Universidad de Oviedo
Universidá d'Uviéu
University of Oviedo

FACULTAD DE CIENCIAS

MASTER DE ANÁLISIS DE DATOS PARA LA INTELIGENCIA DE NEGOCIOS

TRABAJO DE FIN DE MASTER:

Detección de icebergs en imágenes satelitales del Ártico: enfoque basado en redes neuronales convolucionales

Alicia Lojo Iglesias

Dirigido por:
Julia Lastra García
Jesús Alberto Alonso Nanclares

julio, 2023

Dedicado a
mi familia

Resumen

El Ártico enfrenta un preocupante proceso de deshielo irreversible debido al cambio climático, lo que ha creado una zona geográfica antes inaccesible, presentando nuevos desafíos y oportunidades para actividades humanas. En respuesta a esto, surge el proyecto de investigación europeo AI-ARC (*Artificial Intelligence based Virtual Control Room for the Arctic*) con el propósito de mejorar la capacidad situacional, toma de decisiones y comunicación en el entorno marítimo ártico mediante el uso de tecnologías de inteligencia artificial y entornos virtuales.

Como parte de AI-ARC, se desarrolla el presente trabajo con el objetivo de crear una tecnología que detecte automáticamente bloques de hielo en el océano ártico. Para ello, se emplean redes neuronales convolucionales, un algoritmo de aprendizaje profundo que simula el comportamiento del cerebro humano mediante capas de convolución para procesar y reconocer patrones en datos. Estas redes se entrenan con imágenes del satélite Sentinel-1 y el proceso se divide en dos fases: la segmentación de zonas de mar y la detección de icebergs en esas áreas marítimas.

El resultado del proyecto consiste en una tecnología que, al recibir una imagen como entrada, produce una imagen de salida con los bloques de hielo detectados y sus coordenadas, listas para ser enviadas a las navieras. Aunque los resultados obtenidos hasta ahora son prometedores, se están explorando y desarrollando nuevos modelos de segmentación que serán implementados con el fin de potenciar la detección de bloques de hielo.

Palabras clave: Inteligencia artificial, Aprendizaje automático, Aprendizaje profundo, Redes Neuronales Convolucionales, AI-ARC, Visión artificial, Satélite, Sentinel-1, Copernicus.

Abstract

The Arctic is facing a concerning and irreversible process of ice melting due to climate change, creating a geographically accessible region that presents new challenges and opportunities for human activities. In response to this, the European research project AI-ARC (Artificial Intelligence based Virtual Control Room for the Arctic) emerges with the purpose of enhancing situational awareness, decision-making, and communication in the Arctic maritime environment through the utilization of artificial intelligence technologies and virtual environments.

As part of AI-ARC, the current work aims to develop a technology capable of automatically detecting icebergs in the Arctic Ocean. To achieve this, convolutional neural networks are employed, a deep learning algorithm that emulates the human brain's behavior by using convolutional layers to process and recognize patterns in data. These networks are trained with satellite images from Sentinel-1, and the process is divided into two phases: the segmentation of sea zones and the detection of icebergs in these maritime areas.

The project's outcome consists of a technology that, upon receiving an image as input, produces an output image with the detected ice blocks and their corresponding coordinates, ready to be sent to the shipping companies. Although the obtained results thus far are promising, we are currently exploring and developing novel segmentation models to enhance the detection of ice blocks.

Keywords: Artificial Intelligence, Machine Learning, Deep Learning, Convolutional Neural Networks, AI-ARC, Computer Vision, Satellite, Sentinel-1, Copernicus.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

El Ártico constituye una región ubicada en el extremo septentrional del planeta Tierra, en las proximidades del polo norte. Su área geográfica está compuesta principalmente por el Océano Ártico, un vasto océano cubierto por una gruesa capa de hielo, junto con tierras que rodean este océano, en las que se incluyen partes de Rusia, Estados Unidos, Canadá, Noruega, Dinamarca, Suecia, Finlandia e Islandia.

La capa helada desempeña un papel crucial en el sistema climático global [20]; el hielo marino refleja la radiación solar que regresa al espacio, de forma que contribuye significativamente en el mantenimiento del equilibrio energético terrestre. No obstante, en las últimas décadas se ha ido observando un alarmante proceso irreversible de deshielo en la región, atribuido, en gran medida, al cambio climático. La descongelación gradual está dando lugar a la apertura de un espacio geográfico históricamente inaccesible, lo que plantea nuevos desafíos y oportunidades para el desarrollo de actividades humanas.

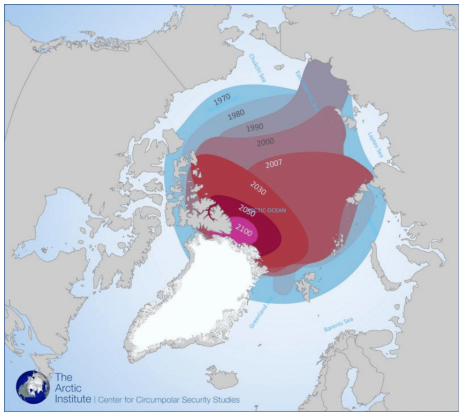

Figura 1.1: Estimación del deshielo ártico desde 1970 hasta 2100

1.1. Motivación

La creciente apertura del Ártico como resultado del deshielo ha generado cambios notables en las condiciones de navegación, incluyendo mareas y ubicación de bloques de hielo, lo que ha dado lugar a la creación de nuevas rutas marítimas.

