el presente capítulo se describen los métodos utilizados para el estudio del problema tratado en este trabajo.

4.1. Introducción

Este capítulo desarrolla los principales conceptos de aquellas arquitecturas de redes neuronales profundas utilizadas para el estudio del problema de este proyecto. Empezando por la presentación de las redes convolucionales en general, se prosigue con las de otras que son extensiones de la primera con ciertas mejoras que se explican en cada sección. En particular, además de las redes convoluciones, el capítulo también introduce la redes ResNet (y sus variantes), ResNeXt, completamente convolucionales y U-Net. Finalmente, la última sección comenta el origen y objetivo del conjunto de datos ImageNet.

4.2. Redes neuronales convolucionales

La presente sección desarrolla los conceptos necesarios para entender el funcionamiento de las redes neuronales convolucionales.

4.2.1. Introducción

De forma análoga a las ANNs, las redes neuronales convolucionales o convolutional neural networks [41] (CNN) están compuestas por neuronas que

optimizan su desempeño en una tarea a través del aprendizaje de los pesos de cada capa. Así mismo, la última capa también consiste en una función de pérdida asociada con las clases a categorizar. La principal diferencia con las ANNs consiste en que las CNNs son especialmente utilizadas para el reconocimiento de imágenes. Esto permite utilizar codificaciones de atributos específicos de imágenes en la arquitectura, lo que la hace más apropiada para las tareas en este campo.

Una limitante importante de las ANNs convencionales es que tienden a tener complicaciones con la complejidad computacional que implica procesar imágenes de alta resolución. Esto se da porque el dato de entrada, de tres dimensiones (largo, ancho y canal/color) debe convertirse en uno de una sola dimensión para poder ser procesado por la ANN. Como consecuencia de dicha transformación se tienen dos resultados negativos: el número de parámetros a entrenar crece drásticamente según crece el tamaño de la imagen y se pierden sus características espaciales.

4.2.2. Arquitectura

La arquitectura de una CNN, representada en la figura 4.1 consiste de tres clases de capas: las de convolución, las capas de agrupación (pooling) y las completamente conexas. Las funcionalidades de una arquitectura tipo de CNN pueden dividirse en cuatro secciones:

1. La capa de entrada contiene los valores de los píxeles de la imagen. La funcionalidad de esta capa es la misma que para una ANN convencional.
2. La capa de convolución determina la salida de las neuronas que están conectadas a regiones localizadas de la entrada mediante el cálculo del producto escalar entre sus pesos en dicha región. Usualmente se utiliza la función ReLu como función de activación sobre la salida de la capa anterior.
3. La capa de pooling realiza una reducción en la dimensionalidad (también llamada downsampling) de la entrada recibida, reduciendo la cantidad de parámetros para esa activación.
4. Las capas completamente conexas tienen la misma función que para el caso de las ANNs convencionales: generar puntajes para cada clase que son utilizados para la tarea de clasificación.

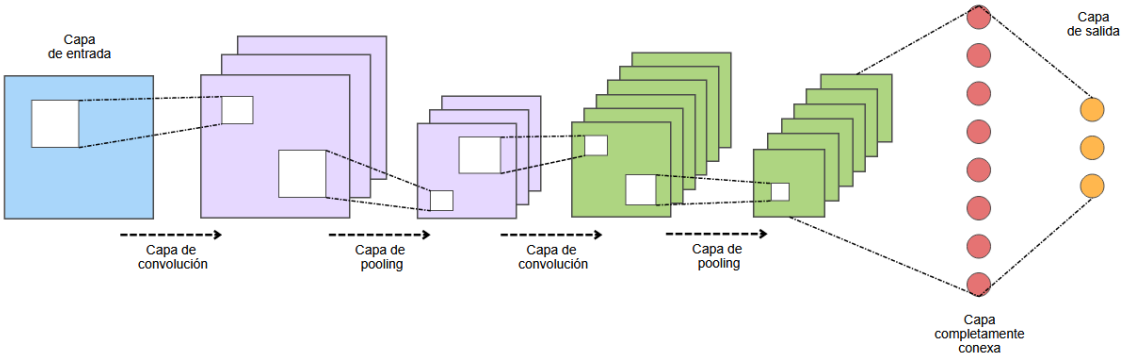


Figura 4.1: Arquitectura de una CNN con dos ciclos de capa de convolución y pooling.

4.2.2.1. Capas de convolución

Estas capas se enfocan en el uso de núcleos (o kernels), cuyos parámetros deben ser aprendidos. Los núcleos usualmente tienen una dimensionalidad espacial pequeña, pero comprenden a toda la profundidad (canales) de la entrada. Al llegar los datos a la capa, esta convoluciona cada kernel con la entrada, generando un mapa de activación bidimensional. La convolución es el resultado del producto interno entre los valores de entrada del kernel y los de los datos. Como se mencionó, cada kernel tiene una dimensionalidad menor a la de la entrada, por lo que para calcular el producto se debe tomar una región de la entrada del mismo tamaño; a esta región se la denomina como el campo receptivo de la neurona. Para un kernel, el resultado de la convolución es la aplicación del producto interno entre sí mismo y cada campo receptor, donde el campo receptor $i+1$ tiene un corrimiento de s píxeles con respecto al campo i, siendo s el tamaño de paso o stride. De esta manera, cada neurona en una capa convolucional solo está conectada a una pequeña región del volumen de entrada.

En general, las capas de convolución optimizan su salida con el ajuste de tres hiperparámetros: stride, ya definido anteriormente, profundidad y relleno. La profundidad refiere a la cantidad de kernels a aplicar en cada paso de convolución. El relleno, en cambio, es la acción de agregar ceros a los bordes de la imagen, de esta manera se tiene un mejor control de su dimensionalidad, lo que muchas veces es necesario para que la convolución funcione.

Fijados los valores de stride, profundidad y relleno, se puede calcular la dimensión de la salida O de cada capa de convolución, la cual está definida

por la Ec. 4.1, donde W es el tamaño del volumen de entrada, F el tamaño del campo receptor, P la cantidad de relleno agregado y s el stride. Observar que es necesario que O sea un valor entero para que la convolución se pueda calcular correctamente, en caso contrario los campos receptores al borde de la imagen quedarían de menor tamaño.

$$O = \frac{W - F + 2P}{s + 1} \tag{4.1}$$

4.2.2.2. Capas de pooling

El objetivo de las capas de pooling es el de gradualmente reducir la dimensionalidad de la representación, lograr una reducción también en los parámetros a aprender, la complejidad general del modelo y la posibilidad de generar sobreajuste. Para ello, el proceso busca mantener la información más importante, desechando el resto. Debido a la naturaleza destructiva del pooling no se deben definir ventanas muy grandes, ya que en ese caso el proceso de pooling descartaría mucha información.

El proceso de pooling consiste en tomar una ventana de la entrada de un tamaño definido y aplicarle una operación, típicamente max, entre sus valores. De forma similar al proceso de convolución, la ventana se irá deslizando por la entrada, quedándose con el resultado de la operación en cada paso. Una vez procesada toda la entrada, se tiene como resultado una salida de menor dimensionalidad con la información más importante de cada una de las ventanas originales.

4.2.2.3. Capas completamente conexas

Las capas completamente conexas funcionan igual que para el caso de las ANNs presentado en la sección 2.5.2, con la diferencia que tienen como entrada las activaciones resultado de la aplicación secuencial de varias capas de convolución y pooling.

Los distintos modelos existentes utilizan diversas combinaciones de capas de convolución y pooling. Las variaciones no solo son a nivel del tamaño de los kernels o ventanas de pooling, sino también sobre la cantidad de veces y orden en el que se emplean. A la secuencia de aplicación de las capas de convolución y pooling de una red usualmente se la denomina backbone.

4.3. Redes neuronales residuales

Esta sección presenta las redes neuronales residuales.

4.3.1. Introducción

La profundidad de las redes neuronales es un factor importante para su desempeño y ha demostrado ser uno de los parámetros que permite aprender funciones cada vez más complejas [72, 77]. No obstante, el incremento del número de capas de profundidad a una red trae dos problemas inmediatos: el del desvanecimiento de gradiente [5, 19] y el de degradación de desempeño.

El desvanecimiento del gradiente es generado porque a medida que se profundiza en la red, los gradientes desde donde se calcula la función de pérdida se reducen fácilmente a cero después de varias aplicaciones de la regla de la cadena, lo que impide la convergencia del modelo. Si bien existen técnicas ya probadas que logran abordar este problema efectivamente [19, 42, 67], se observó que aún aplicándolas, cuanto mayor es la profundidad de la red, su precisión igualmente se saturaba y luego se degradaba rápidamente [25, 26, 75]. También se probó que la degradación no era debido al sobreajuste, ya que a mayor cantidad de capas peor era el error de entrenamiento.

Como una solución a estos problemas surgió la arquitectura ResNet (Residual Networks) [26] con sus variantes. La idea principal de ResNet es introducir una conexión de acceso directo de identidad (skip connection), que omite una o más capas, como se muestra en la figura 4.2. Con ResNets, los gradientes pueden fluir directamente a través de las conexiones de salto hacia atrás desde las capas posteriores a los filtros iniciales. Formalmente, si $H(x)$ es el mapa de asignaciones buscado, se genera que las capas no lineales se ajusten a otro mapeo distinto, $F(x) = H(x) + x$.

Kaiming et al. [26] argumentaron que las conexiones de salto en ResNet resuelven el problema de la desaparición del gradiente en las redes neuronales profundas al permitir que el gradiente fluya por el atajo creado entre capas. Las conexiones de salto también permiten que el modelo aprenda las funciones de identidad, lo que garantiza que la capa superior funcionará al menos tan bien como la capa inferior, y no peor. Por ejemplo, dado un escenario donde se tiene una red llana y una red profunda que mapea una entrada x a una salida y usando la función $H(x)$, se desea que la red profunda funcione al menos tan bien como la llana y no degrade el rendimiento como sucede en

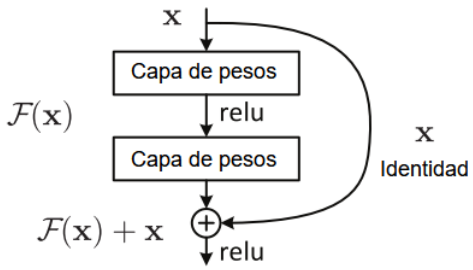


Figura 4.2: Conexión de acceso directo de identidad - skip connection. Imagen basada en He et al. [26]

el caso de las ANNs (sin bloques residuales). Una forma de lograrlo es si las capas adicionales en una red profunda aprenden la función de identidad y, por lo tanto, su salida es igual a las entradas, lo que no les permite degradar el rendimiento incluso con capas adicionales. En el trabajo de He K. et al. [26] se reportó que los bloques residuales hacen que sea excepcionalmente fácil para las capas aprender funciones de identidad.

4.3.2. Arquitectura

Existe más de un tipo de ResNet, donde en cada una varía principalmente la cantidad de bloques por capa y tamaño de convoluciones aplicadas en cada uno de ellos. Sin embargo, se presenta la variante Resnet34 ya que es suficiente para comprender las nociones detrás de las arquitecturas ResNet.

ResNet cuenta primeramente con una convolución 7×7 de 64 kernels y un stride de 2, seguido de un 3×3 max pool con stride de 2. Luego siguen cuatro agrupaciones A_i constituidas por:

- A_1: 6 convoluciones 3×3 con mapa de asignación con dimensión de 64.
- A_2: 8 convoluciones 3×3 con mapa de asignación con dimensión de 128.
- A_3: 12 convoluciones 3×3 con mapa de asignación con dimensión de 256.
- A_4: 6 convoluciones 3×3 con mapa de asignación con dimensión de 512.

Todas las agrupaciones aplican un skip connection cada dos capas, como se muestra en la figura 4.2. En estas capas el downsampling no se realiza con una capa de pooling sino que estableciendo un stride de 2 para la primer capa de convolución de las agrupaciones A_2, A_3 y A_4. Para el caso donde la conexión de acceso directo se da entre dos capas de distinta dimensión, los autores proponen dos estrategias: la conexión sigue siendo la identidad pero se rellena

con ceros hasta igualar las dimensiones, o se realiza una proyección lineal por medio de una capa de convolución 1×1, obteniendo el mismo resultado. Por último se tiene una capa de pooling con operación average (promedio) y una capa completamente conexa con softmax como función de activación. La arquitectura completa de ResNet34 se puede ver en la figura 4.3 y la composición de las otras variantes ResNet en la tabla 4.1.

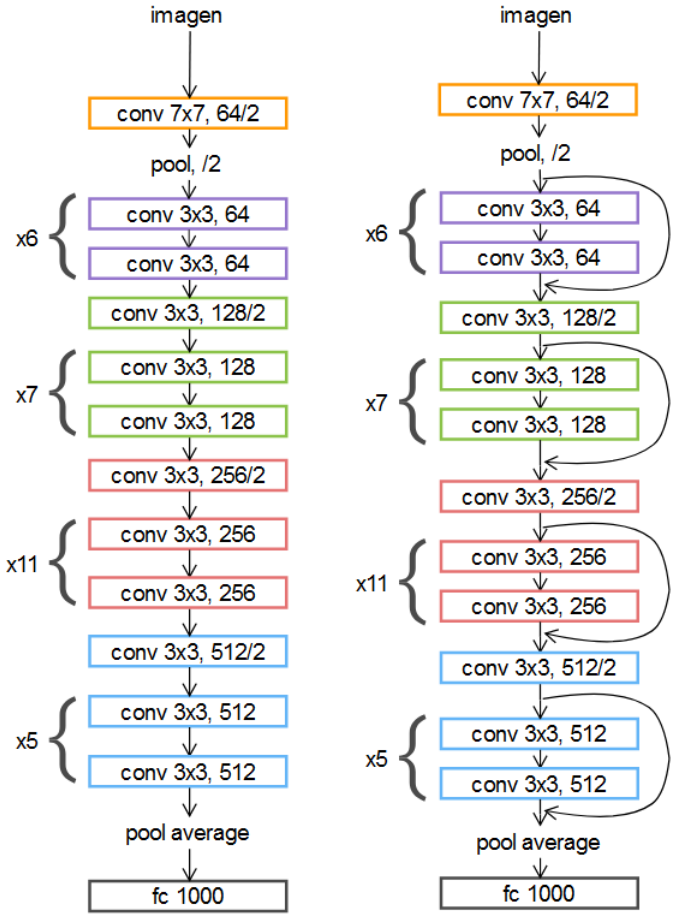


Figura 4.3: Arquitectura de 34 capas sin conexiones residuales (izquierda). ResNet34, contraparte de la anterior con conexiones residuales (derecha).

4.4. Redes neuronales residuales - variante ResNeXt

En la presente sección se introduce ResNeXt, una arquitectura que surgió como una extensión de ResNet.

Etapa	*Tamaño salida*	*ResNet34*	*ResNet50*	*ResNet101*
conv1	112×112	7×7, 64, stride 2		
conv2_x	56×56	3×3 max pool, stride 2		
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4_x	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5_x	7×7	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, fc (1000-d), softmax		

Tabla 4.1: Composición de arquitecturas ResNet utilizadas en este proyecto. Los bloques de convolución se muestran entre corchetes con el número de bloques apilados. El downsampling se realiza mediante conv3 1, conv4 1 y conv5 1 con un stride de 2.

4.4.1. Introducción

ResNeXt es una red neuronal homogénea que reduce la cantidad de hiperparámetros requeridos por ResNet convencional. Esto se logra mediante el uso de la "cardinalidad", una dimensión adicional además del ancho y la profundidad de ResNet que define el tamaño del conjunto de transformaciones.

4.4.2. Arquitectura

La arquitectura ResNeXt [90] surgió como una extensión de ResNet que reemplaza el bloque residual estándar con uno que aprovecha la estrategia "dividir-transformar-fusionar". Resumidamente, en lugar de realizar convoluciones sobre el mapa de asignaciones de entrada completo, la entrada del bloque se proyecta en un conjunto de representaciones (canales) de menor dimensión a las que se les aplica por separado filtros convolucionales para finalmente fusionar los resultados. A cada uno de estos conjuntos se los denomina caminos de convolución y la cantidad de caminos es un hiperparámetro en sí mismo llamado cardinalidad (C). Cabe destacar que todos los caminos tienen la misma topología de convolución.

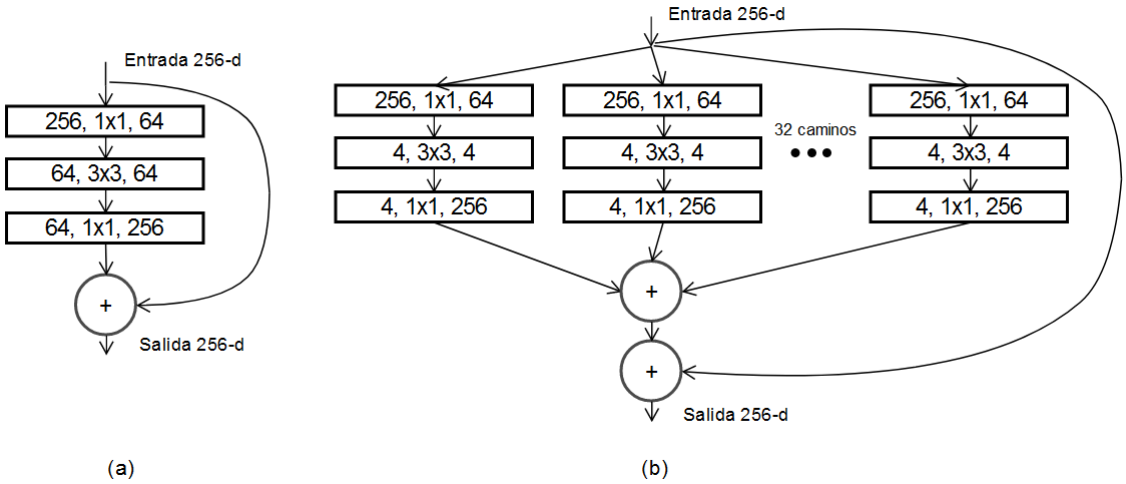


Figura 4.4: (a) bloque residual de ResNet50. (b) bloque de ResNeXt con cardinalidad = 32. Cada capa es representada como (# canales de entrada, tamaño del kernel, # canales de salida).

En la figura 4.4 se observa la comparativa de un bloque residual de ResNet50 y uno de ResNeXt con $C = 32$. En el bloque ResNeXt se puede ver que en una primera instancia la entrada es dividida en 32 canales de dimensionalidad 4 ("dividir"). Luego, las convoluciones son realizadas sobre los canales de menor dimensionalidad en cada uno de los caminos por separado ("transformar") para finalmente recuperar nuevamente su dimensionalidad original y ser sumados entre sí ("fusionar"). Por último, se aplica el skip connection, igual que para el caso de ResNet. Las estrategias posibles de convolución son las mismas que las presentadas en la sección 4.3. Una comparación más extensiva de ambas arquitecturas completas se puede observar en la tabla 4.2.

4.5. Redes completamente convolucionales

Esta sección desarrolla modelo de redes neuronales completamente convolucionales.

4.5.1. Introducción

Long et al. [48] expresaron que el principal problema de tener capas completamente conexas (MLP) en una red convolucional es que estas aprenden recogiendo información global de la red, pero carecen de la noción de localidad.

Etapa	*Tamaño salida*	*ResNet50*	*ResNeXt50 (32×4d)*
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	
		$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128, C = 32 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3x	28×28	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256, C = 32 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512, C = 32 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
conv5x	7×7	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 1024 \\ 3 \times 3, 1024, C = 32 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, fc (1000-d), softmax	

Tabla 4.2: (Izquierda) ResNet-50. (Derecha) ResNeXt-50. Dentro de los corchetes está representada la estructura de cada bloque residual y fuera de ellos el número de bloques apilados en cada etapa. $C = 32$ denota la cardinalidad.