Sin embargo, el tránsito marítimo en el Océano Ártico cuenta con algunas problemáticas que deben abordarse. Debido a su latitud norte, carece de una infraestructura adecuada de comunicaciones por satélite, así como de activos de búsqueda y rescate o áreas de refugio.

Estas limitaciones, sumadas a las aguas heladas, presencia de icebergs, patrones climáticos erráticos y una creciente población de navegantes inexpertos en la región ártica provoca que el Ártico sea extremadamente peligroso para la navegación, por lo que se ve necesario recurrir a otro tipo de tecnologías punteras que ayuden a conocer la nueva cartografía de la región.

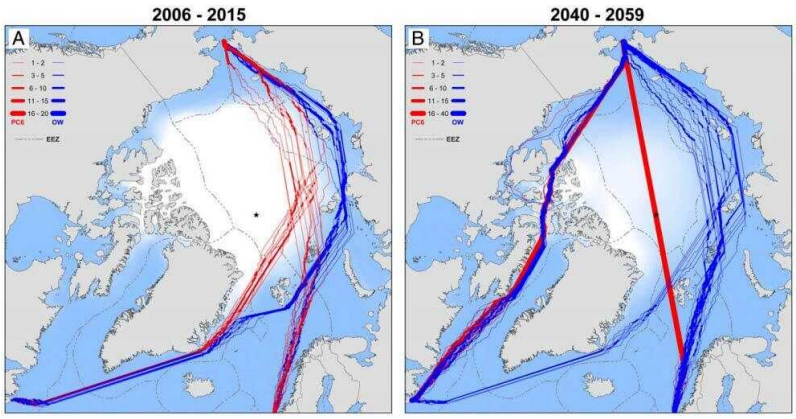

Figura 1.2: Cambios en las rutas marítimas del Ártico debido al deshielo

1.1.1. AI-ARC

En este contexto, emerge el proyecto de investigación europeo **AI-ARC** (*Artificial Intelligence based Virtual Control Room for the Arctic*), traducido como Sala de Control Virtual basada en Inteligencia Artificial para el Ártico. El proyecto AI-ARC tiene como objetivo primordial mejorar la conciencia situacional, la toma de decisiones y la comunicación en el entorno marítimo ártico mediante la aplicación de tecnologías de inteligencia artificial (IA) y entornos virtuales.

Se busca desarrollar una plataforma innovadora y fácil de usar, llamada ***Virtual Control Room*** (VCR), que integre servicios de IA para mejorar la seguridad y la eficiencia de las operaciones marítimas en el Ártico. La plataforma permitirá a los actores marítimos tener acceso a información relevante y actualizada, facilitando la coordinación y la toma de decisiones. Para lograr su desarrollo, el proyecto AI-ARC cuenta con la participación de un consorcio formado por 22 colaboradores de 12 países diferentes (Figura 1.3), entre ellos Tree Technologic empresa asturiana que hace posible la realización de el presente trabajo de fin de máster. Tree Technologic, dentro de las funciones asignadas para la consecución de los objetivos, desarrolla dos tecnologías: detección de anomalías en el movimiento de los buques y detección de bloques de hielo, tarea descrita en el presente trabajo.

Figura 1.3: Socios del proyecto AI-ARC

Parte del proyecto AI-ARC se basa en la utilización de datos y servicios proporcionados por Copernicus, al igual que los progresos incluidos en este trabajo. Copernicus [1] es un programa de observación de la Tierra desarrollado por la Unión Europea en colaboración con la Agencia Espacial Europea (ESA) y otros socios. Este programa tiene como objetivo proporcionar datos y servicios de observación de la Tierra gratuitos y accesibles. Se utiliza para abordar desafíos ambientales, mejorar la gestión de desastres y apoyar la toma de decisiones en diversos sectores. Dentro del programa Copernicus, la ESA tiene a su cargo la gestión de varios satélites, incluido el Sentinel-1. Este satélite desempeña un papel fundamental en este informe, ya que proporciona imágenes de alta resolución mediante observaciones de radar utilizadas para la consecución de los objetivos marcados (Sección 1.2).

[1] https://www.copernicus.eu/en/copernicus-and-free-open-source-software-community

AI-ARC, financiado por el programa de investigación e innovación Horizon 2020 de la Unión Europea bajo el Acuerdo de Subvención No. 101021271, tiene una duración de dos años. En octubre de 2023, se llevará a cabo una demostración de la Sala de Control Virtual (VCR) en los cuarteles generales de la guardia costera de Islandia en Reykjavik, donde se presentarán las tecnologías desarrolladas por cada uno de los participantes del consorcio, así como los avances logrados en el actual TFM. La presentación se realizará frente a guardacostas y marinas para que aprueben su funcionamiento en entornos marítimos reales. Esto permitirá que la solución sea utilizada por navieras que surcan aguas con condiciones similares.