Los autores plantean un mecanismo para mantener el concepto de localidad presente en el aprendizaje: sustituir la MLP por capas convolucionales 1×1. De esta forma en las redes completamente convolucionales (FCN por sus siglas en inglés) las capas se adicionan como un filtro más que se espera respete la localidad, aspecto trascendente en el contexto de segmentación de imágenes. Otro punto de mejora respecto a las CNNs con MLP es que las FCNs no necesariamente deben recibir entradas cuyas dimensiones son de tamaño fijo, sino que pueden variar.

4.5.2. Downsampling y Upsampling

Las CNNs convencionales aplican dos o más capas de convolución y pooling, lo que reduce el tamaño de la imagen a medida que esta pasa por ellas. Esto se puede ver como la conversión de una imagen de alta resolución a una de baja que deseablemente mantiene las características más importantes de la primera, ayudando a los filtros en las capas más profundas a enfocarse en

campos receptores (contextos) más grandes. Por otro lado, el número de canales aumenta gradualmente tras cada convolución, y de este modo se facilita la extracción de características más complejas. Este proceso es el denominado downsampling e, intuitivamente, podemos decir que tras aplicarlo el modelo entiende mejor el "QUÉ" está representado en la imagen, pero pierde información acerca de el "DÓNDE". Sin embargo, en el contexto de segmentación de imágenes cada píxel es clasificado, por lo que la noción de ubicación de los objetos es fundamental. Para abordar esta tarea, es necesario convertir la imagen de baja resolución obtenida en el downsampling de nuevo a una de alta resolución y poder recuperar la información de localización. Este proceso es el denominado upsampling y existe más de un método para realizarlo [31, 37, 83].

4.5.3. Arquitectura

Los autores presentan tres variantes de arquitectura FCN que comparten el camino de downsampling conformado por dos ciclos de convolución doble y pooling, seguidos de dos ciclos más de convolución triple y pooling. Por último, se tiene una convolución doble, seguida de una final hacia el mapeo de clases a clasificar. Las distinciones entre las arquitecturas expuestas están en la etapa de upsampling como se aprecia en la figura 4.5.

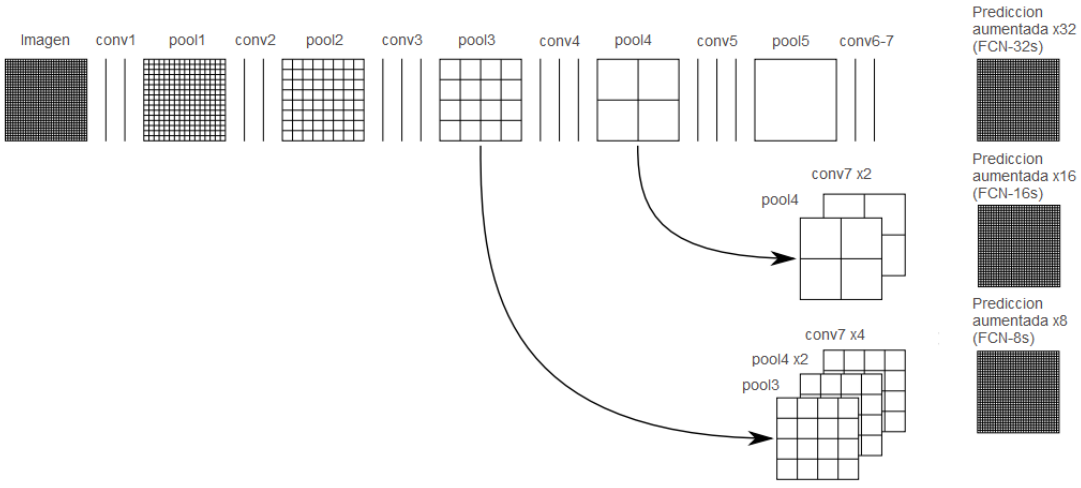


Figura 4.5: Primera fila: FCN-32s. Segunda fila: Operación adicional a FCN-32s para obtener FCN-16s. Tercera fila: Operación adicional a FCN-16s para obtener FCN-8s. Imagen basada en [48].

Las arquitecturas presentadas por Long et al. fueron tres:

- FCN-32s: se hace un upsampling $\times 32$ a la salida de la última capa de convolución para llegar a las dimensiones originales de la entrada.
- FCN-16s: se hace un upsampling $\times 2$ a la salida de la última capa de convolución, luego se le suma el resultado de la capa 4 de pooling y por último se aplica upsampling $\times 16$ a ese resultado para llegar a las dimensiones originales de la entrada. Es posible notar que al utilizar los valores de un estado intermedio en una etapa posterior se está recurriendo a la técnica de skip connection.
- FCN-8s: además del skip connection presente en FCN-16s, se adiciona uno que suma el upsampling $\times 4$ de la convolución 7, el upsampling $\times 2$ de la capa 4 de pooling, y la capa de pooling 3. Por último se aplica upsampling $\times 8$ al resultado anterior para, una vez más, llegar a las dimensiones originales de la entrada.

En el trabajo publicado también se menciona que utilizar conexiones entre diferentes capas mejora el detalle de segmentación, por lo que FCN-8s, al contar con dos operaciones de este estilo, brinda un desempeño superior frente a las otras versiones.

4.6. Redes U-Net

Esta sección introduce el modelo de redes neuronales U-Net.

4.6.1. Introducción

La red U-Net fue por primera vez presentada por Ronneberger et al. [63] para la tarea de segmentación de imágenes en el campo de la bio-medicina. Su arquitectura está basada en las redes completamente convolucionales (FCN) [48] y puede entrenarse con pocas imágenes haciendo un fuerte uso de la aumento de datos. La arquitectura U-Net superó el desempeño de redes convolucionales en el desafío ISBI del año 2015 [35] de segmentación de estructuras neuronales en imágenes de microscopios electrónicos, lo que les valió el premio de dicha edición.

4.6.2. Arquitectura

Como se muestra en la figura 4.6, la arquitectura U-Net consta de un camino de contracción (mitad izquierda, también llamado encoder) y uno de expansión (mitad derecha, también llamado decoder). El encoder es similar al de la arquitectura FCN y su objetivo es el de capturar el contexto en la imagen, mientras que el decoder tiene la función de precisar la ubicación de los elementos.

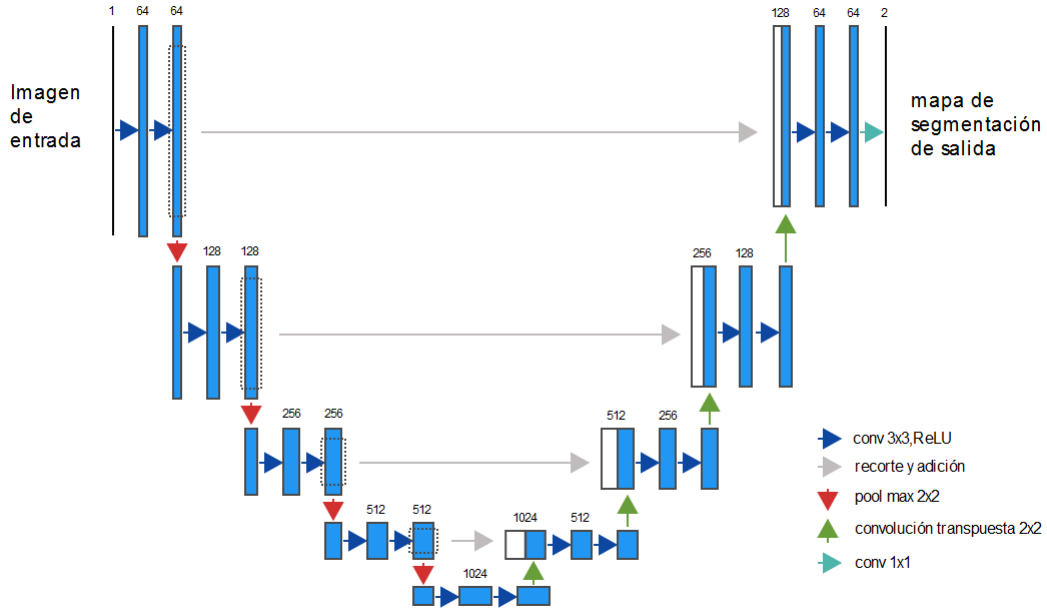


Figura 4.6: Arquitectura U-Net (ejemplo para 32×32 píxeles). Cada cuadro azul corresponde a un mapeo de características multicanal. El número de canales se indica en la parte superior de la caja. El tamaño x-y se proporciona en el borde inferior izquierdo del cuadro. Los cuadros blancos representan mapas de características copiados. Las flechas indican las diferentes operaciones descritas en la imagen. Imagen basada en [63].

El encoder consiste de la aplicación iterada de dos convoluciones 3×3 sin relleno, cada una seguida de la función ReLU y un max pooling de 2×2 con stride de 2 para el downsampling. Antes de aplicar el pooling, se hace un recorte de la salida obtenida al momento. El recorte se realiza para igualar las dimensiones con el decoder y poder aplicar la operación skip connection. Si las convoluciones se hubieran realizado con relleno, el recorte podría no ser necesario.

El decoder consta de un aumento de resolución de muestreo (upsampling) del mapa de activación obtenido por medio de una convolución transpuesta 2×2 que reduce a la mitad los canales de activación, continuado por la concatenación con el recorte correspondiente del encoder (operación skip connection), y luego dos convoluciones 3×3, cada una seguida de la función ReLU. La última capa es una convolución 1×1 utilizada para mapear cada vector de activación al número deseado de clases.

4.7. ImageNet

ImageNet [16] es un conjunto de datos público que contiene más de 14 millones de imágenes anotadas de acuerdo con la jerarquía de WordNet [89]. Desde 2010, el conjunto de datos se utiliza en el ImageNet Large Scale Visual Recognition Challenge (ILSVRC), una competición que sirve como referencia en clasificación de imágenes y detección de objetos. Las anotaciones de ILSVRC se dividen en una de dos categorías: (1) anotación a nivel de imagen de una etiqueta binaria para la presencia o ausencia de una clase de objeto en la imagen, por ejemplo, "hay perros en esta imagen" pero "no hay tigres", y (2) anotación a nivel de objeto de un cuadro delimitador y una etiqueta de clase alrededor de las instancias de objeto en la imagen, por ejemplo, "hay una frutilla centrada en la posición (20,25) con un ancho de 50 píxeles y una altura de 30 píxeles", un ejemplo puede encontrarse en la figura 4.7. Los dos tipos anotaciones permiten trabajar sobre el dataset principalmente en las siguientes dos tareas:

- Clasificación de imágenes: los algoritmos producen una lista de categorías de objetos presentes en la imagen.
- Detección de objetos: los algoritmos producen una lista de categorías de objetos presentes en la imagen junto con un cuadro delimitador alineado con el eje que indica la posición y escala de cada instancia de cada categoría de objeto.

A partir del ILSVRC, se han presentado numerosos modelos que hoy son tomados como de referencia en el procesamiento de imágenes. Algunos de ellos son AlexNet [39] (ILSVRC-2012), Inception [77] y VGG [72] (ILSVRC-2014), ResNet [26] (ILSVRC-2015), ResNeXt [90] (ILSVRC-2016).

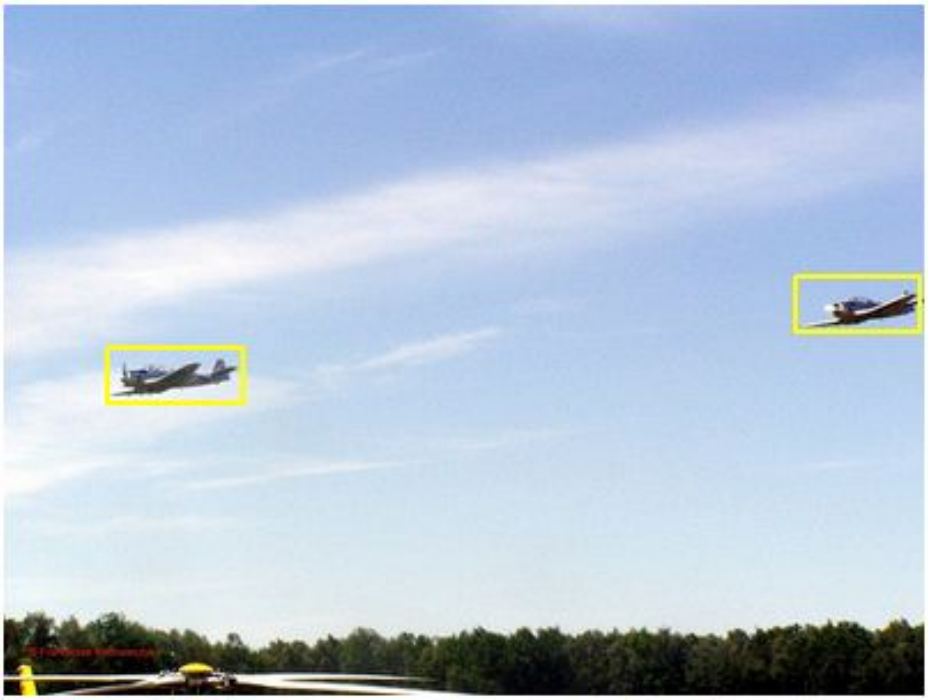


Figura 4.7: Dos tipos de anotaciones son posibles para la imagen: (1) "hay aviones" y "no hay tigres" (u otros objetos ausentes) y (2) "hay un avión centrado en la posición (x_1, y_1) con un ancho de a píxeles y una altura de h píxeles" por cada avión en la imagen.

Capítulo 5

Implementación

El presente capítulo contiene las técnicas de procesamiento de datos, modelos y bibliotecas de software utilizadas en este trabajo.

5.1. Introducción

Se propone plantear el problema como uno de aprendizaje supervisado partiendo de los conjuntos de datos expuestos en la sección 3.2 para utilizarlos directamente o aplicarles alguna técnica de data augmentation para generar nuevos conjuntos. Siguiendo el principio de "no hay almuerzo gratis" se tomaron en cuenta más de un modelo de redes neuronales profundas para resolver el problema, con el fin explorar cada uno de ellos y llegar al que obtiene las mejores predicciones en función de las métricas definidas.

Todo el proceso está compuesto por distintas etapas donde en cada una, por separado pero de forma secuencial, se buscó seleccionar las mejores arquitecturas, datasets e hiperparámetros para la tarea planteada. Luego, la cuarta etapa tuvo como objetivo el calibrar los modelos conseguidos hasta el momento y, por último, una etapa final donde se utilizó la votación de modelos para la segmentación de cada imagen. El contenido de cada etapa es explicado en detalle en la sección 5.4.

5.2. Procesamiento de datos

Como ya se adelantó en la sección 2.3, el problema a resolver es uno de aprendizaje supervisado. Para plantrearlo de esa manera, se debieron conside-

rar los datos como pares $(x_1, y_1), ..., (x_n, y_n)$ con $(x_i, y_i) \in X \times Y$ donde x_i es un vector que representa a la imagen i-ésima, e y_i representa a su mapa de clases. Cada vector x_i es de dimensión $l_i \times w_i \times 3$ donde l_i y w_i son la cantidad de píxeles que tiene la imagen i de largo y ancho respectivamente, y 3 refiere a los canales RGB. Cada y_i mantiene una dimensión $l_i \times w_i \times 2$, siendo 2 el tamaño de dimensión obtenido al aplicar one hot encoding a las posibles clases de cada píxel: Edificio y No Edificio.

5.2.1. Preprocesamiento con datos ImageNet

En tareas de aprendizaje automático es habitual encontrarse con que los atributos de las instancias de entrenamiento están representados en distintas unidades o escalas. De no ser tratadas adecuadamente estas variantes entre los atributos puede afectar sensiblemente el desempeño del modelo, ya que es probable que los atributos que tomen valores de magnitudes mayores tengan también un mayor impacto en los resultados de las predicciones. Una técnica común para minimizar este efecto es la estandarización, cuya aplicación tiene como resultado que la distribución de los datos tenga una media de 0 y desviación estándar de 1. Si bien existe más de una forma de estandarizar, en este trabajo se usó la puntuación Z o Z-score, calculado según la Ec. 5.1 y donde μ es la media de las instancias y σ la desviación estándar.

$$\hat{x} = \frac{x - \mu}{\sigma} \tag{5.1}$$

Si bien en el contexto de las imágenes no hay problemas de escala o unidades ya que todos los valores posibles están comprendidos entre 0 y 255, sí puede haber diferencias en las condiciones en que fueron tomadas las imágenes. Por ejemplo, si una imagen tiene más iluminación que otra por factores ajenos a los objetos que allí aparecen (clima, sombras, etc.), entonces aplicando la estandarización los efectos agregados por los agentes externos serían minimizados.

Un criterio habitual al trabajar con conjuntos de datos de imágenes es el de reutilizar los valores μ y σ de ImageNet, los cuales son calculados sobre las millones de imágenes que contiene. Como las imágenes provienen de diversos dominios, y los conjuntos MBD, IAD y WHU pertenecen a un dominio particular, el de imágenes aéreas, parecía oportuno estandarizar según los valores de media y desviación estándar particulares a esos conjuntos. Sin embargo,

luego de experimentar algunos modelos, se observó que la diferencia entre utilizar los valores de media y desviación estándar propios de los datasets o los de ImageNet no tenía un impacto significativo sobre los resultados. Por este motivo en la implementación los datos fueron estandarizados según los valores de ImageNet.

5.2.2. Data Augmentation

Se tomaron dos enfoques a la hora de aplicar técnicas de data augmentation: (1) la generación de nuevos archivos que conforman un nuevo conjunto a partir de uno original con ciertas transformaciones (offline augmentation), y (2) la aplicación de transformaciones sobre la imagen durante la etapa de entrenamiento (online augmentation).

A partir de (1) se obtuvieron como resultado conjuntos cuyas imágenes eran de menor dimensión respecto a las originales y en algunos casos con una transformación de canales RGB a BGR. La reducción de tamaño no solo buscaba la generación de un nuevo conjunto para contar con más datos, sino que para el caso de IAD además existía una limitante por ser sus imágenes muy grandes (5000×5000). Para poder procesarlas durante el entrenamiento se observó que los ambientes disponibles debían hacer un recorte de máximo (aproximadamente) de 2016×2016 para no quedarse sin memoria, por lo que mucha información se perdía. Para WHU y EAII no se generaron datos ya que se consideró que estaban lo suficientemente representados con las muestras originales. Los conjuntos resultado de aplicar offline augmentation son los siguientes:

- IAD2500: recortes en mosaico de tamaño 2500×2500 de imágenes originales de IAD e inversión de canales RGB a BGR.
- IAD1250: recortes en mosaico de tamaño 1250×1250 de imágenes originales de IAD e inversión de canales RGB a BGR.
- IAD1000: recortes en mosaico de tamaño 1000×1000 de imágenes originales de IAD.
- MBD750: recortes en mosaico de tamaño 750×750 de imágenes originales de MDB.
- MBD500: recortes en mosaico de tamaño 500×500 de imágenes originales de MDB.

Por otro lado, las transformaciones online se aplican a cada imagen de entrenamiento y consisten de:

- Un recorte aleatorio de tamaño 480×480.
- Con una probabilidad de 75 %, aplicar una (y solo una) de las siguientes transformaciones:
  - inversión horizontal.
  - inversión vertical.
  - rotación de 90° aleatoria.

5.2.3. Representatividad

Dado que la cardinalidad de los conjuntos en algunos casos difiere bastante -un ejemplo de esto puede ser el caso de IAD1000 (3750 imágenes) y MBD (137 imágenes)- en cada partición de entrenamiento se optó por aumentar la proporcionalidad de aquellos conjuntos que quedarían muy poco representados debido a las pocas muestras con las que cuentan. Por ejemplo, si consideráramos el 25 % tanto de IAD1000 como de MBD tendríamos que para esa instancia de entrenamiento nos quedaríamos con 937 y 34 imágenes respectivamente, lo cual dejaría muy pocas imágenes de MDB para entrenar. Con el fin de mitigar esta situación, se optó por seguir el criterio definido por el algoritmo 1 a la hora de segmentar los conjuntos de entrenamiento.