Todos estos nuevos proyectos que surgen a nivel internacional son posibles debido a los notorios avances experimentados en la inteligencia artificial desde su surgimiento, en la década de los 50, hasta la actualidad. Este conjunto de tecnologías, descritas con mayor profundidad en la sección 2, surgen a raíz del interés de los investigadores por explorar la posibilidad de crear máquinas capaces de simular el comportamiento humano, sentando las bases de lo que hoy se conoce como IA.

1.1.2. Visión artificial

La visión artificial, subcampo de la inteligencia artificial, permite analizar imágenes y extraer conclusiones de ellas. En el contexto del trabajo, se dispone de un gran corpus de imágenes del satélite Sentinel-1 [2] para abordar el problema específico del Ártico sobre el reconocimiento y detección de masas de hielo o icebergs, por lo tanto, el uso de esta tecnología parece adecuado para contribuir a una toma de decisiones informada y precisa, que mejorará la seguridad de las operaciones marítimas, facilitando la planificación y ejecución de actividades en entornos desafiantes como el Ártico.

La visión por computador o visión artificial consiste en el empleo de técnicas que permiten reconocer, analizar y comprender automáticamente imágenes o vídeos. Esencialmente, se trata de enseñar a una computadora a "ver" tal y como lo haría un ser humano.

La visión artificial es una área muy estudiada dentro de la inteligencia artificial debido a sus numerosas aplicaciones, que pueden ir desde la mejora en el enfoque de cámaras fotográficas hasta la conducción autónoma.

[2] https://sentinels.copernicus.eu/web/sentinel/missions/sentinel-1

Dentro de ella se encuentra la detección de objetos, enfoque del proyecto. A pesar de que los humanos realizan esta tarea de manera innata, para un computador representa un desafío debido a la naturaleza compleja de las imágenes. La ubicación espacial y la comprensión semántica de alto nivel de los objetos presentes resultan difíciles de lograr únicamente a partir de la representación de píxeles de colores, que es lo que entiende un ordenador como imagen. Es por eso por lo que parece fundamental el uso de técnicas de aprendizaje profundo para el desarrollo de algoritmos los suficientemente robustos, en este caso se aplicará la visión artificial con **redes neuronales convolucionales**.

1.2. Objetivos

Por lo tanto, y siguiendo las líneas generales que motivan a la realización del proyecto, el principal objetivo de este trabajo es el desarrollo de un algoritmo que permita clasificar, a través de imágenes satelitales del Ártico, zonas navegables y no navegables para las navieras; sumándose la detección de bloques de hielo a esas zonas navegables, con el fin de lograr un sistema que facilite la toma de decisiones y seguridad para todos los actores marítimos.

Para la resolución y consecución del objetivo se deberán alcanzar otros objetivos específicos desarrollados a lo largo del estudio, como pueden ser la recolección de un conjunto de datos de imágenes con la suficiente diversidad y volumen para poder entrenar de forma correcta el algoritmo, así como el conocimiento, diseño y arquitectura del modelo de red neuronal convolucional que se empleará para mejorar el estado del arte en términos de exactitud y precisión.

Capítulo 2

Preliminares

En esta sección, se explorarán los conceptos clave relacionados con las redes neuronales en el contexto de la inteligencia artificial. Estas redes son la tecnología fundamental para alcanzar los objetivos establecidos.

Se comienza con una breve introducción histórica que proporciona el contexto necesario sobre el desarrollo y evolución de estos modelos a lo largo del tiempo. Seguidamente, la sección se centra en las características relevantes que se deben considerar al elegir la red neuronal más adecuada para el entrenamiento.

Contextualización histórica

La inteligencia artificial [12] ha sido uno de los mayores avances tecnológicos del siglo XXI. Su origen se remonta a la década de los años 50, cuando John McCarthy [13] acuña el término. En sus primeros años, los investigadores se centraron en la lógica y el razonamiento simbólico, desarrollando programas basados en reglas lógicas, como la lógica de primer orden, para la resolución de problemas simples.

La lógica de primer orden o de predicados, se trata de un sistema formal que permite describir relaciones entre objetos utilizando predicados y cuantificadores. A medida que la investigación avanzaba, se hizo evidente que este tipo de lógica tenía limitaciones para abordar problemas más complejos.

Sin embargo, no es hasta los años 80 cuando el enfoque del **aprendizaje automático** comienza a ganar popularidad. Este enfoque revolucionario permitió el desarrollo de algoritmos capaces de aprender de los datos sin la necesidad ser programados explícitamente. Áreas como el reconocimiento de voz y la visión por computadora tuvieron un progreso mayor gracias a esta tecnología.

Aunque las redes neuronales artificiales surgieron en los años 40 y 50, con desarrollos como el perceptrón, fue en la década de los 90 cuando se comenzaron a investigar algoritmos de **aprendizaje profundo** [8] basados en estas redes que se inspiran en el conocimiento que tienen los neurocientíficos acerca del funcionamiento del cerebro humano. Estos algoritmos exploraron nuevas posibilidades más allá de las redes neuronales simples, incorporando capas ocultas y funciones de activación.

La revolución del aprendizaje profundo se produjo a partir de 2013, impulsada por la convergencia de tres factores clave. En primer lugar, se produjeron mejoras significativas en el hardware informático, lo que permitió un procesamiento más rápido y eficiente. Además, la disponibilidad de conjuntos de datos masivos proporcionó los recursos necesarios para entrenar modelos más complejos. Por último, el desarrollo de software especializado, como Theano [1], TensorFlow (Sección 4.1) y PyTorch [2] , facilitó la implementación y el entrenamiento de redes neuronales profundas. Esta conjunción de avances tecnológicos posibilitó el entrenamiento de modelos con múltiples capas ocultas y llevó a mejoras sustanciales en tareas como el reconocimiento de patrones y la clasificación de imágenes.

A partir de este momento, la inteligencia artificial ha experimentado un progreso notable. Sus aplicaciones se han vuelto cada vez más comunes en la vida cotidiana, abarcando desde asistentes virtuales hasta conducción automática o sistemas de reconocimiento facial.

2.1. Aprendizaje automático

El *machine learning* o aprendizaje automático [4] es una rama de la inteligencia artificial que se enfoca en el desarrollo de algoritmos y modelos que permiten a las máquinas aprender y mejorar automáticamente a través de la experiencia. Estos algoritmos se entrenan con grandes conjuntos de datos y emplean técnicas estadísticas y matemáticas para la identificación de patrones y relaciones entre variables.

Los conceptos clave que lo forman son los algoritmos, secuencias de instrucciones que resuelven problemas matemáticos y se utilizan para realizar el análisis y la construcción de los modelos predictivos; los modelos, representaciones matemáticas de los patrones identificados en los datos y que se utilizan para la predicción y toma de decisiones; y los conjuntos de entrenamiento y test, datos que se emplean para

[1] https://pypi.org/project/Theano/
[2] https://pytorch.org/

entrenar y evaluar el modelo.

Se debe tener en cuenta que existen diferentes tipos de aprendizaje automático: aprendizaje supervisado, no supervisado y de refuerzo.

- **Aprendizaje supervisado**

  La principal característica del aprendizaje supervisado es que el modelo recibe el resultado deseado para cada objeto y lo utiliza para su entrenamiento. El modelo se ajusta a los datos de entrenamiento a través de un proceso de optimización de la **función de pérdida o coste**. La función de pérdida es una medida de la calidad de ajuste del modelo a los datos. El objetivo del proceso de optimización es minimizar esta función para que el modelo realice predicciones con una mayor precisión.

  Entre los algoritmos más comunes se encuentran la regresión lineal, regresión logística, árboles de decisión y máquinas de vector soporte (SVM).

- **Aprendizaje no supervisado**

  En el aprendizaje no supervisado, ocurre lo contrario; no se utilizan datos etiquetados para entrenar el modelo. El modelo busca patrones y estructuras en los datos que puedan ser útiles, es decir, el modelo aprende de patrones ocultos que utiliza para la agrupación en las predicciones.

  Entre las técnicas de aprendizaje no supervisado [6] más populares se encuentran el clustering, donde los datos similares son agrupados en clusters. Entre los que destacan los algoritmos de k-means, DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*) y de clustering jerárquico. Y la reducción de dimensionalidad, donde resuenan el PCA (Análisis de componentes principales) y el t-SNE. Esta técnica se utiliza para reducir la complejidad de los datos.

- **Aprendizaje por refuerzo**

  El aprendizaje por refuerzo [19] o *Reinforcement Learning* consiste en que el modelo aprende a tomar decisiones óptimas a través de la interacción con un entorno, es decir, debe aprender a través de la retroalimentación del entorno en forma de recompensas o castigos.

  Este enfoque se utiliza en aplicaciones como el control de robots, los juegos y la toma de decisiones automatizada. Algunos algoritmos populares en el aprendizaje por refuerzo incluyen Q-Learning, SARSA y Algoritmos de Monte Carlo.

Aunque las redes neuronales se pueden clasificar en el aprendizaje supervisado y no supervisado, las utilizadas en este trabajo se enmarcan dentro del aprendizaje supervisado. Estas técnicas serán detalladas con mayor profundidad en las siguientes secciones.

2.2. Redes Neuronales Artificiales

En el ámbito del aprendizaje automático, se encuentra una técnica llamada aprendizaje profundo o *deep learning* [3]. Se basa en el uso de estructuras lógicas que se inspiran en la organización del sistema nervioso de los mamíferos. Estas estructuras consisten en capas de unidades de proceso, también conocidas como neuronas artificiales, que se especializan en detectar características específicas presentes en los objetos percibidos.