Algoritmo 1: Algoritmo para representatividad de datasets

```
1 sample_k_global ← (total_imgs_datasets × portion)/cant_datasets;
2 final_dataset ← [];
3 para ds en datasets hacer
4     si sample_k_global ≥ tamano_ds entonces
5         final_dataset ← final_dataset + ds;
6     en otro caso
7         sampled_ds ← obtener_aleatorios(ds, sample_k_global);
8         final_dataset ← final_dataset + sampled_ds;
```

El algoritmo 1 comienza definiendo la cantidad mínima de ejemplos que serían necesarios para cada dataset para cumplir con la representatividad esperada en el nuevo conjunto de datos que los unifica parcialmente. La variable *sample_k_global* entonces es el mismo valor para todos los datasets y se calcula

según la línea (1), donde *total_imgs_datasets* es la cantidad total de imágenes de todos los datasets sumada, $portion \in (0, 1]$ el porcentaje que se busca obtener de cada dataset, y *cant_datasets* la cantidad de datasets individuales a unificar. En la línea (2) se inicializa como vacío el dataset resultado $final\_dataset$. Luego, para cada dataset se considera el caso de borde donde la representación esperada sea mayor a la cantidad de imágenes del dataset individual. Si se da el caso de borde entonces a $final\_dataset$ se le agrega el dataset completo, como se puede ver en la línea (5). En caso contrario, se genera una muestra de *sample_k_global* imágenes aleatorias del dataset individual y se le adiciona a $final\_dataset$ en las líneas (7) y (8) respectivamente.

5.3. Modelos

En esta sección se presentan los modelos considerados en la evaluación empírica y cómo fueron implementados, tanto en su arquitectura como en las librerías utilizadas. También se mencionan los hiperparámetros que se decidieron calibrar durante el proceso experimental.

5.3.1. Línea de base

Como línea base se seleccionó e implementó un algoritmo que clasifica cada rectángulo de $k \times j$ píxeles de la imagen como la clase Edificio con una probabilidad p. Sea una imagen de dimensiones $l \times w$, entonces k y j se definen como $k = l \times 0.05$ y $j = w \times 0.05$. Se decidió que las dimensiones de cada segmentación realizada por la línea base sean del 5 % del largo y ancho total porque aparenta ser visualmente razonable si se compara con las imágenes anotadas de los distintos datasets. A su vez, si E es la cantidad de píxeles anotados como clase Edificio en el mapa de verdad, p se calcula como $p = E/(l \times w)$.

Si bien E es a priori desconocido para imágenes no anotadas, se podría llegar a estimar el valor a partir de promediar el valor de E sobre distintos datasets que sí lo estén. En el marco de este proyecto se decidió tomar en cuenta el valor p exacto con el fin de obtener una línea base más exigente con la cual contrastar luego los modelos de redes neuronales implementados.

5.3.2. Redes completamente convolucionales

Como una primer aproximación se utilizó una red completamente convolucional o fully convolutional network (FCN) ya que este tipo de redes en general son superiores, como se mencionó en 4.5, al rendimiento de las CNN con MLP. En particular, se empleó la arquitectura FCN-8. Los pesos iniciales fueron iniciados de forma aleatoria y se utilizó la librería PyTorch para su implementación.

5.3.3. ResNet + FCN

El modelo utilizado cuenta con un encoder ResNet101 y una FCN que recibe su salida, aplica una convolución que reduce a un 25 % los canales, y que finalmente devuelve la predicción según las clases posibles. Los pesos del modelo implementado fueron pre-entrenados con COCO train2017 y la librería utilizada fue Torchvision.

5.3.4. U-Net

Dados los buenos resultados de U-Net para la tarea de segmentación de imágenes (presentadas en la sección 4.6), se implementaron tres variantes de esta arquitectura, donde el elemento de la arquitectura que difiere en cada una de las implementaciones es la estructura del encoder.

5.3.4.1. U-Net simple

Es la arquitectura original planteada en 4.6.2 donde en el encoder se aplican cuatro ciclos de una convolución doble 3×3, ReLU y downsampling. El decoder también se mantiene según el modelo original. Para su implementación se utilizó la librería PyTorch y los pesos se iniciaron de forma aleatoria.

5.3.4.2. U-Net + ResNet

Esta variante de U-Net incorpora en su arquitectura a la de ResNet. Mientras que el decoder se mantiene como el de U-Net simple, el encoder se sustituye por una arquitectura ResNet. Las variantes utilizadas fueron ResNet34, ResNet50, ResNet101 y ResNeXt50. En este caso, se utilizaron pesos preentrenados con ImageNet para cada uno de los encoders. Las librerías utilizadas para estos modelos fueron PyTorch y Segmentation Models PyTorch.

5.3.5. Hiperparámetros

Dos hiperparámetros a calibrar fueron la tasa de aprendizaje y la función de pérdida. Para el primero se utilizaron los valores de 4×10^{-5}, 8×10^{-5} y $1,6 \times 10^{-4}$. Respecto a las funciones de pérdida, se consideraron dice loss y cross entropy + Tversky.

5.3.6. Calibración de modelos

El método de calibración utilizado fue el de escalado de temperatura minimizando la métrica ECE. Para los mismos modelos se minimizó utilizando NLL, pero se observó que los tiempos de calibración eran mayores y los modelos obtenidos no diferían en desempeño de los calibrados con ECE.

Los datos usados para la calibración fueron las imágenes del conjunto de test de los datasets utilizados para el entrenamiento de cada modelo, no recurriendo al de San José de las Matas en ningún momento del proceso.

5.3.7. Votación de modelos

Por último se implementó la votación de modelos, en donde un conjunto de modelos seleccionados da un valor de clasificación para cada píxel y en función de ellos se decide su clase final. Se consideraron conjuntos de votación de tres modelos que en general dieron un buen nivel de desempeño individualmente. Además, se experimentó con dos tipos de votaciones: hard vote y soft vote. En todos los casos el peso de los votos para todos los modelos es el mismo, no existiendo ponderación de unos sobre otros.

5.4. Definición de etapas

Con el fin de obtener la mejor arquitectura e hiperparámetros para el problema planteado, se propone analizar modelos candidatos de forma progresiva y eliminatoria. El proceso de selección está enmarcado en cinco etapas que se dividen de la siguiente manera:

- Etapa 0: punto de partida. Presentación de arquitecturas e hiperparámetros a evaluar.
- Etapa 1: selección de las tres mejores arquitecturas según su desempeño en promedio para cada dataset.

- Etapa 2: redimensionamiento de datasets con imágenes muy grandes.
- Etapa 3: selección de hiperparámetros para modelos entrenados con la unificación de los datasets.
- Etapa 4: calibración de modelos.
- Etapa 5: votación de modelos.

La métrica para valorar un modelo como mejor que otro fue intersection over union (IoU). Además, se aplicaron dos enfoques distintos durante las etapas descritas. Se siguieron ambos enfoques con el fin de comparar su desempeño al final de la etapa 5. Los enfoques son:

- Enfoque A: en cada paso se midió el IoU resultado de cada modelo con el conjunto de test del dataset con que ese modelo entrenó.
- Enfoque B: en cada paso se midió el IoU resultado de cada modelo con el conjunto San José de las Matas. En cada etapa los modelos fueron entrenados con los conjuntos IAD, WHU, MBD y EAII, pero el IoU tenido en cuenta para su selección fue el conseguido al evaluarlos con el de San José de las Matas.

5.5. Bibliotecas de software

La implementación de todos los modelos y funciones auxiliares de preprocesamiento y evaluación se llevaron a cabo en el lenguaje Python versión 3.7

5.5.1. PyTorch

PyTorch [60] es un marco de trabajo de Python basado en Torch [81], un paquete de aprendizaje automático de código abierto basado en el lenguaje Lua [49]. PyTorch es utilizado para las aplicaciones de visión artificial y procesamiento de lenguaje natural. La unidad principal que implementa y soporta todas las operaciones es el tensor, una matriz multidimensional compuesta por tipos de datos homogéneos por unidad. Los tensores son útiles para realizar operaciones de álgebra lineal de manera eficiente, por lo que se utilizan para el cálculo matemático en redes neuronales para un entrenamiento más rápido. Por lo tanto, tiene sentido representar sus datos de entrada en forma de tensores. PyTorch a su vez permite que las operaciones sobre tensores puedan

hacerse sobre la unidad de procesamiento gráfica (GPU), lo que ayuda a acelerar los cálculos numéricos, que pueden aumentar la velocidad de las redes neuronales en 50 veces o más.

Algunas de las principales funcionalidades de PyTorch son:

- TorchScript: es el entorno de producción de PyTorch que permite a los usuarios realizar una transición sin problemas en cualquier parte del código entre los modos (CPU/GPU). TorchScript optimiza la funcionalidad, la velocidad, la facilidad de uso y la flexibilidad.
- Cálculo de gráficos dinámicos: esta función permite a los usuarios cambiar el comportamiento de la red (arquitectura, orden de precedencia) sobre la marcha, en lugar de esperar a que se ejecute todo el código.
- Diferenciación automática: esta técnica calcula numéricamente la derivada de una función haciendo pases hacia atrás en las redes neuronales.
- Compatibilidad con Python: debido a que PyTorch se basa en Python, se puede usar con bibliotecas y paquetes populares como NumPy [56], SciPy [68], Pandas [59], entre otros.

También se hizo uso de Torchvision [82], librería de visión artificial de PyTorch, la cual contiene conjuntos de datos, arquitecturas de modelos con pesos pre-entrenados y operaciones de transformación relevantes que se utilizan a menudo en el área de procesamiento de imágenes. Es a menudo utilizada como API para importar elementos populares en el área y no tener que implementarlos nuevamente.

5.5.2. Segmentation Models PyTorch

Segmentation Models PyTorch [69] cumple una función similar a Torchvision, con el agregado que las arquitecturas y modelos disponibilizados por la API tienen un fuerte enfoque en la tarea de segmentación de imágenes. Implementa además modelos que no están contemplados en Torchvision y, a diferencia de este último, Segmentation Models PyTorch es un proyecto independiente a PyTorch.

5.5.3. OpenCV

OpenCV (Open Source Computer Vision Library) [58] es una biblioteca multi-plataforma de código abierto de visión artificial y aprendizaje automáti-

co. La librería suma más de 2500 algoritmos optimizados en esos campos, clásicos y del estado del arte, y juega un papel importante en aplicaciones de tiempo real. Se centra principalmente en el procesamiento de imágenes, la captura y el análisis de video. Algunas aplicaciones de ejemplo son: detectar y reconocer rostros, identificar objetos, clasificar acciones humanas en videos, rastrear los movimientos de la cámara, rastrear objetos en movimiento y extraer modelos 3D de objetos.

Si bien está desarrollada en C++, OpenCV cuenta con interfaces para otros lenguajes como Python, MATLAB, Java, Javascript y Octave. Adicionalmente, OpenCV soporta los sistemas operativos Windows, Linux, Mac OS y Android.

5.5.4. Albumentations

Albumentation [8] es una biblioteca de Python para la implementación de data augmentation. El paquete está escrito en NumPy, OpenCV e imgaug [32]. Lo que hace que Albumentation se destaque es la cantidad de técnicas de aumento de datos que están disponibles. Si bien la mayoría de las bibliotecas del estilo incluyen técnicas como recortar, voltear, rotar y escalar, Albumentation proporciona una gama de técnicas de aumento de imágenes más extensivas como el contraste, el desenfoque y la reproducción aleatoria de canales. Algunas características positivas de esta librería son:

- Rendimiento: ofrece uno de los mejores rendimientos en la mayoría de las transformaciones de uso común. Para hacerlo combina varias bibliotecas de manipulación de imágenes de bajo nivel y selecciona la implementación más rápida. Una prueba de esto puede verse en la tabla 5.1.
- Variedad: no solo contiene las técnicas comunes de manipulación de imágenes, sino una amplia variedad de transformaciones de sobre ellas, lo cual es útil para las tareas y las aplicaciones específicas según el dominio.
- Integración: funciona con marcos de aprendizaje profundo populares como PyTorch o TensorFlow [80]. Además pertenece al ecosistema PyTorch.

5.5.5. Jupyter Notebook

Según la definición del sitio oficial de Jupyter [36], los notebooks son documentos producidos por la aplicación Jupyter Notebook, que contienen códi-

Transformación	*albumentations 1.0.0*	*imgaug 0.4.0*	*torchvision*
Volteo Horizontal	**6244**	2281	2208
Volteo Vertical	**5443**	2265	1910
Rotación	**340**	261	146
Mover+Escalar+Rotar	**566**	372	129
Brillo	**2333**	1065	370
Contraste	**2352**	1098	307
Brillo + Contraste	**2331**	613	170
Inversión RGB	**2320**	1059	-
Recorte Aleatorio	**149449**	2750	32824

Tabla 5.1: Resultados de la ejecución de la prueba comparativa en las primeras 2000 imágenes del conjunto de validación de ImageNet utilizando un procesador Intel Xeon E5-2650 v4. Todas las salidas se convierten en una matriz NumPy contigua con el tipo de datos NumPy.uint8. La tabla muestra cuántas imágenes por segundo se pueden procesar en un solo núcleo.

go de programación (por ejemplo Python) y elementos de texto enriquecido (párrafo, ecuaciones, figuras, enlaces). Los notebooks son tanto documentos legibles que contienen distintos formatos de información como cifras, gráficas y tablas, así como documentos ejecutables que se pueden correr para realizar análisis de datos.

Jupyter Notebook es una aplicación servidor-cliente que permite editar y ejecutar documentos de notebook a través de un navegador web. Se puede ejecutar en un escritorio local que no requiera acceso a Internet o se puede instalar en un servidor remoto y acceder a través de Internet.

5.5.6. Google Colab

Colaboratory, también llamado Colab, es un producto de Google Research [21]. Colab permite que todos puedan escribir y ejecutar código arbitrario de Python en el navegador. Es ideal para aplicarlo en proyectos de aprendizaje automático, análisis de datos y educación. Más técnicamente, Colab es un servicio de notebook alojado de Jupyter que no requiere configuración para usarlo y brinda acceso gratuito a recursos computacionales, incluidas GPU. En este proyecto se utilizaron dos suscripciones a su servicio pago Colab Pro, el cual permite tener acceso a GPUs más rápidas, acceso a más memoria y

tiempos de ejecución más largos. Particularmente, la suscripción permite tener un máximo de dos entornos ejecutando con GPU al mismo tiempo. Es por esto que dada la cantidad de modelos a entrenar y el tiempo de entrenamiento de cada uno se optó por contratar dos membresías del servicio.

5.5.7. Google Drive

Google Drive [22] es un servicio de almacenamiento en la nube donde típicamente se guardan archivos de distinto tipo. En el contexto de este proyecto se utilizó para guardar los notebooks de Colab y todos los datasets con los que se trabajó, además de resultados preliminares y los propios modelos.

5.5.8. Labelbox

Labelbox [40] es una plataforma centrada en datos para tareas de inteligencia artificial. Permite al usuario crear datos de entrenamiento con su herramienta de anotación y además ofrece otras características como diagnosticar el rendimiento de un modelo y mejorarlo, todo dentro del mismo ambiente.

En el marco de este proyecto la plataforma Labelbox se utilizó para anotar manualmente el dataset de San José de las Matas y luego importarlo a través de la API que se disponibiliza con este fin.

Capítulo 6

Evaluación experimental

En el presente capítulo se exponen los resultados obtenidos y su análisis.

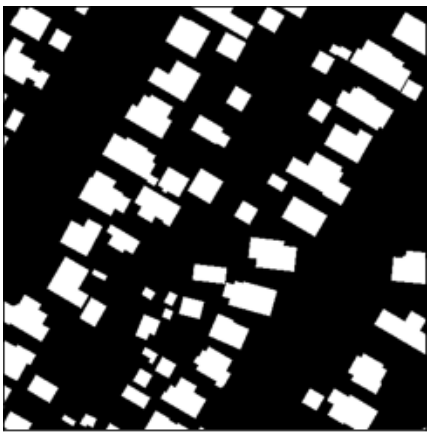
6.1. Entorno de ejecución

La evaluación experimental se realizó sobre entornos de Google Colab [21] con una CPU Intel(R) Xeon(R) CPU @ 2.20 GHz, una GPU NVIDIA Tesla P100-PCIE 16 GB, 25 GB de memoria RAM y sistema operativo Ubuntu 18.04.5 LTS.

6.2. Línea de base

Dada la manera en que se calcula p, el clasificador ya cuenta con información acerca de la imagen que va a segmentar. Este hecho genera que para casos donde no exista ningún píxel clasificado como Edificio, la línea de base tenga un IoU de 1, dado que $p = 0$ para cualquier rectángulo de toda la imagen. También tiene como consecuencia que, para imágenes con mucha presencia de la clase Edificio, aumente la probabilidad de que la clasificación pseudo-aleatoria coincida con el mapa de verdad en más zonas. En el caso donde la presencia de la clase Edificio es baja, sin embargo, ocurre lo opuesto: el clasificador de línea de base tiende a no acertar los pocos rectángulos de clase Edificio que construye. En la figura 6.1 se puede ver un ejemplo de cada uno de estos casos.

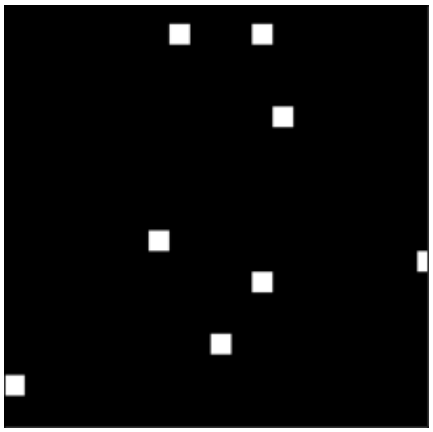
Se utilizó el clasificador para los datasets originales, obteniendo como resultado un valor de IoU de 0.7647.



(a) Mapa de clases con alta presencia de clase Edificio

(b) Predicción de imagen con alta presencia de clase Edificio

(c) Mapa de clases con baja presencia de clase Edificio



(d) Predicción de imagen con baja presencia de clase Edificio

Figura 6.1: Comparación del desempeño del clasificador de línea de base para imágenes con alta ((a) y (b)) y baja ((c) y (d)) presencia de clase Edificio.

6.3. Etapa 0 - Presentación de arquitecturas e hiperparámetros a evaluar

En la sección 5.3 se presentaron un conjunto de modelos e hiperparámetros posibles a aplicar al problema de segmentación de imágenes. En particular, se consideraron siete arquitecturas: FCN-8, FCN con encoder ResNet50, FCN con encoder ResNet101, U-Net con encoder ResNet34, U-Net con encoder ResNet50, U-Net con encoder ResNet101 y U-Net con encoder ResNeXt50. Al mismo tiempo, se consideraron tres posibles valores para la tasa de aprendizaje (4×10^{-5}, 8×10^{-5} y $1{,}6 \times 10^{-4}$) y dos posibles funciones de pérdida (dice loss y cross entropy + Tversky), lo que da un resultado de 42 modelos a evaluar.

6.4. Etapa 1 - Selección de arquitectura y datasets

Esta sección describe el mecanismo de evaluación para la etapa 1 y sus resultados.

6.4.1. Mecanismo de evaluación

Dado que se parte con una cantidad considerable de modelos, y con el fin de simplificar su selección, en la etapa 1 se entrenaron y evaluaron todas las arquitecturas posibles para cada dataset por separado, manteniendo la tasa de aprendizaje y función de pérdida fijos. Luego, de los modelos entrenados se seleccionaron las tres arquitecturas que mejor puntuaron en promedio para los datasets según la métrica IoU para cada enfoque.