El aprendizaje profundo se lleva a cabo mediante el uso de arquitecturas de redes neuronales artificiales, concretamente las conocidas como **redes neuronales profundas**. Estas redes están compuestas por múltiples capas de neuronas, incluyendo capas ocultas, lo que les permite capturar representaciones jerárquicas y complejas de los datos.

En una red neuronal profunda, las capas se organizan en una secuencia, comenzando con una capa de entrada que recibe los datos de entrada, seguida de una o varias capas ocultas y finalmente una capa de salida que produce las predicciones. Las capas ocultas, como su nombre lo indica, no están directamente conectadas con las entradas o salidas, y son responsables de capturar características y representaciones más abstractas de los datos a medida que se profundiza en la red. Estas capas permiten a la red neuronal aprender representaciones más complejas y realizar tareas más sofisticadas.

La capa más sencilla en una red neuronal es la capa de tipo *feed-forward*. Esta capa es comúnmente utilizada y consta de una colección de neuronas interconectadas y es la que se describe en esta sección.

La unidad básica de las redes neuronales es la neurona artificial, que se inspira en las neuronas biológicas del cerebro. Cada neurona recibe una serie de entradas ponderadas, representadas como x_n, y cada entrada tiene un peso correspondiente w_n. Durante el proceso de entrenamiento, estos pesos se ajustan de forma iterativa para minimizar el error y mejorar el rendimiento de la red. La salida de la neurona, representada como f, se calcula mediante la aplicación de una **función de activa-**

ción elegida a la suma ponderada de las entradas y los pesos.

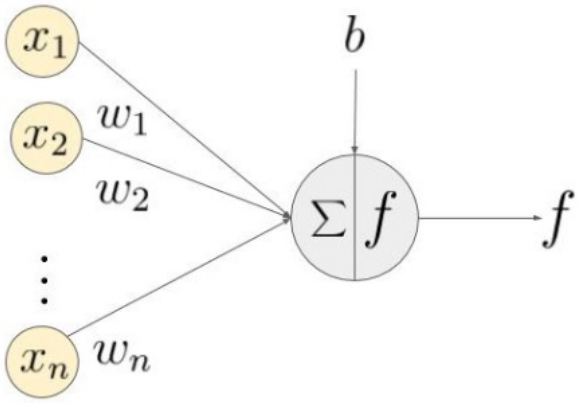

Figura 2.1: Esquema de una neurona

Las neuronas de cada capa se encuentran conectadas con todas las neuronas de la capa anterior y siguiente. En cada neurona, se realiza el producto escalar entre los pesos de las conexiones y las activaciones de las neuronas de la capa anterior. Luego, se aplica la función de activación al resultado obtenido. El sesgo o bias, representado por el término b, se suma a la suma ponderada antes de aplicar la función de activación.

La operación realizada por cada neurona se puede describir utilizando la siguiente fórmula:

$$f = g(b + \sum_i x_i w_i) \tag{2.1}$$

donde f es la activación de la neurona, g es la función de activación, b es el sesgo, x_i es el valor de la entrada i y w_i es el peso asociado a la conexión entre la neurona de la capa anterior y la neurona actual, tal y como se representa en la figura 2.1.

La función de activación tiene como objetivo introducir no linealidad en la red neuronal. Esto permite que la red pueda aprender relaciones y patrones no lineales en los datos. Algunas de las funciones de activación comúnmente utilizadas en redes neuronales profundas incluyen la función sigmoide, la función ReLU (*Rectified Linear Unit*) o la función softmax. Estas funciones cuentan con propiedades y características diferentes que las hacen adecuadas para los diferentes tipos de problemas de aprendizaje:

- La **función sigmoide o logística**, en la que los pesos de salida toman valores entre 0 y 1. Se define como:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

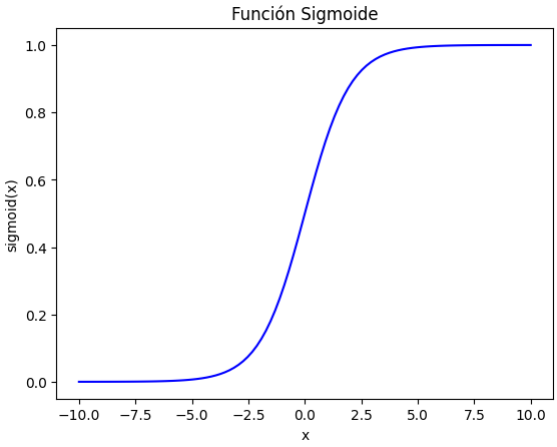

Figura 2.2: Función de activación sigmoide

Es ampliamente utilizada en capas de salida binarias, donde se busca una clasificación entre dos clases. Al mapear los valores de entrada entre 0 y 1 permite interpretar la salida como una probabilidad. Sin embargo, la función sigmoide tiende a saturarse en los extremos, lo que puede dificultar el entrenamiento y generar problemas de desvanecimiento del gradiente en redes neuronales profundas.

- La **función softmax** genera una distribución de probabilidades sobre las diferentes clases en un problema de clasificación multiclase.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \tag{2.3}$$

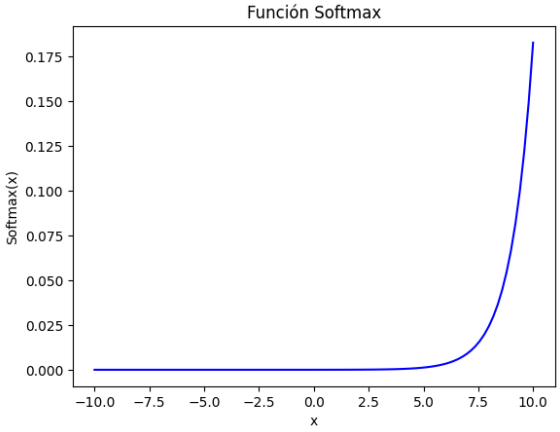

Figura 2.3: Función de activación softmax

Se utiliza típicamente en la capa de salida de redes neuronales para problemas de clasificación multiclase excluyentes.

- La **función ReLU**, que se define como:

$$\sigma(x) = \text{máx}(0, x) \tag{2.4}$$

Es importante tener en cuenta que la función ReLU no se expresa en términos de una expresión matemática continua y diferenciable, como la función sigmoide, sino que se define utilizando una operación de máximo entre 0 y el valor de entrada x.

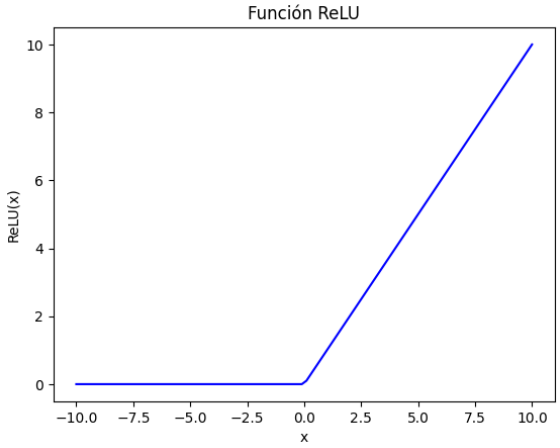

Figura 2.4: Función de activación ReLU

Se utiliza comúnmente en capas ocultas de redes neuronales profundas. Su principal ventaja es que es computacionalmente eficiente y evita el problema del desvanecimiento del gradiente. Es especialmente útil para capturar relaciones no lineales en los datos.

El proceso de entrenamiento de una red neuronal profunda implica ajustar los pesos y sesgos de las conexiones para minimizar el error entre las salidas predichas por la red y las salidas deseadas. Esto se logra mediante **algoritmos de optimización**, que ajustan gradualmente los pesos y sesgos en función de las señales de error propagadas hacia atrás a través de la red (***backpropagation***).

2.2.1. Backpropagation

En las redes neuronales, el proceso de propagación hacia adelante es esencial para el funcionamiento de la red. En esta etapa, los datos de entrada se transmiten a través de las capas de la red, desde la capa de entrada hasta la capa de salida, pasando por las capas ocultas intermedias.

Una vez que la información se propaga hacia adelante y alcanza la capa de salida, se obtiene una predicción o una salida estimada. Sin embargo, para entrenar adecuadamente la red neuronal y ajustar los pesos sinápticos, se necesita un proceso adicional llamado *backpropagation*, también conocido como retropropagación del error, es un algoritmo utilizado para calcular los gradientes de la función de coste con respecto a los parámetros de la red neuronal. Su objetivo principal es proporcionar los gradientes necesarios para ajustar los parámetros de la red de manera que se minimice la función de coste, consiguiéndose mediante métodos de optimización numérica.

Durante la propagación hacia adelante, se calcula la suma ponderada de las entradas de cada neurona en cada capa, incluyendo los sesgos, y se aplica una función de activación no lineal a este resultado. Esto permite generar las salidas de cada capa y propagar la información a través de la red. En la propagación hacia atrás, se calculan los gradientes de la función de coste con respecto a los parámetros de la red neuronal. Comenzando desde la última capa y utilizando la regla de la cadena, se calcula el gradiente de la función de coste con respecto a las salidas de cada capa. Luego, se calcula el gradiente de la función de coste con respecto a los pesos y sesgos de cada neurona en cada capa.

Este proceso de cálculo de gradientes se realiza de manera recursiva, utilizando los gradientes de la capa posterior para calcular los gradientes de la capa anterior. De esta manera, se aprovecha la información del error en la capa de salida para

ajustar los parámetros en las capas anteriores de la red.