El valor fijo seleccionado para la presente etapa para la tasa de aprendizaje fue 8×10^{-5}, mientras que la función de pérdida utilizada fue dice loss.

6.4.2. Resultados

En la tabla 6.1 se reportan los resultados para los enfoques A y B definidos en la sección 5.4. Tanto para el enfoque A como para el B se obtuvo que dos de los tres mejores desempeños fueron alcanzados por modelos con arquitectura U-Net y el restante por la arquitectura FCN.

Enfoque A					
Arquitectura	*Dataset*				
	IAD	*MBD*	*WHU*	*EAII*	*Promedio*
FCN-8	0,8386	0,7803	0,9572	0,9450	0,8803
FCN+ResNet50	0,8965	0,8172	0,9700	0,9629	**0,9117**
FCN+ResNet101	0,8848	0,8130	0,9685	0,9624	0,9072
U-Net+ResNet34	0,8715	0,8267	0,9692	0,9619	0,9073
U-Net+ResNet50	0,8780	0,8097	0,9693	0,9615	0,9046
U-Net+ResNet101	0,9177	0,8279	0,9690	0,9597	**0,9186**
U-Net+ResNeXt50	0,8965	0,8147	0,9720	0,9612	**0,9111**
Enfoque B					
Arquitectura	*Dataset*				
	IAD	*MBD*	*WHU*	*EAII*	*Promedio*
FCN-8	0,8342	0,8293	0,8161	0,8709	0,8376
FCN+ResNet50	0,8706	0,8128	0,8445	0,8655	0,8484
FCN+ResNet101	0,8704	0,8398	0,8297	0,8891	**0,8573**
U-Net+ResNet34	0,8738	0,8185	0,8390	0,8846	**0,8540**
U-Net+ResNet50	0,7727	0,8255	0,8553	0,8865	0,8350
U-Net+ResNet101	0,8781	0,8188	0,8270	0,8869	**0,8527**
U-Net+ResNeXt50	0,8805	0,8281	0,8212	0,8733	0,8508

Tabla 6.1: Valores de IoU obtenidos para cada arquitectura para los enfoques A y B de la etapa 1.

6.5. Etapa 2 - Redimensionamiento de datasets con imágenes de grandes dimensiones

Esta sección describe el mecanismo de evaluación para la etapa 2 y sus resultados.

6.5.1. Mecanismo de evaluación

Los datasets recolectados presentan entre sí diferencias significativas en cuanto a la cantidad de imágenes y su calidad. Por un lado, los datasets WHU y EAII cuentan con una gran cantidad de imágenes de poco tamaño, mientras que IAD y MBD poseen una cantidad mucho menor, pero mucho más gran-

des en tamaño. El hecho que las imágenes de IAD y MBD sean de grandes dimensiones tiene como consecuencia que los recortes que se generan en cada instancia de entrenamiento sean muy pequeños (480×480 píxeles) en comparación con el tamaño original de la imagen, por lo que se estaría ignorando mucha información. Aunque la pérdida de información puede mitigarse aumentando la cantidad de épocas de entrenamiento, el plan para las etapas posteriores es el de unificar los datasets, por lo que un enfoque presumiblemente mejor es el de llevar los datasets a escalas más pequeñas, similares a las de los datasets WHU y EAII.

Con el fin de abordar el problema de la pérdida de información para imágenes de grandes dimensiones, se generaron los datasets auxiliares IAD2500, IAD1250, IAD1000, MBD750 y MBD500 ya introducidos en la sección 5.2.2. El objetivo fue mantener un dataset auxiliar por cada dataset original. Al igual que para la selección de arquitecturas, el que se mantuvo fue aquel que puntuó mejor en promedio la métrica IoU para los modelos obtenidos en la etapa 1.

6.5.2. Resultados

A priori no se reconoce un patrón de mejor desempeño que esté relacionado con el tamaño del recorte de imagen. Para el caso de IAD, como se puede ver en la tabla 6.2, hubo una tendencia a lograr un mejor desempeño para recortes más grandes para ambos enfoques, con la única excepción de IAD1000 para el enfoque B. Para ese caso en particular, se tomó la decisión excepcional de seleccionar al segundo mejor dataset según IoU porque si bien el IoU de IAD es ligeramente mayor, se asumió que la pérdida de información de recortar imágenes de dimensiones tan grandes como las de IAD (5000×5000 px) durante las transformaciones online durante el entrenamiento (480×480 px) tendría un impacto negativo en etapas posteriores, donde la cantidad de épocas fue menor (10 épocas) a la que se utilizó para la etapa 2 con IAD (40 épocas). Además, otro hecho que permite tomar esta decisión es que la diferencia entre ambos IoUs es menor a 0,01.

En otro orden, en MBD se tiene que el mejor desempeño se alcanzó en MBD500 para el enfoque A y en el original para el enfoque B, tal como se presenta en la tabla 6.3. Los resultados podrían interpretarse como análogos al caso anteriormente descrito para IAD, pero se decidió mantener MBD y no tomar el segundo mejor (MBD500) por dos razones: primero, porque la dife-

rencia entre ambos ($\approx$ 0,02) fue considerada como no despreciable. Segundo, la información perdida es menor que para el caso de IAD, ya que la dimensión de las imágenes de MBD es 1500×1500 px.

Enfoque A				
Dataset	*Arquitectura*			
	FCN+ ResNet50	*U-Net+ ResNet101*	*U-Net+ ResNeXt50*	*Promedio*
IAD1000	0,8469	0,8802	0,8694	0,8694
IAD1250	0,8666	0,8611	0,8771	0,8683
IAD1666	0,8782	0,8534	0,8816	0,8711
IAD2500	0,9181	0,9057	0,9102	**0,9113**
IAD	0,8965	0,9177	0,8965	0,9036
Enfoque B				
Dataset	*Arquitectura*			
	FCN+ ResNet50	*U-Net+ ResNet101*	*U-Net+ ResNeXt50*	*Promedio*
IAD1000	0,8782	0,8441	0,8757	**0,8660***
IAD1250	0,8071	0,8374	0,7861	0,8102
IAD1666	0,8624	0,7898	0,8707	0,8410
IAD2500	0,8571	0,8207	0,8443	0,8407
IAD	0,8704	0,8738	0,8781	**0,8741***

Tabla 6.2: Valores de IoU obtenidos para cada dataset auxiliar de IAD para los enfoques A y B de la etapa 2. (*) En este caso particular se optó por seleccionar I1000 por más que IAD obtuvo en promedio un mejor IOU. Los principales motivos son que la diferencia entre ambos es muy poca ($< 0{,}01$) y, a pesar de ese resultado preliminar, se estimó que la pérdida de información de recortar imágenes tan grandes (5000×5000) durante las transformaciones online durante el entrenamiento tendría un impacto negativo en etapas posteriores.

Enfoque A				
Arquitectura	*Dataset*			
	FCN+ ResNet50	*U-Net+ ResNet101*	*U-Net+ ResNeXt50*	*Promedio*
MBD500	0,8185	0,8232	0,8274	**0,8230**
MBD750	0,8100	0,8145	0,8235	0,8160
MBD	0,8172	0,8279	0,8147	0,8199
Enfoque B				
Arquitectura	*Dataset*			
	FCN+ ResNet50	*U-Net+ ResNet101*	*U-Net+ ResNeXt50*	*Promedio*
MBD500	0,8116	0,8056	0,8051	0,8074
MBD750	0,8272	0,7354	0,7877	0,7834
MBD	0,8398	0,8185	0,8270	**0,8284**

Tabla 6.3: Valores de IoU obtenidos para cada dataset auxiliar de MBD para los enfoques A y B.

6.6. Etapa 3 - Selección de hiperparámetros según desempeño para el dataset de San José de las Matas

Esta sección describe el mecanismo de evaluación para la etapa 3 y sus resultados.

6.6.1. Mecanismo de evaluación

Como resultado de la etapa 2 se obtuvieron tres arquitecturas candidatas y dos datasets que probaron ser una mejor partición respecto a sus competidores según la métrica de desempeño IoU.

En la etapa 3 se buscó calibrar los hiperparámetros que se mantuvieron como fijos en las anteriores: la función de costo y la tasa de aprendizaje. Además, en la etapa 3 los modelos fueron entrenados con la unión de los datasets resultado de la etapa 2 con dos variantes. Una de las variantes fue la de entrenar los modelos con el 100 % de las imágenes del dataset y la otra solo con el 50 %.

Funciones de costo. Se evaluaron las funciones de dice loss y una combinación de cross entropy y focal Tversky, definida según la Ec. 6.1.

$$EFTL(c) = CE(c) + \frac{FTL(c)}{2} \tag{6.1}$$

Los valores de α y β para FTL fueron 0.99 y 0.01 respectivamente. Se eligieron estos valores para minimizar los falsos negativos. En lo que respecta a γ, se eligieron los valores $\gamma = 0.25$ y $\gamma = 0.1$. La elección de $\gamma < 1$ se hizo porque dado el contexto de la tarea, la red neuronal se aplicaría sobre zonas donde hay población (y por lo tanto edificios), de modo que se espera que la red logre aprender los ejemplos "fáciles", según se explica en la sección 2.5.4.4.

Tasa de aprendizaje. Se entrenaron los modelos con tres posibles valores para la tasa de aprendizaje: 4×10^{-5}, 8×10^{-5} y $1{,}6 \times 10^{-4}$.

Porcentaje de datasets. Esta etapa fue la primera donde los modelos fueron entrenados utilizando la unificación de los datasets y no con cada dataset por separado, como sucedió en las etapas anteriores. Se asumió que usar la unión de los datasets para el entrenamiento podría aumentar la capacidad de generalización de los modelos y tener un mejor desempeño sobre el total. Sin embargo, la unificación de los datasets (IAD1000, WHU, MBD, EAII) sumó una gran cantidad de imágenes de entrenamiento (13.301), lo que demandó un mayor tiempo para entrenar los modelos. Con el propósito de validar si los modelos son capaces de alcanzar buenos desempeños con menos datos (y por lo tanto con menos tiempo de entrenamiento), se planteó entrenarlos no solo con el 100 % de las imágenes de los datasets, sino también generar instancias de entrenamiento con el 50 % del total los datos y comparar sus resultados. El mecanismo de selección de las imágenes para el armado del conjunto con el 50 % de las imágenes del total se explica en la sección 5.2.3. Finalmente, los hiperparámetros empleados para entrenar los modelos candidatos se presentan en la tabla 6.4.

Hiperparámetro	*Valor*	*Identificador*
Función de costo	dice loss	DL
Función de costo	$CEFTL, \gamma = 0.25$	$CEFTL_1$
Función de costo	$CEFTL, \gamma = 0.10$	$CEFTL_2$
Tasa de aprendizaje	4×10^{-5}	-
Tasa de aprendizaje	8×10^{-5}	-
Tasa de aprendizaje	$1{,}6 \times 10^{-4}$	-
Porcentaje de dataset	100 %	All_100
Porcentaje de dataset	50 %	All_50

Tabla 6.4: Hiperparámetros a valorar en la etapa 3.

6.6.2. Resultados - Enfoque A

Esta sección presenta los resultados de IoU para los modelos generados a partir de todas las combinaciones de hiperparámetros posibles, comenzando por los valores obtenidos de IoU para el enfoque A, cuyos modelos entrenaron con los datasets IAD2500, MBD500, WHU y EAII.

6.6.2.1. Introducción

En la tabla 6.5 se reportan los puntajes de los modelos que son producto de cada combinación de hiperparámetros para el 100 % del conjunto de imágenes de los datasets. La identificación de cada modelo contiene un "100" en su nombre en representación del porcentaje del dataset utilizado para el entrenamiento. En la columna "IoU" se muestra el valor de la métrica homónima para el conjunto de test de la unión de IAD2500, MBD500, WHU y EAII. La columna "IoU SJM", por otro lado, refiere al desempeño que tuvo el modelo para el dataset de San José de las Matas a modo informativo, ya que ese valor no es considerado en ninguna decisión para el enfoque A. En la tabla 6.6 se exponen los rendimientos de los modelos al ser entrenados con los mismos hiperparámetros pero con el 50 % de la unión de los datasets.

Si bien todos los rendimientos de la etapa 3 para el enfoque A están unificados en las tablas 6.5 y 6.6, en la presente sección se analizan los resultados segmentándolos por cada tipo de hiperparámetro tenido en cuenta: función de costo, tasa de aprendizaje, arquitectura y proporción de dataset. En cada caso se intenta encontrar una relación, si la hay, entre el desempeño y las

variables utilizadas. Para una mejor lectura e interpretación de los datos se provee la figura 6.2, que ofrece una presentación gráfica que facilita identificar las diferencias de desempeño para cada hiperparámetro. Finalmente, se hace un análisis del rendimiento de los modelos del enfoque A con el dataset SJM y se muestra un ejemplo de predicción del modelo All_50_16 sobre una imagen de San José de las Matas en la figura 6.5.

Id. modelo	*Arquitectura*	*F. de costo*	*Tasa*	*IoU*	*IoU SJM*
All_100_11	U-Net+ResNet101	DL	4×10^{-5}	0,9644	0,8690
All_100_12	U-Net+ResNet101	DL	8×10^{-5}	0,9645	0,8903
All_100_13	U-Net+ResNet101	DL	$1{,}5 \times 10^{-4}$	0,9611	0,8709
All_100_14	U-Net+ResNet101	$CEFTL_1$	4×10^{-5}	0,9657	0,8871
All_100_15	U-Net+ResNet101	$CEFTL_1$	8×10^{-5}	0,9639	0,8681
All_100_16	U-Net+ResNet101	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,9641	0,8909
All_100_17	U-Net+ResNet101	$CEFTL_2$	4×10^{-5}	0,9655	0,8954
All_100_18	U-Net+ResNet101	$CEFTL_2$	8×10^{-5}	0,9651	0,8709
All_100_19	U-Net+ResNet101	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9604	0,8785
All_100_21	U-Net+ResNeXt50	DL	4×10^{-5}	0,9648	0,8712
All_100_22	U-Net+ResNeXt50	DL	8×10^{-5}	0,9658	0,8809
All_100_23	U-Net+ResNeXt50	DL	$1{,}5 \times 10^{-4}$	0,9651	0,8818
All_100_24	U-Net+ResNeXt50	$CEFTL_1$	4×10^{-5}	**0,9665**	0,8534
All_100_25	U-Net+ResNeXt50	$CEFTL_1$	8×10^{-5}	0,9661	0,8979
All_100_26	U-Net+ResNeXt50	$CEFTL_1$	$1,5 \times 10^{-4}$	0,9657	0,8524
All_100_27	U-Net+ResNeXt50	$CEFTL_2$	4×10^{-5}	0,9654	0,8837
All_100_28	U-Net+ResNeXt50	$CEFTL_2$	8×10^{-5}	0,9658	0,8828
All_100_29	U-Net+ResNeXt50	$CEFTL_2$	$1,5 \times 10^{-4}$	0,9658	0,8999
All_100_31	FCN+ResNet50	DL	4×10^{-5}	0,9636	0,8876
All_100_32	FCN+ResNet50	DL	8×10^{-5}	0,9626	0,8649
All_100_33	FCN+ResNet50	DL	$1{,}5 \times 10^{-4}$	0,9626	0,8785
All_100_35	FCN+ResNet50	$CEFTL_1$	8×10^{-5}	0,9629	0,8565
All_100_36	FCN+ResNet50	$CEFTL_1$	$1,5 \times 10^{-4}$	0,9598	0,8466
All_100_37	FCN+ResNet50	$CEFTL_2$	4×10^{-5}	0,9619	0,8648
All_100_38	FCN+ResNet50	$CEFTL_2$	8×10^{-5}	0,9616	0,8833
All_100_39	FCN+ResNet50	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9628	0,8854

Tabla 6.5: Resultados de IoU para cada combinación de hiperparámetros según el enfoque A para el 100 % de imágenes del conjunto de datos.

6.6.2.2. Desempeño por función de costo

La tres funciones de costo tuvieron valores parecidos entre sí. Mientras que el IoU promedio para DL fue 0,9629, para $CEFTL_1$ y $CEFTL_2$ se obtuvieron los valores 0,9628 y 0,9619 respectivamente. Si bien por una mínima diferencia

Id. modelo	*Arquitectura*	*F. de costo*	*Tasa*	*IoU*	*IoU SJM*
All_50_11	U-Net+ResNet101	DL	4×10^{-5}	0,9630	0,8613
All_50_12	U-Net+ResNet101	DL	8×10^{-5}	0,9640	0,8790
All_50_13	U-Net+ResNet101	DL	$1{,}5 \times 10^{-4}$	0,9601	0,8865
All_50_14	U-Net+ResNet101	$CEFTL_1$	4×10^{-5}	0,9616	0,8643
All_50_15	U-Net+ResNet101	$CEFTL_1$	8×10^{-5}	0,9599	0,8924
All_50_16	U-Net+ResNet101	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,9580	**0,9071**
All_50_17	U-Net+ResNet101	$CEFTL_2$	4×10^{-5}	0,9609	0,8952
All_50_18	U-Net+ResNet101	$CEFTL_2$	8×10^{-5}	0,9611	0,9025
All_50_19	U-Net+ResNet101	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9586	0,8928
All_50_21	U-Net+ResNeXt50	DL	4×10^{-5}	0,9607	0,8854
All_50_22	U-Net+ResNeXt50	DL	8×10^{-5}	0,9650	0,8787
All_50_23	U-Net+ResNeXt50	DL	$1{,}5 \times 10^{-4}$	0,9624	0,8978
All_50_24	U-Net+ResNeXt50	$CEFTL_1$	4×10^{-5}	0,9639	0,8859
All_50_25	U-Net+ResNeXt50	$CEFTL_1$	8×10^{-5}	0,9649	0,8735
All_50_26	U-Net+ResNeXt50	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,9651	0,8934
All_50_27	U-Net+ResNeXt50	$CEFTL_2$	4×10^{-5}	0,9637	0,8845
All_50_28	U-Net+ResNeXt50	$CEFTL_2$	8×10^{-5}	0,9644	0,8904
All_50_29	U-Net+ResNeXt50	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9632	0,8694
All_50_31	FCN+ResNet50	DL	4×10^{-5}	0,9620	0,8639
All_50_32	FCN+ResNet50	DL	8×10^{-5}	0,9626	0,8888
All_50_33	FCN+ResNet50	DL	$1{,}5 \times 10^{-4}$	0,9586	0,8772
All_50_34	FCN+ResNet50	$CEFTL_1$	4×10^{-5}	0,9611	0,8803
All_50_35	FCN+ResNet50	$CEFTL_1$	8×10^{-5}	0,9593	0,8573
All_50_36	FCN+ResNet50	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,9609	0,8752
All_50_37	FCN+ResNet50	$CEFTL_2$	4×10^{-5}	0,9506	0,8761
All_50_38	FCN+ResNet50	$CEFTL_2$	8×10^{-5}	0,9601	0,8765
All_50_39	FCN+ResNet50	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9581	0,8937

Tabla 6.6: Resultados de IoU para cada combinación de hiperparámetros según el enfoque A para el 50 % de imágenes del conjunto de datos.

DL obtuvo las mejores métricas en promedio, el mejor IoU se obtuvo con $CEFTL_1$ (modelo All_100_24). A partir de la comparación de los IoU promedio para DL, $CEFTL_1$ y $CEFTL_2$ se concluye que para el enfoque A no existen grandes variaciones en el desempeño para las funciones de costo consideradas.