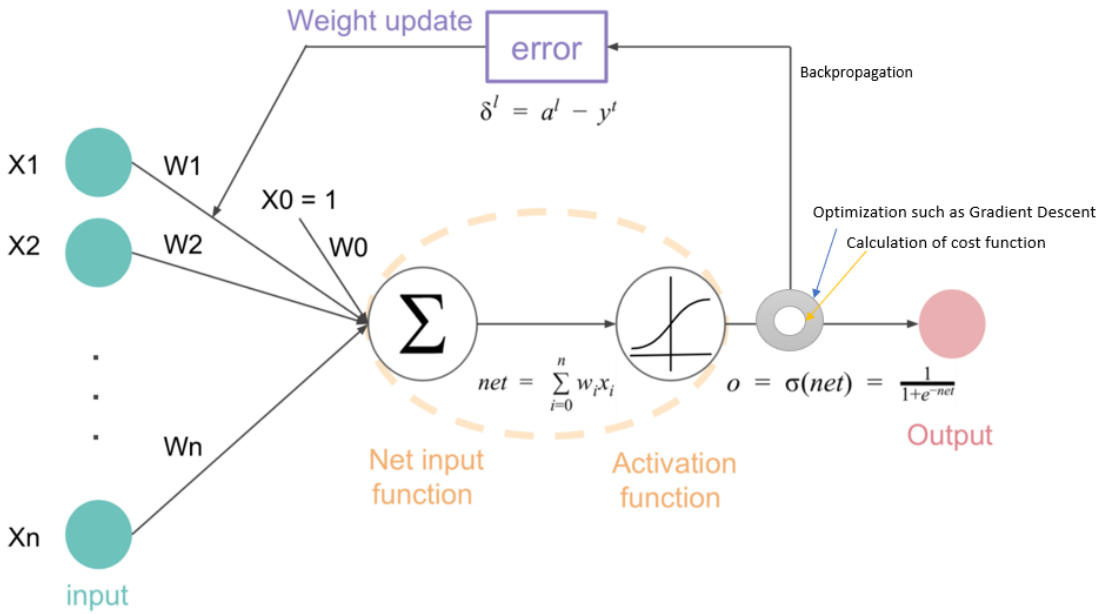

Figura 2.5: Esquema completo del proceso de aprendizaje de una red neuronal

En la actualización de los parámetros, se utiliza un algoritmo de optimización para ajustar los pesos y sesgos en la dirección opuesta a los gradientes obtenidos. Esto se realiza multiplicando los gradientes por la tasa de aprendizaje y restando estos valores de los pesos y sesgos actuales.

Entre los algoritmos de optimización mas recurrentes se encuentran el descenso de gradiente y el algoritmo Adam:

- **Descenso de gradiente**

  La actualización de parámetros se realiza en dirección opuesta al gradiente de la función de coste. Su formula general es:

$$\theta = \theta - \alpha \cdot \nabla J(\theta) \tag{2.5}$$

- **Algoritmo Adam** Combina el descenso de gradiente con una adaptación de la tasa de aprendizaje. Utiliza momentos acumulados para actualizar los parámetros.

El primer momento acumula la media de los gradientes a lo largo del tiempo.

$$m = \beta_1 \cdot m + (1 - \beta_1) \cdot \nabla J(\theta) \tag{2.6}$$

El segundo momento acumula la media de los cuadrados de los gradientes a lo largo del tiempo.

$$v = \beta_2 \cdot v + (1 - \beta_2) \cdot (\nabla J(\theta))^2 \tag{2.7}$$

Corrigen el sesgo inicial de los momentos del primer y segundo momento respectivamente. A medida que aumenta el número de iteraciones t, se reduce el sesgo inicial y se estabilizan los momentos acumulados.

$$\hat{m} = \frac{m}{1 - \beta_1^t} \tag{2.8}$$

$$\hat{v} = \frac{v}{1 - \beta_2^t} \tag{2.9}$$

Utilizando los momentos corregidos y la tasa de aprendizaje (α), se ajustan los parámetros de la red neuronal. La división por $\sqrt{\hat{v}} + \epsilon$ normaliza el gradiente según la variabilidad de los gradientes acumulados.

$$\theta = \theta - \alpha \cdot \frac{\hat{m}}{\sqrt{\hat{v}} + \epsilon} \tag{2.10}$$

Por otro lado, los **algoritmos de regularización** se utilizan para evitar el sobreajuste (***overfitting***) en el modelo. Estos algoritmos agregan términos adicionales a la función de coste con el propósito de penalizar los valores extremos de los parámetros. Esto ayuda también a controlar la complejidad del modelo, lo que puede mejorar el rendimiento en conjuntos de datos nuevos.

2.2.2. Overfitting y Underfitting

Uno de los mayores desafíos del *Machine Learning* es lograr una buena generalización en situaciones nuevas para el modelo. Para abordar este desafío, se divide el conjunto de datos disponible en dos conjuntos: **entrenamiento** y **test**. Durante el entrenamiento, se busca reducir el error en el conjunto de entrenamiento, y luego se evalúa el rendimiento del algoritmo en el conjunto de test, con un error que puede ser igual o mayor al del entrenamiento. El objetivo es minimizar el error de entrenamiento y mantener la diferencia entre los errores de ambos conjuntos lo más pequeña

posible, lo cual indica una buena generalización. En algunas ocasiones, el conjunto es dividido en tres. Se reserva sobre un 20 % de los datos de entrenamiento para crear un conjunto de **validación**, utilizado para realizar evaluaciones intermedias durante la construcción del modelo.