6.6.2.3. Desempeño por tasa de aprendizaje

Aunque la tasa con mejor desempeño en promedio fue 8×10^{-5} con IoU promedio de 0,9633, el valor máximo de IoU entre todos los modelos fue 0,9665 y se alcanzó con la tasa 4×10^{-5}. Para el caso de $1,5 \times 10^{-4}$ se obtuvo el peor IoU en promedio (0,9618) para el hiperparámetro, lo que nos indica que la función de costo converge para valores menores a $1{,}5 \times 10^{-4}$ y seguramente se dio el efecto de "bandeo" para ese valor.

6.6.2.4. Desempeño por arquitectura

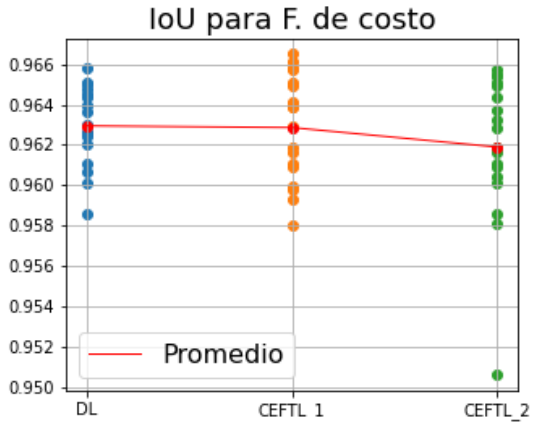
La arquitectura es el primer hiperparámetro de los evaluados hasta ahora donde hay una mayor diferencia en promedio entre los valores considerados. U-Net+ResNeXt50 promedió un puntaje de IoU de 0,9646, mientras que U-Net+ResNet50 y FCN+ResNet50 promediaron 0,9623 y 0,9607 correspondientemente. A partir de los datos se nota un mejor desempeño de la arquitectura U-Net por sobre FCN, la cual yace en el último lugar de todos los valores de hiperparámetros estudiados, como se puede ver en la figura 6.2c.

6.6.2.5. Desempeño por proporción de dataset

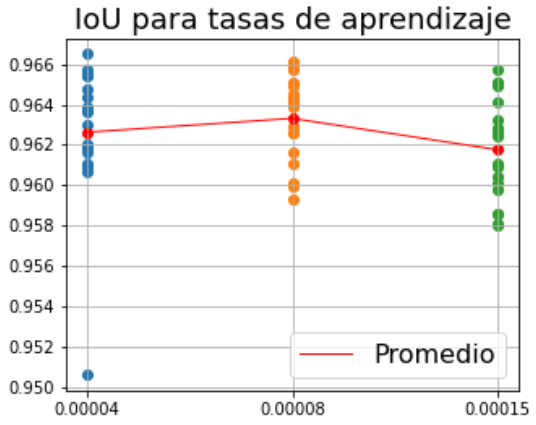
En lo que respecta a la proporción del dataset, se cumplió que el conjunto que contenía la totalidad de datos de entrenamiento disponibles presentó un mejor desempeño frente al que solo contaba con la mitad de ellos. Concretamente, los modelos entrenados con el 100 % de los datos obtuvo un IoU promedio de 0,9639 en tanto que los modelos entrenados con el 50 % de los datos promediaron un puntaje de 0,9613. Aún cuando obtener un mejor desempeño de los modelos que entrenaron con más datos era un comportamiento esperado, no necesariamente significaba que los modelos obtenidos fueran mejores, principalmente porque luego serían evaluados con datos con características disímiles con los que entrenaron. La causa de obtener un mejor desempeño cuando con más datos se cuenta en general está dada por la existencia de un sobreajuste sobre el conjunto de entrenamiento, lo cual es una situación deseablemente evitable.

Además del valor obtenido para IoU, se valoró como importante estudiar los tiempos de entrenamiento de los modelos que usaban el 50 % de los datos frente a los que tomaban el 100 %. Como consecuencia se obtuvo que en promedio

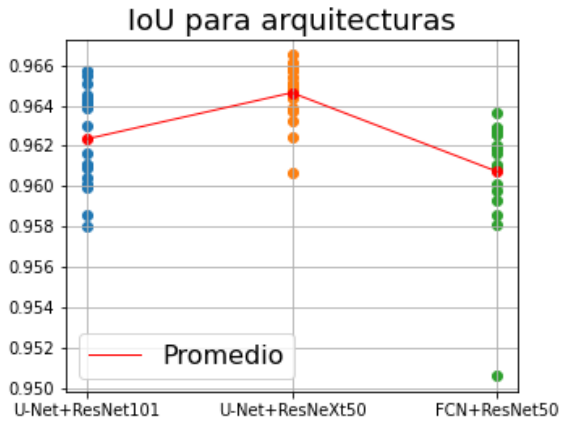
los modelos que hacen uso de la totalidad de los datos demoran un 66 % de tiempo más en comparación a los que utilizan la mitad de los datos.



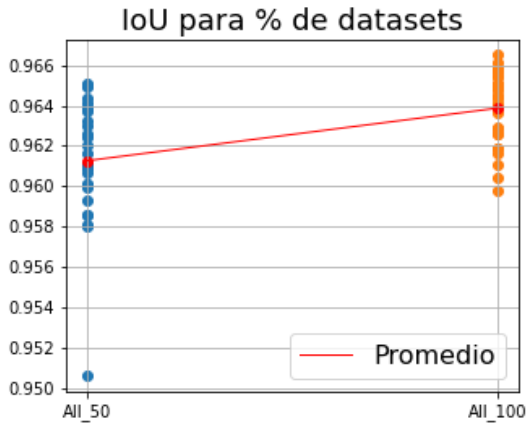
(a) IoU segmentado por función de costo



(b) IoU segmentado por tasa de aprendizaje



(c) IoU segmentado por arquitectura



(d) IoU segmentado por proporción de dataset

Figura 6.2: Gráficas de valores de IoU obtenidos para cada modelo según el enfoque A. Cada gráfica está segmentada por hiperparámetro considerado.

6.6.2.6. Desempeño en dataset de entrenamiento vs. San José de las Matas

Esta sección analiza el comportamiento de los modelos entrenados en la etapa 3 para el enfoque A a la hora de evaluarlos frente a al dataset de San José de las Matas.

En la figura 6.3 se muestra una comparativa de los valores de IoU logrados por los modelos para su conjunto de entrenamiento y luego el obtenido para San José de las Matas. Se decidió dividirlos entre los modelos que entrenaron con el

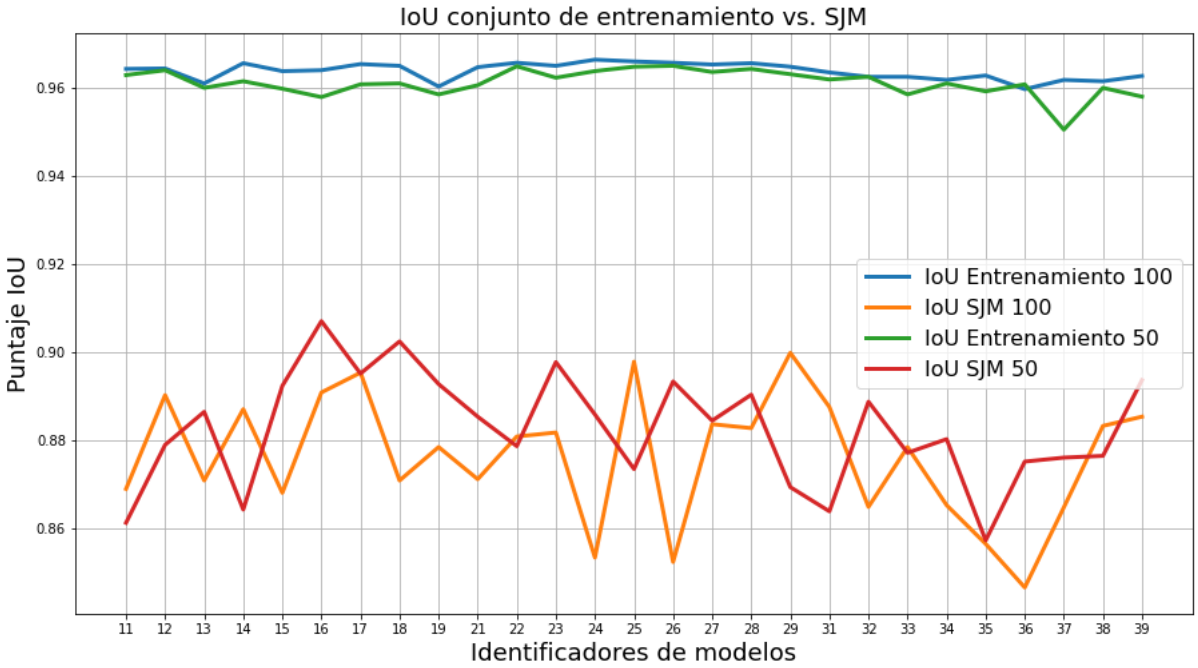


Figura 6.3: Comparación de valores de IoU obtenidos para los modelos del enfoque A para su conjunto de entrenamiento y el dataset de San José de las Matas.

50 % y 100 % del dataset respectivamente. A partir de la figura se pueden hacer dos observaciones importantes: la primera es que no existe un patrón claro que relacione el comportamiento del modelo para su dataset de entrenamiento con el de San José de las Matas. Para modelos donde el IoU de entrenamiento es bajo en comparación al resto, no se cumple lo mismo para el valor obtenido en SJM. Un ejemplo claro es All_50_16, donde el IoU de entrenamiento es de los más bajos, pero el IoU obtenido para SJM es de los más altos.

La segunda observación que se hace es que en general los modelos entrenados con el 50 % del dataset obtuvieron mejores IoUs en SJM, lo que sustenta lo mencionado en la sección 6.6.2.5, donde se estimaba que los modelos entrenados con la totalidad del dataset probablemente padecerían de sobreajuste. Los valores exactos de IoU para SJM por porcentaje de dataset se muestran en la figura 6.4.

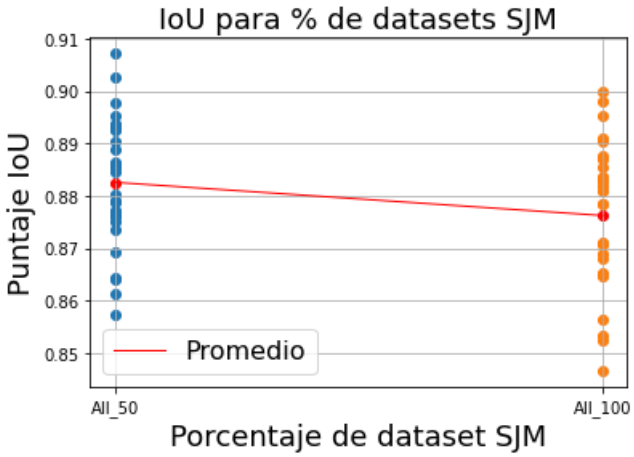


Figura 6.4: Desempeño de modelos de enfoque A sobre San José de las Matas divididos entre los entrenados con el 50 % y 100 % del dataset respectivamente.

(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.5: Predicción del modelo All_50_16 sobre la imagen "image1920_3200.jpg" del conjunto San José de las Matas.

6.6.3. Resultados - Enfoque B

Esta sección presenta los resultados de la etapa 3 para el enfoque B, cuyos modelos fueron entrenados con los datasets IAD1000, MBD, WHU y EAII.

6.6.3.1. Introducción

En la tabla 6.7 se encuentran los desempeños de cada modelo para el caso de la consideración del 100 % del dataset y en la tabla 6.8 para el caso de 50 %. La diferencia con las tablas presentadas en el enfoque A es que las del enfoque B no cuentan con la columna "IoU", dado que el valor de la métrica en la cual

se basan todas las decisiones en el enfoque B es del obtenido para San José de las Matas, manifiesto en la columna "IoU SJM".

Análogamente al enfoque A, se muestran los resultados para cada hiperparámetro considerado: función de costo, tasa de aprendizaje, arquitectura y proporción de dataset. También se provee una figura, la 6.6, para una mejor lectura de los datos resultantes. Por último se presenta la figura 6.7, donde se muestra un ejemplo de predicción del modelo All2_50_17 sobre una imagen de San José de las Matas.

Id. modelo	*Arquitectura*	*F. de costo*	*Tasa*	*IoU SJM*
All2_100_11	U-Net+ResNet101	DL	4×10^{-5}	0,8919
All2_100_12	U-Net+ResNet101	DL	8×10^{-5}	0,8989
All2_100_13	U-Net+ResNet101	DL	$1{,}5 \times 10^{-4}$	0,8826
All2_100_14	U-Net+ResNet101	$CEFTL_1$	4×10^{-5}	0,9014
All2_100_15	U-Net+ResNet101	$CEFTL_1$	8×10^{-5}	0,9074
All2_100_16	U-Net+ResNet101	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,8938
All2_100_17	U-Net+ResNet101	$CEFTL_2$	4×10^{-5}	0,8906
All2_100_18	U-Net+ResNet101	$CEFTL_2$	8×10^{-5}	0,8965
All2_100_19	U-Net+ResNet101	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9066
All2_100_21	U-Net+ResNet34	DL	4×10^{-5}	0,8943
All2_100_22	U-Net+ResNet34	DL	8×10^{-5}	0,9001
All2_100_23	U-Net+ResNet34	DL	$1{,}5 \times 10^{-4}$	0,8935
All2_100_24	U-Net+ResNet34	$CEFTL_1$	4×10^{-5}	0,8980
All2_100_25	U-Net+ResNet34	$CEFTL_1$	8×10^{-5}	0,9079
All2_100_26	U-Net+ResNet34	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,9011
All2_100_27	U-Net+ResNet34	$CEFTL_2$	4×10^{-5}	0,9059
All2_100_28	U-Net+ResNet34	$CEFTL_2$	8×10^{-5}	0,8943
All2_100_29	U-Net+ResNet34	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,8930
All2_100_31	FCN+ResNet101	DL	4×10^{-5}	0,8789
All2_100_32	FCN+ResNet101	DL	8×10^{-5}	0,8975
All2_100_33	FCN+ResNet101	DL	$1{,}5 \times 10^{-4}$	0,8903
All2_100_34	FCN+ResNet101	$CEFTL_1$	4×10^{-5}	0,9042
All2_100_35	FCN+ResNet101	$CEFTL_1$	8×10^{-5}	0,9064
All2_100_36	FCN+ResNet101	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,8898
All2_100_37	FCN+ResNet101	$CEFTL_2$	4×10^{-5}	0,8805
All2_100_38	FCN+ResNet101	$CEFTL_2$	8×10^{-5}	0,8888
All2_100_39	FCN+ResNet101	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,8937

Tabla 6.7: Resultados de IoU para cada combinación de hiperparámetros según el enfoque B para el 100 % de imágenes del conjunto de datos.

Id. modelo	*Arquitectura*	*F. de costo*	*Tasa*	*IoU SJM*
All2_50_11	U-Net+ResNet101	DL	4×10^{-5}	0,8981
All2_50_12	U-Net+ResNet101	DL	8×10^{-5}	0,8945
All2_50_13	U-Net+ResNet101	DL	$1{,}5 \times 10^{-4}$	0,8884
All2_50_14	U-Net+ResNet101	$CEFTL_1$	4×10^{-5}	0,9014
All2_50_15	U-Net+ResNet101	$CEFTL_1$	8×10^{-5}	0,8957
All2_50_16	U-Net+ResNet101	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,8871
All2_50_17	U-Net+ResNet101	$CEFTL_2$	4×10^{-5}	**0,9101**
All2_50_18	U-Net+ResNet101	$CEFTL_2$	8×10^{-5}	0,8900
All2_50_19	U-Net+ResNet101	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,8671
All2_50_21	U-Net+ResNet34	DL	4×10^{-5}	0,8845
All2_50_22	U-Net+ResNet34	DL	8×10^{-5}	0,8952
All2_50_23	U-Net+ResNet34	DL	$1{,}5 \times 10^{-4}$	0,8942
All2_50_24	U-Net+ResNet34	$CEFTL_1$	4×10^{-5}	0,9073
All2_50_25	U-Net+ResNet34	$CEFTL_1$	8×10^{-5}	0,9044
All2_50_26	U-Net+ResNet34	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,9022
All2_50_27	U-Net+ResNet34	$CEFTL_2$	4×10^{-5}	0,9009
All2_50_28	U-Net+ResNet34	$CEFTL_2$	8×10^{-5}	0,9035
All2_50_29	U-Net+ResNet34	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,9058
All2_50_31	FCN+ResNet101	DL	4×10^{-5}	0,8885
All2_50_32	FCN+ResNet101	DL	8×10^{-5}	0,8905
All2_50_33	FCN+ResNet101	DL	$1{,}5 \times 10^{-4}$	0,8748
All2_50_34	FCN+ResNet101	$CEFTL_1$	4×10^{-5}	0,8949
All2_50_35	FCN+ResNet101	$CEFTL_1$	8×10^{-5}	0,8724
All2_50_36	FCN+ResNet101	$CEFTL_1$	$1{,}5 \times 10^{-4}$	0,8579
All2_50_37	FCN+ResNet101	$CEFTL_2$	4×10^{-5}	0,9022
All2_50_38	FCN+ResNet101	$CEFTL_2$	8×10^{-5}	0,8785
All2_50_39	FCN+ResNet101	$CEFTL_2$	$1{,}5 \times 10^{-4}$	0,8778

Tabla 6.8: Resultados de IoU para cada combinación de hiperparámetros según el enfoque B para el 50 % de imágenes del conjunto de datos.

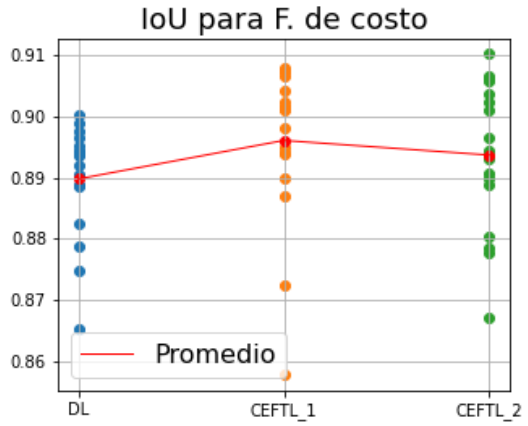
6.6.3.2. Desempeño por función de costo

Mientras que el IoU promedio para DL fue 0,8909, para $CEFTL_1$ y $CEFTL_2$ se obtuvieron los valores 0,8963 y 0,8946 respectivamente, lo que indica un mejor desempeño de la función de costo CEFTL en general por sobre DL.

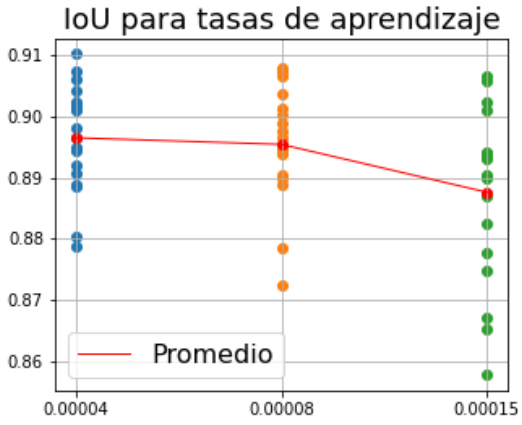
6.6.3.3. Desempeño por tasa de aprendizaje

En el caso de la tasa de aprendizaje se obtuvieron valores muy cercanos para los dos valores más pequeños de entre los evaluados. Los modelos entrenados con un valor de tasa de 4×10^{-5} lograron un IoU promedio de 0,8956 y los

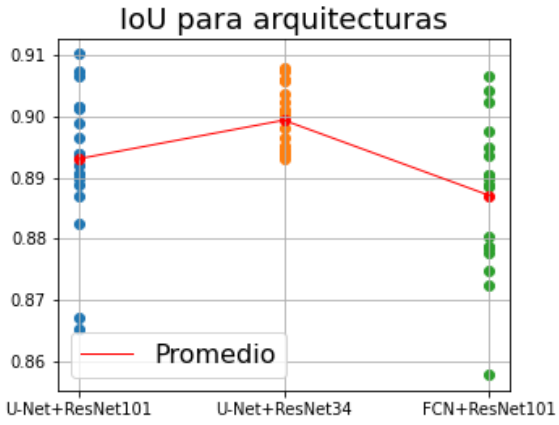
que lo hicieron con tasa de 8×10^{-5} promediaron el valor 0,8958. La cercanía entre los resultados para estos dos valores de tasa de aprendizaje indica que probablemente para ambos se convergió a los mínimos locales de las funciones. El valor restante, sin embargo, presenta un IoU promedio de 0,8899. Al darse el mismo escenario que para el enfoque A explicado en la sección 6.6.2.3, podemos asumir que $1{,}5 \times 10^{-4}$ es una tasa alta para las funciones de costo que se tomaron en cuenta.



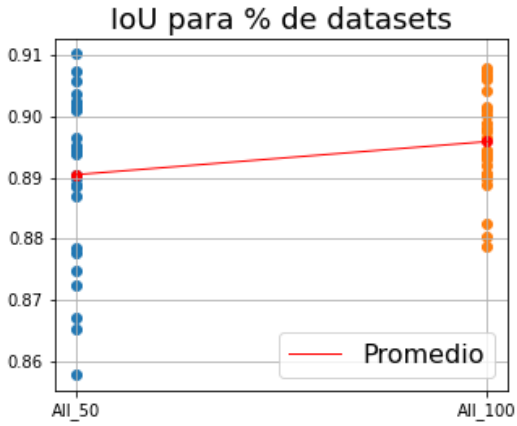
(a) IoU segmentado por función de costo



(b) IoU segmentado por tasa de aprendizaje



(c) IoU segmentado por arquitectura



(d) IoU segmentado por proporción de dataset

Figura 6.6: Gráficas de valores de IoU obtenidos para cada modelo según el enfoque B. Cada gráfica está segmentada por hiperparámetro considerado.

6.6.3.4. Desempeño por arquitectura

Como observación destacable de los resultados para las arquitecturas se tiene que hubo una predominancia de las arquitecturas U-Net sobre la FCN, lo cual es un comportamiento que también fue identificado durante los resultados del enfoque A para este hiperparámetro. U-Net+ResNet34 promedió un puntaje de IoU de 0,8992, mientras que U-Net+ResNet101 y FCN+ResNet101 promediaron 0,8954 y 0,8871 correspondientemente. La diferencia entre los dos mejores valores no llegó al céntimo de unidad, por lo que los encoders utilizados en esos dos casos no afectaron sensiblemente al IoU final.

6.6.3.5. Desempeño por proporción de dataset

Los modelos entrenados con el 100 % de los datos obtuvo un IoU promedio de 0,8958, en tanto que los modelos entrenados con el 50 % de los datos promediaron un puntaje de 0,8914. Al igual que para el enfoque A, se obtuvo que la mejor proporción de dataset es la de 100 %. Sin embargo, para el enfoque B no puede considerarse un escenario de sobreajuste como la causa de este resultado, dado que en el enfoque B los modelos fueron evaluados sobre un dataset cuyas imágenes tienen características diferentes a las de los conjuntos con los cuales entrenaron.

Por otra parte, si bien es cierto que la política del enfoque B durante la etapa 2 tuvo como consecuencia el mantener los datasets más propensos a obtener mejores resultados para SJM, esa política a priori no aseguraba que una mayor proporción de los datasets en el entrenamiento tendría como producto un mejor cumplimiento de la tarea.

(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.7: Predicción del modelo All2_50_17 sobre la imagen "image1280_-1280.jpg" del conjunto San José de las Matas.

6.7. Etapa 4 - Calibración de modelos

Como resultado de la sección anterior se obtuvo que los mejores modelos según el enfoque A fueron All_100_24, All_100_25 y All_100_22, mientras que para el B fueron All2_50_17, All2_100_25 y All2_100_15. Una vez seleccionados los modelos entrenados fue de interés calibrarlos con alguno de los mecanismos de calibración expuestos en la sección 2.8.3.

En la tabla 6.9 se pueden observar los resultados numéricos de la calibración. Allí se muestran las temperaturas T encontradas que minimizan ECE. Los valores buscados de T pertenecen al intervalo $[1; 10]$ con un paso de 0,1. Adicionalmente, se evidencia que para los casos particulares de los modelos seleccionados en la etapa 3 el hecho de calibrarlos hizo que mejorara su métrica de IoU. La mejora en la métrica IoU tras el proceso de calibración se puede explicar por el comportamiento manifiesto en los diagramas de las figuras 6.8 y 6.9, donde se aprecia que los modelos, previos a ser calibrados, subestiman para las probabilidades bajas ($[0{,}2; 0{,}5)$) y sobrestiman para las altas ($[0{,}6; 1{,}0)$). Por ejemplo, para las muestras en las que los modelos predijeron que la posibilidad de ser positivas era de alrededor del 30 %, en todos los casos por encima del 40 % lo eran. De forma inversa, cuando los modelos predijeron que la posibilidad de ser positivas era de alrededor del 90 %, en todos los casos solo por debajo del 60 % lo eran. Los diagramas de confianza como los expuestos en las figuras 6.8 y 6.9 son consecuencia de unas predicciones que no son acertadas en promedio para los datos comprendidos por los intervalos donde

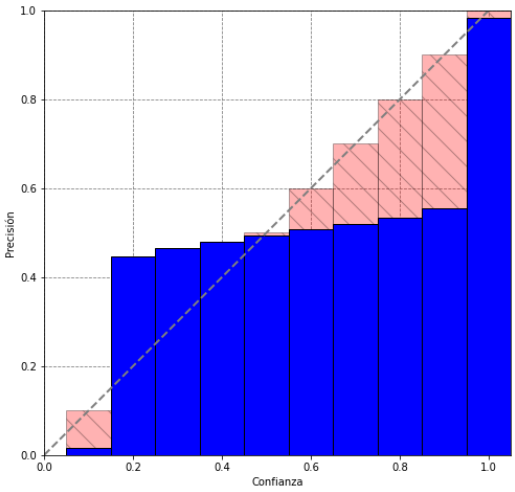
hay subestimación y sobrestimación. Al calibrar los modelos, las muestras de los intervalos bajos son trasladadas gráficamente hacia unos más altos, lo que genera un aumento en los aciertos de los ejemplos predichos como positivos (confianza superior a 0,5) y finalmente en una mejora en el IoU final.

Es importante mencionar que aunque es esperable que un modelo calibrado cambie su precisión respecto a su versión no calibrada (sobre todo por los casos en el entorno de confianza con valor 0,5), esa variación no siempre es en favor de la métrica y puede incluso degradarla. El hecho que para los modelos estudiados haya una mejora en el desempeño tras la calibración es una consecuencia de sus particularidades intrínsecas y de las de los datasets utilizados.

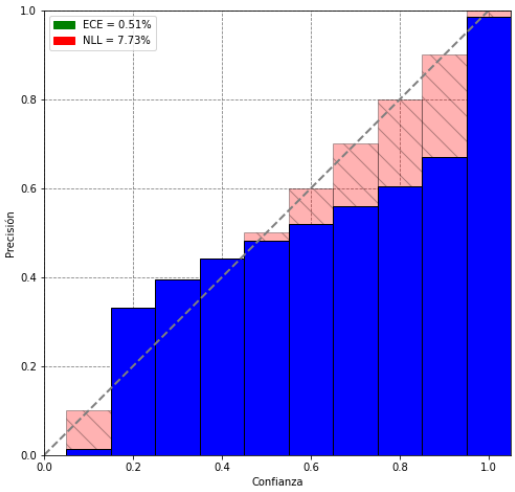
Modelo	*Temperatura*	*IoU pre-cal.*	*IoU post-cal.*
All_100_24	2,8423	0,9665	0,9679
All_100_25	3.1500	0,9661	0,9676
All_100_22	3,5640	0,9658	0,9676
All2_50_17	1,9800	0,9593	0,9604
All2_100_25	2,7900	0,9557	0,9577
All2_100_15	3,5640	0,9592	0,9607

Tabla 6.9: Temperatura encontrada y variación en IoU para los mejores modelos según los enfoques A y B.

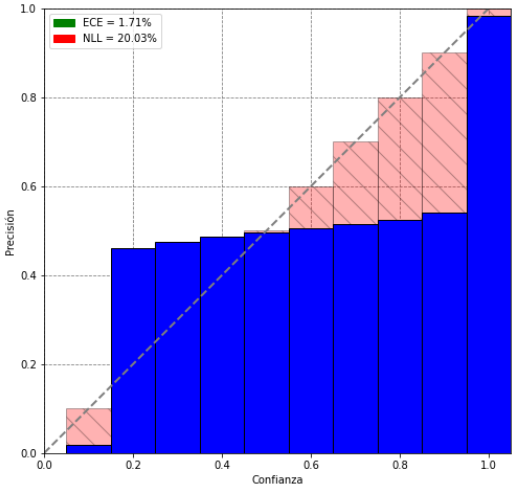
Continuando con el análisis de los diagramas de confianza producto de la calibración, para los extremos 0 y 1 la tendencia cambia respecto a los referidos anteriormente. Para el extremo 1 el modelo predijo con una precisión muy cercana a su confianza, lo que significa que cuando la confianza era muy cercana a ese valor, el modelo acertó la gran mayoría de las veces. Un comportamiento inverso se puede ver en el intervalo $[0{,}1; 0{,}2)$. En lo que refiere al intervalo $[0{,}5; 0{,}6)$, en él la confianza y precisión alcanzaron valores muy similares para todos los modelos. Finalmente, se agrega la figura 6.10, donde se comparan dos predicciones, cada una perteneciente a un modelo calibrado de cada enfoque.



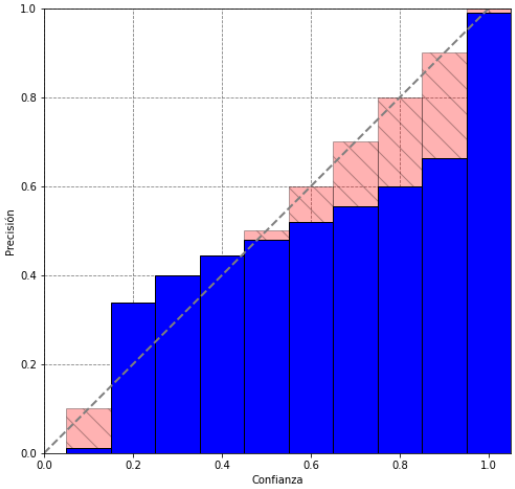
(a) All_100_24 sin calibrar



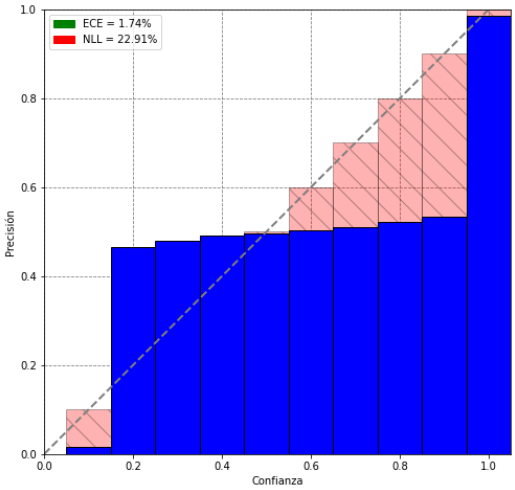
(b) All_100_24 luego de calibrado minimizando ECE



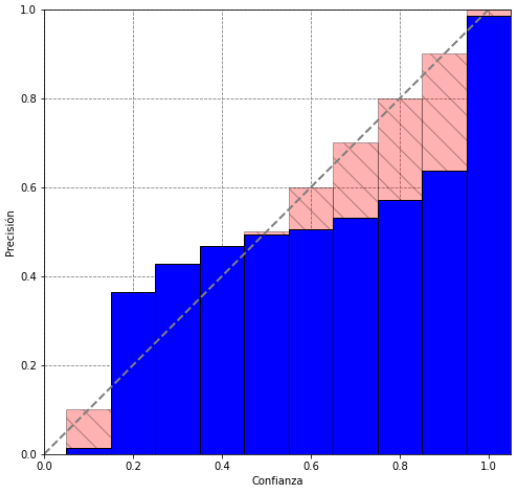
(c) All_100_24 sin calibrar



(d) All_100_25 luego de calibrado minimizando ECE

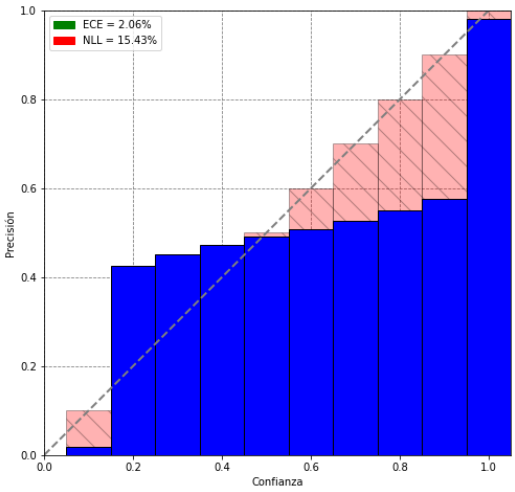


(e) All_100_22 sin calibrar

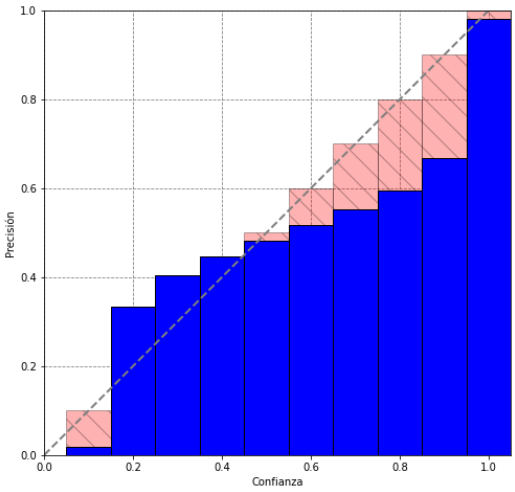


(f) All_100_22 luego de calibrado minimizando ECE

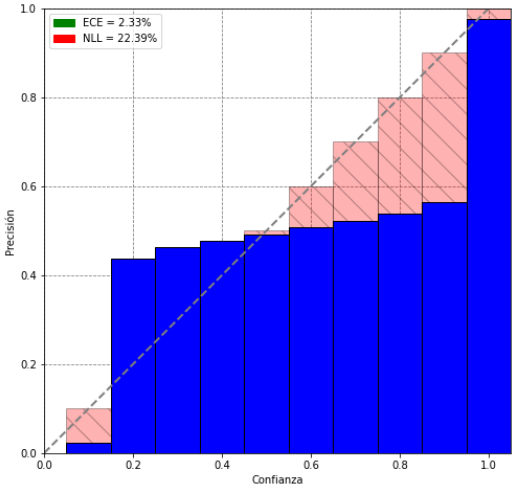
Figura 6.8: Diagramas de confianza para los tres mejores modelos según el enfoque A antes y luego del proceso de calibración.



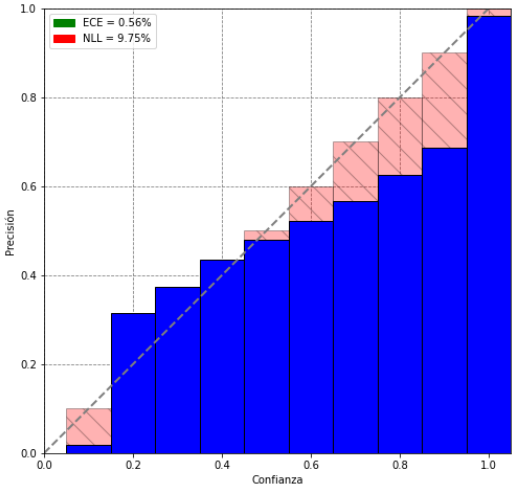
(a) All2_50_17 sin calibrar



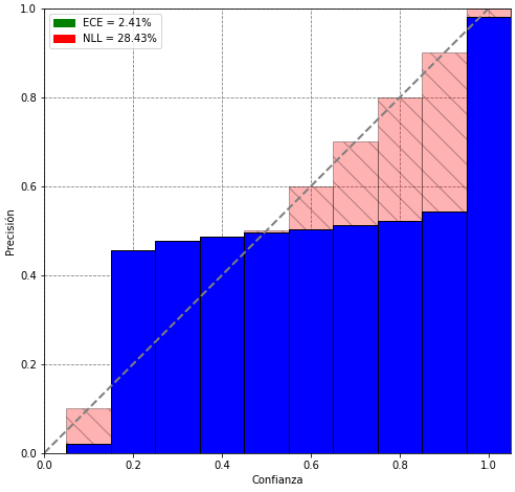
(b) All2_50_17 luego de calibrado minimizando ECE



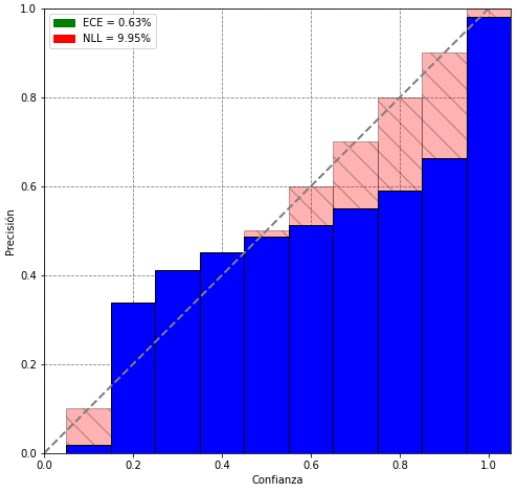
(c) All2_100_25 sin calibrar



(d) All2_100_25 luego de calibrado minimizando ECE



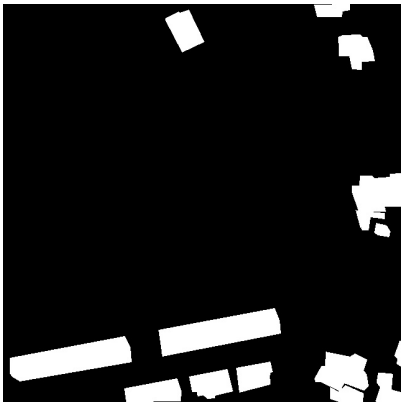
(e) All2_100_15 sin calibrar



(f) All2_100_15 luego de calibrado minimizando ECE

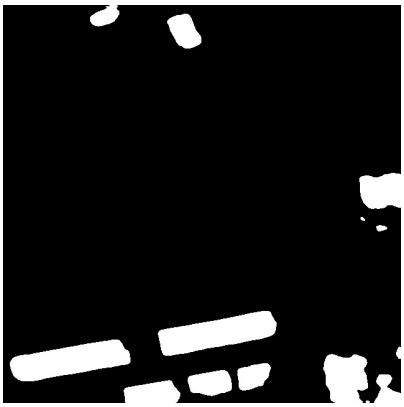
Figura 6.9: Diagramas de confianza para los tres mejores modelos según el enfoque B antes y luego del proceso de calibración.

(a) Imagen original



(b) Mapa de clases verdadero

(c) Predicción de modelo All_50_16 (enfoque A) calibrado



(d) Predicción de modelo All2_50_17 (enfoque B) calibrado

Figura 6.10: Cuadro comparativo donde se muestra una imagen original de SJM, su correspondiente mapa de clases y las predicciones de los modelos All_50_16 (enfoque A) y All2_50_17 (enfoque B) calibrados.

6.8. Etapa 5 - Votación de modelos

A partir de los modelos calibrados obtenidos en la etapa 4, se plantearon dos estrategias de votación para clasificar cada píxel de cada imagen del dataset de San José de las Matas. Las estrategias son el soft y hard voting sin ponderación para ninguno de los modelos.

6.8.1. Resultados

Los resultados de aplicar la votación de modelos se exponen en la tabla 6.10, donde es posible comprobar que hubo un mejor rendimiento del soft voting por encima de hard voting en ambos enfoques (A y B). Adicionalmente, existe una diferencia en el desempeño de ambos enfoques en favor del B, tanto para el hard como el soft voting. Este resultado puede considerarse como el esperado dado que todas las decisiones a lo largo de las distintas etapas para el enfoque B se tomaron eligiendo aquellos hiperparámetros que maximizaban el rendimiento sobre SJM, dataset que es el tenido en cuenta en la presente etapa.

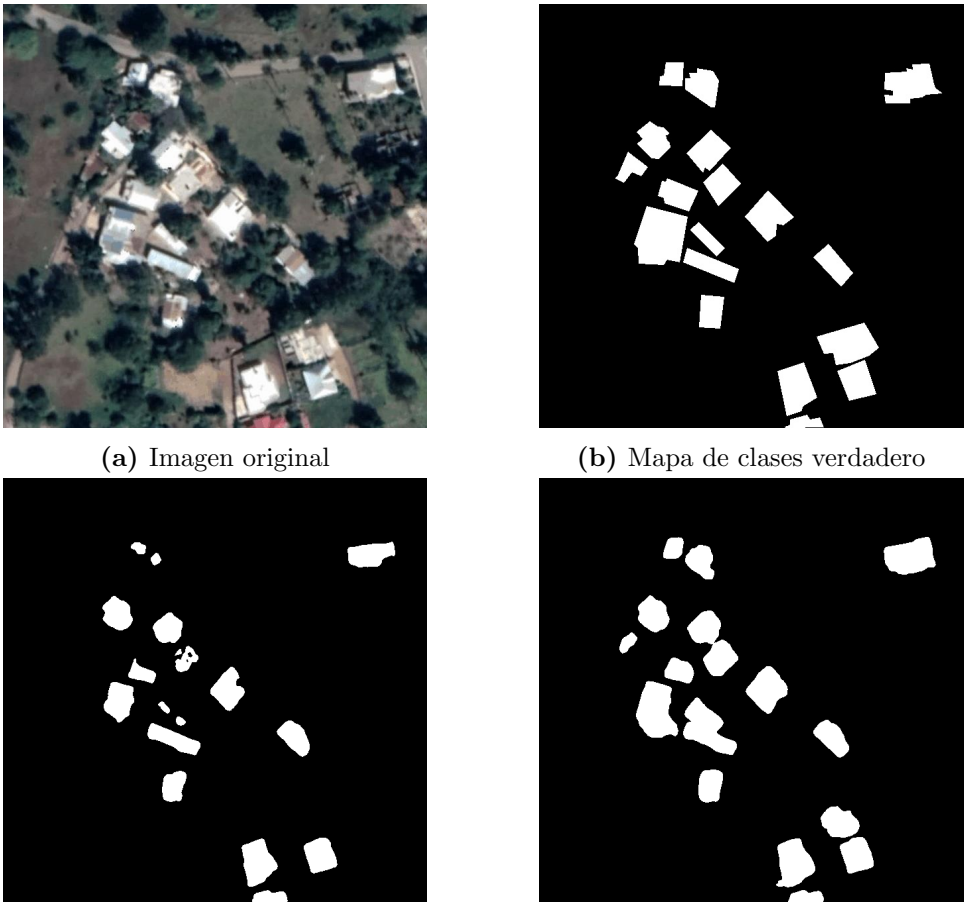
Enfoque	*Votación*	*IoU*
A	soft	0,8850
A	hard	0,8774
B	soft	0,9163
B	hard	0,9089

Tabla 6.10: Resultados de aplicar soft y hard voting para los tres mejores modelos de los enfoques A y B

En lo que respecta a la comparación del cumplimiento de la votación de modelos en contraposición a los obtenidos de forma individual por cada modelo, se obtuvo que el soft voting del enfoque A fue mejor al individual de los modelos All_100_24 y All_100_22 pero empeoró para el que había sido conseguido por All_100_25 (tabla 6.5). Para el caso de hard voting solo supera al rendimiento individual de All_100_24.

En el escenario del enfoque B, sin embargo, el desempeño del soft voting superó a todos los rendimientos individuales de los modelos que participan en él: All2_50_17 (tabla 6.8), All2_100_25 y All2_100_15 (tabla 6.7). Evaluando el hard voting, nos encontramos con que su puntaje fue inferior al que había sido conseguido por All2_50_17, pero superior al de los otros dos restantes.

En la figura 6.11 se encuentra un ejemplo de predicción para soft voting de ambos enfoques.



(a) Imagen original

(b) Mapa de clases verdadero

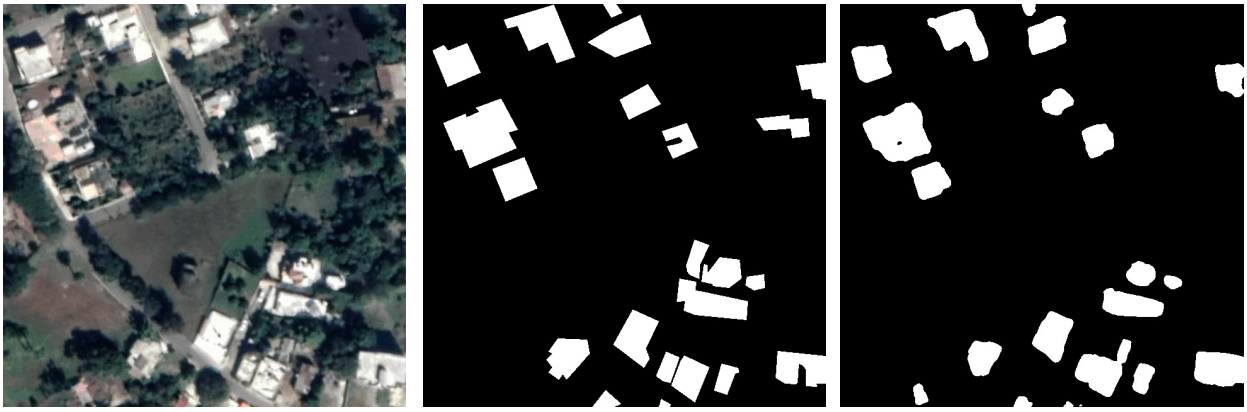
(c) Predicción soft voting de modelos del enfoque A

(d) Predicción soft voting de modelos del enfoque B

Figura 6.11: Cuadro comparativo donde se muestra una imagen original de SJM, su mapa de clases y las predicciones soft voting de los enfoques A y B.

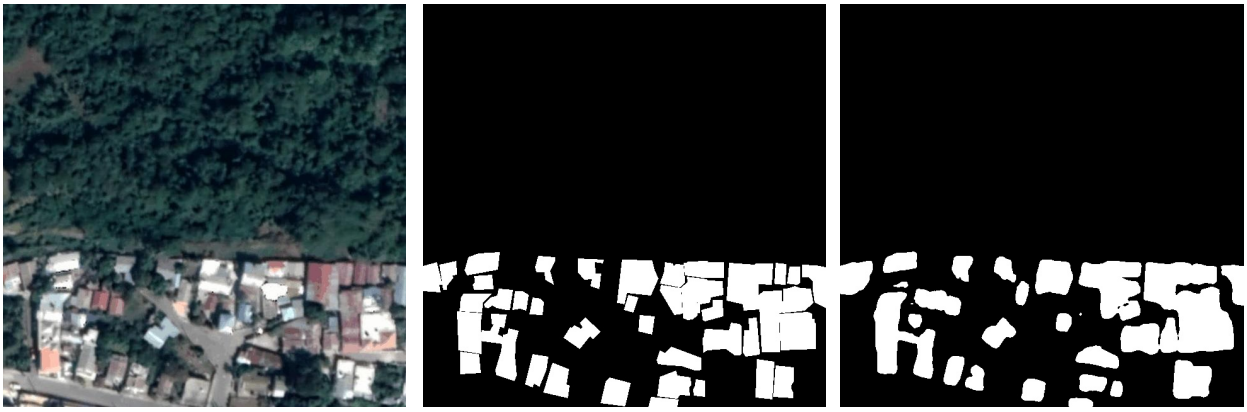
6.8.2. Ejemplos de predicciones

En la presente sección se muestran ejemplos de predicciones resultado de aplicar soft voting entre los modelos All2_50_17, All2_100_25 y All2_100_15 sobre imágenes del dataset de SJM, con el fin de presentar visualmente el desempeño conseguido. Los ejemplos presentan imágenes de diferentes características, desde zonas residenciales con viviendas próximas entre sí hasta áreas rurales o con otros tipos de construcciones, como un campo de juego de béisbol.



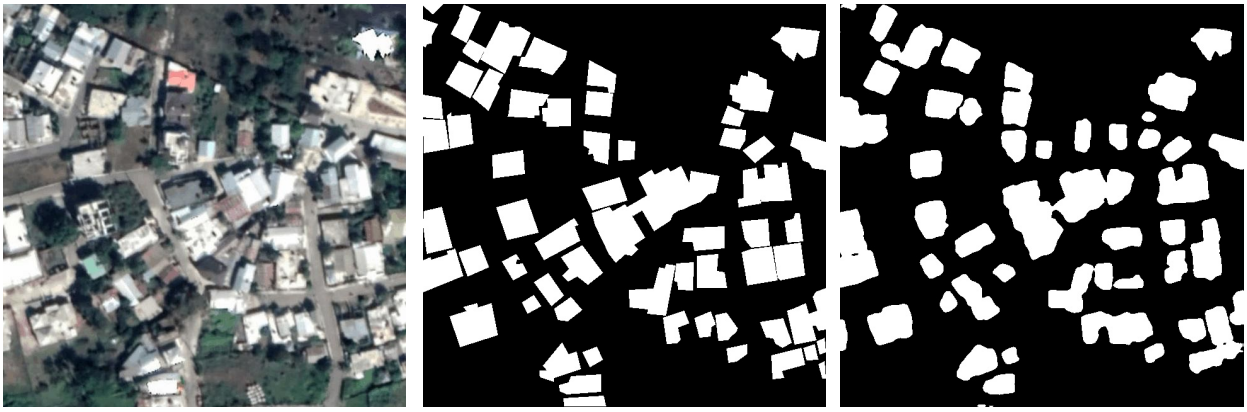
(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.12: "image0_640.jpg" del conjunto San José de las Matas



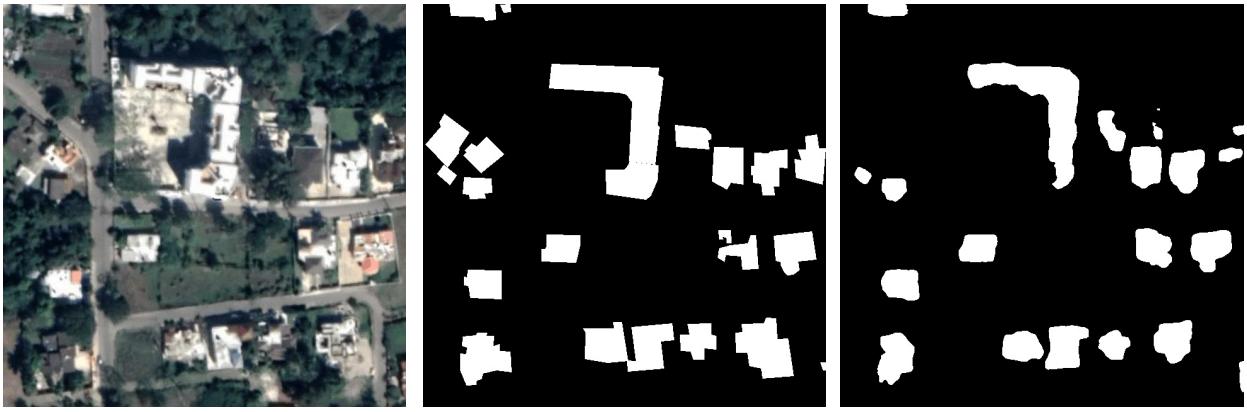
(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.13: "image640_5760.jpg" del conjunto San José de las Matas



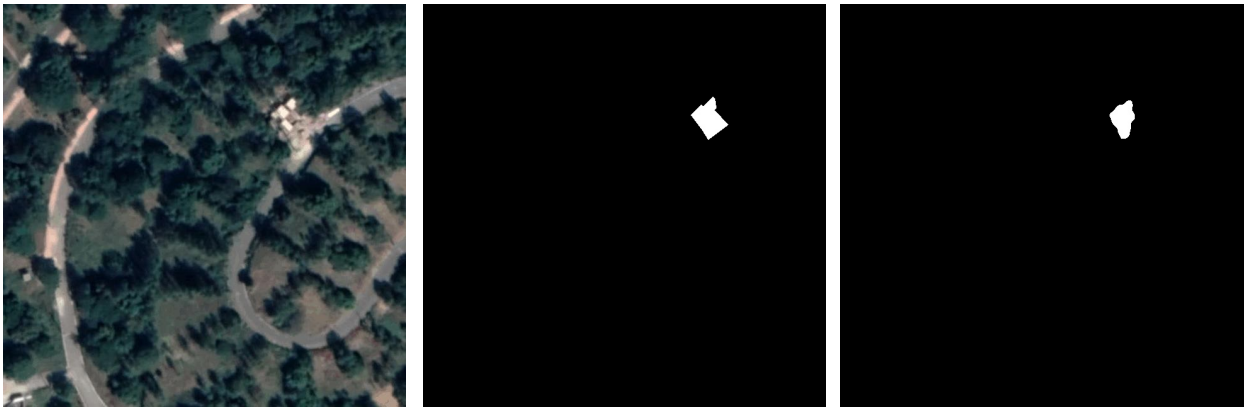
(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.14: "image640_3200.jpg" del conjunto San José de las Matas



(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.15: "image640_1280.jpg" del conjunto San José de las Matas



(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.16: "image3200_7680.jpg" del conjunto San José de las Matas

(a) Imagen original **(b)** Mapa de verdad **(c)** Predicción

Figura 6.17: "image3200_3840.jpg" del conjunto San José de las Matas

6.9. Aplicación de los modelos para estimación de la población - caso de estudio

Con el fin de validar la utilidad de la extracción de edificios sobre una zona poblada en la estimación de la cantidad de sus habitantes, se aplicaron los modelos obtenidos en una área particular de Montevideo, Uruguay.

6.9.1. Área de estudio

El área de estudio, referenciada de aquí en más como "ConcE", se expone en la figura 6.18 y comprende aproximadamente 36 hectáreas del barrio Conciliación, perteneciente al Centro Comunal Zonal 13 (CCZ 13) de la ciudad de Montevideo. Se optó por seleccionar una región con edificaciones bajas con el objetivo de evitar edificios de gran altura, ya que esa es una dimensión que no es bien identificada desde imágenes satelitales. Aunque ConcE está en su mayoría ocupada por edificaciones, también presenta regiones donde hay territorio sin edificar.

Figura 6.18: Área sobre la que se estimó la población en barrio Conciliación, Montevideo, Uruguay.

Según el Informe Censos 2011 para Montevideo y Área Metropolitana [33], realizado por la Unidad de Estadística y Gestión Estratégica de la Intendencia Municipal de Montevideo (IMM), la densidad de viviendas en el CCZ 13 es de 20 viviendas por hectárea, con un promedio de tres habitantes por vivienda. Al remitirse a estos datos, se estimó que en 36,2 hectáreas (área comprendida por ConcE) del CCZ 13 existen $36{,}2 \times 20 = 724$ viviendas y $724 \times 3 = 2.172$ habitantes. No obstante, el CCZ 13 abarca hasta cuatro barrios distintos, por lo que ConcE podría no estar bien representada en la dimensión total del centro comunal al que te pertenece. Con el propósito de contar con datos más específicos sobre ConcE, se recurrió al Sistema de Información Geográfica (SIG) [74] de la IMM, donde se halló un grado de especificidad mayor en cuanto a la cantidad de personas que habitan en una zona acotada, de tamaño muy similares a las manzanas. Basándose en la delimitación de ConcE en SIG, como se muestra en la figura 6.19, y en el mapa de habitantes según el color expuesto en la tabla 6.11, se llegaron a estimar 3.158 habitantes en ConcE.

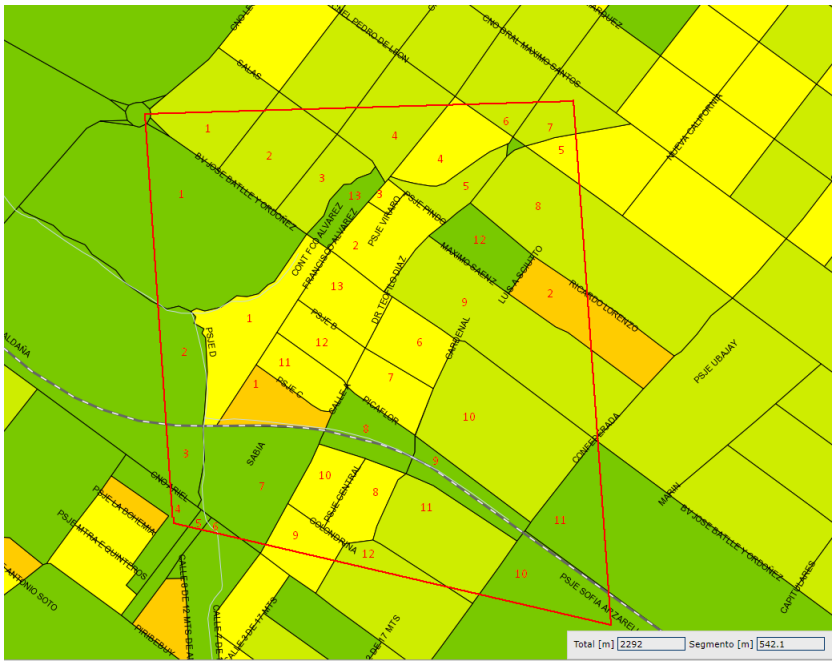


Figura 6.19: Imagen obtenida de SIG, las líneas rojas delimitan ConcE, los números identifican cada zona para cada color en ConcE.

Color	*Rango (habitantes)*	*Valor medio (habitantes)*	*Cant. en ConcE*	*Habitantes estimados*
	0 - 40	20	13	$20 \times 13 = 260$
	40,01 - 96	68	12	$68 \times 12 = 816$
	96,01 - 160	128	13	$128 \times 13 = 1664$
	160,01 - 258	209	2	$209 \times 2 = 418$

Tabla 6.11: Tabla utilizada para la estimación de habitantes de ConcE según la información obtenida de SIG.

6.9.2. Resultados

Una dificultad con la que se encuentran los modelos utilizados en la segmentación de edificios es la de delimitar correctamente cada instancia cuando las edificaciones se encuentran muy cercanas en el espacio. Los modelos en este trabajo construidos no fueron ajenos a ese problema, por lo que el saber el tamaño promedio de las edificaciones en metros cuadrados era un dato útil a la hora de decidir cuántas viviendas comprende una instancia de segmentación de la clase Edificio. Sin embargo, no se encontró información acerca de ese dato, por lo que se estimó en $70m^2$ promedio por vivienda a partir de la observación y cálculo de una muestra de viviendas de la zona, utilizando las herramientas de medición que provee Google Maps. Un ejemplo de estimación a partir de una segmentación se aprecia en la figura 6.20, donde se contabilizan 25 viviendas (asumiendo que cada una en promedio tiene un tamaño de $70m^2$) y, por lo tanto, una población de 75 personas.

Tras aplicar las clasificaciones y cálculos presentados en los párrafos anteriores en esta sección sobre la zona definida, se tuvo como resultado una estimación de 930 viviendas y 2.790 habitantes totales. Esta estimación es un 28,5 % mayor a la obtenida con la información del Informe de Censo 2011 (2.172 habitantes) y un 11,7 % menor a la conseguida con la información de SIG (3.158). Si bien es positivo que la estimación lograda a partir de los modelos construidos se encuentre en el mismo orden de otras generadas a partir de datos reales, hay que considerar que el valor utilizado como tamaño promedio de una vivienda en ConcE fue inferido (aproximadamente) a partir de la medición de un grupo reducido de casos.

(a) **(b)**

Figura 6.20: Ejemplo de segmentación resultado de aplicar la votación de modelos en zona del barrio Conciliación, Montevideo. Al lado de cada instancia figura el número de viviendas que se contabilizan por cada una en función del área que ocupan y el tamaño de vivienda promedio estimado para esa zona.

Capítulo 7

Conclusiones y trabajo futuro

En el presente capítulo se exponen las conclusiones y las consideraciones finales del proyecto, así como las lineas de trabajo futuro.

7.1. Conclusiones

El objetivo principal del presente proyecto fue valorar la capacidad de distintas arquitecturas de redes neuronales profundas en la tarea de extracción de edificios en escenarios definidos sobre ciudades con poca o nula planificación urbana. A partir de los resultados de este trabajo es posible desarrollar estrategias automáticas para estimar la población y planificar el saneamiento de aguas residuales de las zonas de estudio.

A lo largo de este proyecto se hizo un análisis de las principales arquitecturas de redes neuronales profundas utilizadas para la segmentación de imágenes, particularmente para la tarea de extracción de edificios utilizando imágenes aéreas y satelitales. En base a los trabajos relacionados se seleccionaron las arquitecturas FCN, ResNet, U-Net y sus combinaciones, e hiperparámetros a ajustar en función de la métrica de desempeño IoU. Se hizo especial énfasis en la exploración del encoder de la red, utilizando la arquitectura ResNet y sus variantes con ese fin. Como línea de base se implementó un algoritmo de segmentación pseudo-aleatorio que contaba con información de cada imagen previo a segmentarla.

Para el entrenamiento de los modelos fueron utilizados cuatro conjuntos de datos (Massachusetts Buildings Dataset, Inria Aerial Image Labeling Dataset, WHU Building Dataset - Christchurch, Satellite dataset II East Asia),

que comprenden ciudades con una evidente planificación urbana y áreas no urbanizadas. Los conjuntos de datos incluyen imágenes de distintas dimensiones y calidad. Por otra parte, se generó un dataset para la ciudad San José de las Matas (República Dominicana), que se desarrolló con poca planificación urbana. El dataset generado fue anotado manualmente para ser utilizado en este trabajo como conjunto de verificación.

Con el fin de obtener las mejores arquitecturas e hiperparámetros para el problema planteado, se evaluaron los modelos candidatos de forma progresiva y eliminatoria. El proceso de selección se enmarcó en tres etapas: (1) selección de las tres mejores arquitecturas según su desempeño promedio para cada dataset; (2) redimensionamiento de datasets con imágenes muy grandes y (3) selección de hiperparámetros para modelos entrenados con la unificación de los datasets. Adicionalmente, se definieron dos enfoques para determinar si un modelo debía seleccionarse por sobre otro: uno se basó en el desempeño para el conjunto de verificación del dataset con el que entrenó (enfoque A), y el otro en el desempeño para el conjunto de San José de las Matas (enfoque B).

Los modelos obtenidos en la etapa 1 alcanzaron un IoU mayor al conseguido por la línea de base. A medida que avanzaron las etapas, se observó un mejor desempeño de las arquitecturas basadas en U-Net frente a las variantes de FCN con encoder ResNet consideradas. Concretamente, el mejor IoU obtenido para San José de las Matas por un modelo con arquitectura FCN fue 0,9064, mientras que cuatro modelos basados en U-Net superaron ese valor, teniendo como máximo un IoU de 0,9101. Como evaluación final, se aplicó la votación de modelos sobre el conjunto San José de las Matas, obteniendo 0,9163 como valor máximo de IoU para el caso de soft voting con modelos calibrados. Estos resultados validan la capacidad de generalización de los modelos entrenados, capaces de extraer edificios de una ciudad con características topográficas disímiles a las de los conjuntos de entrenamiento.

Por último, se tomó un área de estudio perteneciente a Montevideo, de la cual se tienen datos sobre la densidad de viviendas y habitantes, para estudiar la viabilidad de estimación de la población de la zona a partir de la extracción de edificios. El producto del proceso de estimación aplicado tuvo un error de un $11,7\,\%$ respecto a la estimación más fiable para esa área en particular.

Los métodos desarrollados son componentes valiosos en el diseño de un sistema automático que aplique inteligencia computacional para la planificación y el desarrollo de entornos urbanos bajo el paradigma de ciudades inteligentes.

7.2. Trabajo futuro

A continuación se mencionan posibles mejoras y alternativas a explorar en la resolución del problema.

Se considera que la exploración de otras arquitecturas de redes neuronales para la tarea de segmentación de imágenes, como DeepLabV3 [9] o Gated-SCNN [78], puede ser provechosa. En el mismo sentido, podrían considerarse nuevas funciones de pérdida como Lovász hinge o las basadas en ponderación de píxeles [73].

Como se mencionó en la sección 5.2.3, existe un claro desbalance en cantidad de imágenes entre algunos datasets considerados para el entrenamiento. Por esta razón, es recomendable aplicar más técnicas de data augmentation sobre los conjuntos menos representados para generar más datos sobre las ciudades comprendidas por ellos. Otra tarea que podría incrementar la calidad de los modelos entrenados es la de generación de más conjuntos de datos anotados sobre nuevas ciudades.

Si bien se hizo una estimación de población para una zona dada, el área que comprende es bastante acotada, por lo que se encuentra oportuno realizar una que abarque más territorio. Para ampliar la zona de estudio es necesario no solo contar con datos demográficos más específicos (habitantes por manzana, tamaño de vivienda promedio), sino que además la información debiera ser actualizada.

Capítulo 8

Referencias bibliográficas

[1] Abraham N y Khan NM. «A Novel Focal Tversky Loss Function With Improved Attention U-Net for Lesion Segmentation». En: *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. 2019, págs. 683-687.

[2] Audebert N, Le Saux B y Lefèvre S. «Segment-before-Detect: Vehicle Detection and Classification through Semantic Segmentation of Aerial Images». En: *Remote Sensing* 9.4 (2017).

[3] Badrinarayanan V, Kendall A y Cipolla R. «SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.12 (2017), págs. 2481-2495.

[4] Benedek C, Descombes X y Zerubia J. «Building Development Monitoring in Multitemporal Remotely Sensed Image Pairs with Stochastic Birth-Death Dynamics». En: *IEEE Trans. Pattern Anal. Mach. Intell.* 34.1 (ene. de 2012), págs. 33-50.

[5] Bengio Y, Simard P y Frasconi P. «Learning Long-Term Dependencies with Gradient Descent is Difficult». En: *Trans. Neur. Netw.* 5.2 (mar. de 1994), págs. 157-166.

[6] Bengio Y. «Learning Deep Architectures for AI». En: *Foundations and Trends in Machine Learning* 2.1 (2009), págs. 1-127.

[7] Boykov Y y Jolly M. «Interactive graph cuts for optimal boundary region segmentation of objects in N-D images». En: *Computer Vision,*

2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on. Vol. 1. 2001, 105-112 vol.1.

[8] Buslaev A y col. «Albumentations: Fast and Flexible Image Augmentations». En: *Information* 11.2 (2020).

[9] Chen LC y col. «Rethinking Atrous Convolution for Semantic Image Segmentation». En: *CoRR* abs/1706.05587 (2017).

[10] Chen X y col. «Vehicle Detection in Satellite Images by Hybrid Deep Convolutional Neural Networks». En: *IEEE Geoscience and Remote Sensing Letters* 11.10 (2014), págs. 1797-1801.

[11] Chen Z y col. «Automatic Detection of Track and Fields in China from High-Resolution Satellite Images Using Multi-Scale-Fused Single Shot MultiBox Detector». En: *Remote Sensing* 11.11 (2019).

[12] Cheng G y col. «Accurate Urban Road Centerline Extraction from VHR Imagery via Multiscale Segmentation and Tensor Voting». En: *CoRR* abs/1508.06163 (2015).

[13] Cheng G y col. «Urban road extraction via graph cuts based probability propagation». En: *2014 IEEE International Conference on Image Processing (ICIP)* (2014), págs. 5072-5076.

[14] Dahl M y Javadi S. «Analytical Modeling for a Video-Based Vehicle Speed Measurement Framework». En: *Sensors* 20.1 (2020).

[15] Das S y Varghese K. *Use of salient features for the design of a multistage framework to extract roads from high-resolution multispectral satellite images*. 2011.

[16] Deng J y col. «ImageNet: A large-scale hierarchical image database». En: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, págs. 248-255.

[17] Esteva A y col. «Dermatologist-level classification of skin cancer with deep neural networks». En: *Nature* 542 (ene. de 2017), págs. 115-.

[18] Forsyth D. *Computer vision : a modern approach*. 2003.

[19] Glorot X y Bengio Y. «Understanding the difficulty of training deep feedforward neural networks». En: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Vol. 9. 13–15 May de 2010, págs. 249-256.

[20] Goodfellow I, Bengio Y y Courville A. *Deep Learning*. 2016.

[21] *Google Colab*. https://research.google.com/colaboratory/faq.html. Accedido: 08-03-2022.

[22] *Google Drive*. https://www.google.com/intl/es/drive/. Accedido: 08-03-2022.

[23] Guo C y col. *On Calibration of Modern Neural Networks*. 2017.

[24] Guo M y col. «Building Extraction Based on U-Net with an Attention Block and Multiple Losses». En: *Remote Sensing* 12 (2020).

[25] He K y Sun J. «Convolutional Neural Networks at Constrained Time Cost». En: *CoRR* abs/1412.1710 (2014).

[26] He K y col. «Deep Residual Learning for Image Recognition». En: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, págs. 770-778.

[27] Hebb D. *The organization of behavior: A neuropsychological theory*. Jun. de 1949.

[28] Hu J y col. «Road network extraction and intersection detection from aerial images by tracking road footprints». En: *IEEE Transactions on Geoscience and Remote Sensing* 45.12 (dic. de 2007), págs. 4144-4157.

[29] Huang T. «Computer vision: Evolution and promise». En: *CERN School of Computing*. 1996, págs. 21-25.

[30] Huang Z y col. «Building extraction from multi-source remote sensing images via deep deconvolution neural networks». En: *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2016, págs. 1835-1838.

[31] Hurtik P y Madrid N. «Bilinear Interpolation over fuzzified images: Enlargement». En: ago. de 2015.

[32] *imageaug*. https://imgaug.readthedocs.io/en/latest/. Accedido: 07-09-2021.

[33] *Informe Censos 2011: Montevideo y Área Metropolitana*. https://montevideo.gub.uy/sites/default/files/informe_censos_2011_mdeo_y_area_metro.pdf. Accedido: 13-05-2022.

[34] *Inria Aerial Image Labeling Dataset*. https://project.inria.fr/aerialimagelabeling/. Accedido: 07-09-2021.

[35] *International Symposium on Biomedical Imaging 2015 Challenges*. http://biomedicalimaging.org/2015/program/isbi-challenges/. Accedido: 07-09-2021.

[36] *Jupyter Notebook*. https://jupyter.org/. Accedido: 08-03-2022.

[37] Keys R. «Cubic convolution interpolation for digital image processing». En: *IEEE Transactions on Acoustics, Speech, and Signal Processing* 29.6 (1981), págs. 1153-1160.

[38] Kingma D y Ba J. *Adam: A Method for Stochastic Optimization*. 2014.

[39] Krizhevsky A, Sutskever I e Hinton G. «ImageNet Classification with Deep Convolutional Neural Networks». En: *Advances in Neural Information Processing Systems*. Vol. 25. 2012.

[40] *Labelbox*. https://labelbox.com/. Accedido: 08-03-2022.

[41] Lecun Y y col. «Gradient-based learning applied to document recognition». En: *Proceedings of the IEEE* 86.11 (1998), págs. 2278-2324.

[42] LeCun Y y col. «Efficient BackProp.» En: *Neural Networks: Tricks of the Trade (2nd ed.)* Vol. 7700. 2012, págs. 9-48.

[43] Lee T. «A Minimum Description Length-Based Image Segmentation Procedure, and Its Comparison with a Cross-Validation-Based Segmentation Procedure». En: *Journal of the American Statistical Association* 95.449 (2000), págs. 259-270.

[44] Lefevre S, Weber J y Sheeren D. «Automatic Building Extraction in VHR Images Using Advanced Morphological Operators». En: *2007 Urban Remote Sensing Joint Event*. 2007, págs. 1-5.

[45] Li J y col. «Hierarchical Disentangling Network for Building Extraction from Very High Resolution Optical Remote Sensing Imagery». En: *Remote Sensing* 14 (2022).

[46] Li Z y col. «A Deep Learning-Based Framework for Automated Extraction of Building Footprint Polygons from Very High-Resolution Aerial Imagery». En: *Remote Sensing* 13.18 (2021).

[47] Liu W y col. «SSD: Single Shot MultiBox Detector.» En: vol. 9905. 2016, págs. 21-37.

[48] Long J, Shelhamer E y Darrell T. «Fully Convolutional Networks for Semantic Segmentation». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Jun. de 2015.

[49] *Lua*. https://www.lua.org/. Accedido: 07-09-2021.

[50] Maggiori E y col. «Can Semantic Labeling Methods Generalize to Any City? The Inria Aerial Image Labeling Benchmark». En: *IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*. 2017.

[51] McCulloch W y Pitts W. «A logical calculus of the ideas immanent in nervous activity.» En: *Bulletin of Mathematical Biology* 5.4 (1943), págs. 115-133. ISSN: 00928240.

[52] Minsky M y Papert S. *Perceptrons: An Introduction to Computational Geometry*. 1969.

[53] Mitchell T. *Machine Learning*. 1997.

[54] Mnih V. «Machine Learning for Aerial Image Labeling». Tesis doct. University of Toronto, 2013.

[55] Mo G y Zhang S. «Vehicles detection in Traffic Flow». En: *2010 Sixth International Conference on Natural Computation*. Vol. 2. 2010, págs. 751-754.

[56] *NumPy*. https://numpy.org/. Accedido: 07-09-2021.

[57] Ojala T, Pietikainen M y Maenpaa T. «Multiresolution gray-scale and rotation invariant texture classification with local binary patterns». En: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.7 (2002), págs. 971-987.

[58] *OpenCV*. https://opencv.org/. Accedido: 07-09-2021.

[59] *Pandas*. https://pandas.pydata.org/. Accedido: 07-09-2021.

[60] *PyTorch*. https://pytorch.org/. Accedido: 07-09-2021.

[61] Ren S y col. «Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks». En: *Advances in Neural Information Processing Systems*. Vol. 28. 2015.

[62] *Road and Building Detection Datasets*. https://www.cs.toronto.edu/~vmnih/data/. Accedido: 07-09-2021.

[63] Ronneberger O, Fischer P y Brox T. «U-Net: Convolutional Networks for Biomedical Image Segmentation». En: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*. 2015, págs. 234-241.

[64] Rosenblatt F. «The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain». En: *Psychological Review* (1958), págs. 65-386.

[65] Rumelhart D, Hinton G y Williams R. «Learning Representations by Back-propagating Errors». En: *Nature* 323.6088 (1986), págs. 533-536.

[66] Salehi SSM, Erdogmus D y Gholipour A. *Tversky loss function for image segmentation using 3D fully convolutional deep networks*. 2017.

[67] Saxe A, McClelland J y Ganguli S. *Exact solutions to the nonlinear dynamics of learning in deep linear neural networks*. 2013.

[68] *SciPy*. https://www.scipy.org/. Accedido: 07-09-2021.

[69] *Segmentation Models PyTorch*. https://segmentation-modelspytorch.readthedocs.io/en/latest/. Accedido: 07-09-2021.

[70] Shorten C y Khoshgoftaar T. «A survey on Image Data Augmentation for Deep Learning». En: *Journal of Big Data* 6 (2019), págs. 1-48.

[71] Shunping Ji Shiqing Wei ML. «Fully building segmentation fromConvolutional Networks for Multi-Source Building Extraction from An Open Aerial and Satellite Imagery Dataset». En: *IEEE Transactions on geoscience and remote sensing* (2018).

[72] Simonyan K y Zisserman A. «Very deep convolutional networks for large-scale image recognition». En: (2014).

[73] Sirko W y col. *Continental-Scale Building Detection from High Resolution Satellite Imagery*. 2021.

[74] *Sistema de Información Geográfica - Intendencia Municipal de Montevideo*. https://sig.montevideo.gub.uy/. Accedido: 13-05-2022.

[75] Srivastava RK, Greff K y Schmidhuber J. *Highway Networks*. 2015.

[76] Sun X, Wang H y Fu K. «Automatic Detection of Geospatial Objects Using Taxonomic Semantics». En: *IEEE Geoscience and Remote Sensing Letters* 7.1 (2010), págs. 23-27.

[77] Szegedy C y col. «Going deeper with convolutions». En: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, págs. 1-9.

[78] Takikawa T y col. «Gated-SCNN: Gated Shape CNNs for Semantic Segmentation». En: *CoRR* abs/1907.05740 (2019).

[79] Tang T y col. «Vehicle Detection in Aerial Images Based on Region Convolutional Neural Networks and Hard Negative Example Mining». En: *Sensors* 17.2 (2017).

[80] *TensorFlow*. https://www.tensorflow.org/. Accedido: 07-09-2021.

[81] *Torch*. http://torch.ch/. Accedido: 07-09-2021.

[82] *Torchvision*. https://pytorch.org/vision/stable/index.html. Accedido: 07-09-2021.

[83] Turchenko V, Chalmers E y Luczak A. *A Deep Convolutional Auto-Encoder with Pooling - Unpooling Layers in Caffe*. 2017.

[84] Vakalopoulou M y col. «Building detection in very high resolution multispectral data with deep learning features». En: *Geoscience and Remote Sensing Symposium (IGARSS), 2015 IEEE International*. Jul. de 2015, págs. 1873-1876.

[85] Volpi M y col. «Supervised change detection in VHR images using contextual information and support vector machines». En: *International Journal of Applied Earth Observation and Geoinformation* 20 (feb. de 2013), págs. 77-85.

[86] Wang Ss, Hou X y Zhao X. «Automatic Building Extraction From High-Resolution Aerial Imagery via Fully Convolutional Encoder-Decoder Network With Non-Local Block». En: *IEEE Access* 8 (2020), págs. 7313-7322.

[87] *WHU Building Dataset*. http://gpcv.whu.edu.cn/data/building_dataset.html. Accedido: 07-09-2021.

[88] Wolpert D y Macready W. «No free lunch theorems for optimization.» En: *IEEE Transactions on Evolutionary Computation, Evolutionary Computation, IEEE Transactions on, IEEE Trans. Evol. Computat* 1.1 (1997), págs. 67-82. ISSN: 1941-0026.

[89] *WordNet: An Electronic Lexical Database*. 1998.

[90] Xie S y col. «Aggregated Residual Transformations for Deep Neural Networks». En: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, págs. 5987-5995.

[91] Yan Z y col. «Web-Net: A Novel Nest Networks with Ultra-Hierarchical Sampling for Building Extraction from Aerial Imageries.» En: *Remote Sensing* 11.16 (2019), pág. 1897.

[92] Zhu J y col. «Urban Traffic Density Estimation Based on Ultrahigh-Resolution UAV Video and Deep Neural Network». En: *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 11.12 (2018), págs. 4968-4981.