La capacidad del modelo se refiere a su capacidad para ajustarse a una variedad de funciones. Un modelo con alta capacidad puede capturar patrones más complejos que uno con baja capacidad. Sin embargo, a medida que aumenta la capacidad del modelo, la discrepancia entre los errores de entrenamiento y prueba tiende a aumentar, y esto disminuye a medida que se incrementa el número de datos. La capacidad del modelo se puede controlar de varias formas, como ajustar el número de parámetros en el modelo. En la práctica, se busca encontrar un equilibrio, considerando el principio de parsimonia, que implica seleccionar la hipótesis más simple entre aquellas que tienen la misma capacidad para explicar los datos.

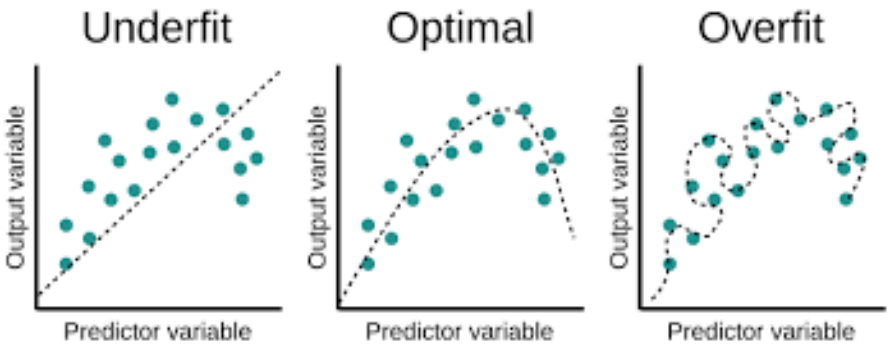

Figura 2.6: Overfitting y Underfitting

El subajuste o *underfitting* ocurre cuando un modelo es demasiado simple para capturar los patrones en los datos de entrenamiento, mientras que el sobreajuste u *overfitting* ocurre cuando el modelo es demasiado complejo y se ajusta al ruido en los datos en lugar de los patrones subyacentes. Se busca encontrar un equilibrio entre ambos, ya que aumentar la capacidad del modelo puede disminuir el error de entrenamiento, pero también puede aumentar la "brecha de generalización", es decir, la diferencia entre los errores de entrenamiento y test.

La regularización es una técnica utilizada para abordar el subajuste y el sobreajuste. Consiste en realizar modificaciones en el algoritmo de aprendizaje con el objetivo de reducir el error de generalización sin afectar significativamente el error de entrenamiento. Existen diferentes técnicas de regularización, y su elección depende

del caso de estudio. Además, se pueden tomar decisiones en el diseño del algoritmo para aplicar la regularización.

Entre las técnicas de regularización se ha decidido destacar dos de ellas. El **dropout**, donde durante el entrenamiento, se desactivan aleatoriamente un porcentaje de las neuronas en cada paso, lo que ayuda a evitar la dependencia excesiva de algunas unidades específicas y promueve una mayor generalización.

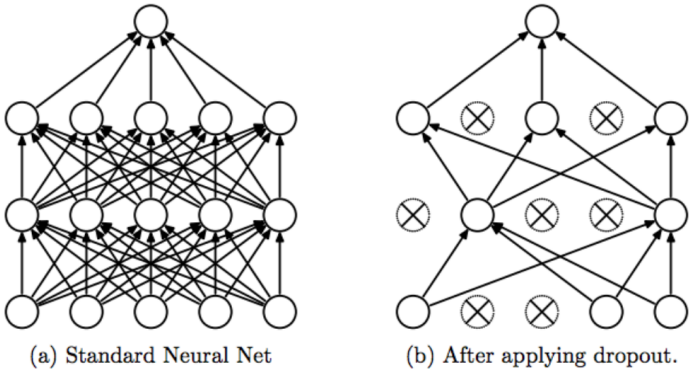

Figura 2.7: Técnica de regularización Dropout

Y la técnica de **Early Stopping**, que consiste en detener el entrenamiento del modelo cuando el rendimiento del conjunto de validación deja de mejorar. Esto ayuda a evitar el sobreajuste y permite seleccionar el modelo que mejor se generaliza antes de que ocurra el sobreajuste.

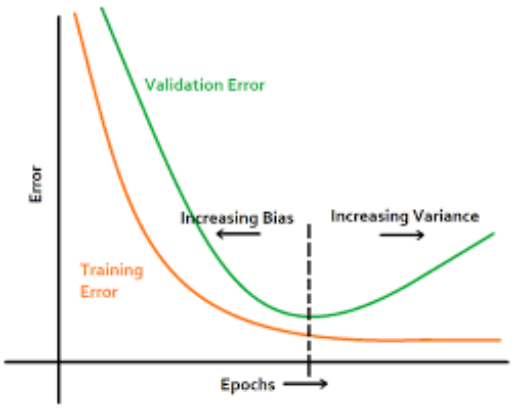

Figura 2.8: Técnica de regularización Early Stopping

2.2.3. Hiperpárametros

Los hiperparámetros son parámetros que no se determinan automáticamente durante el entrenamiento, sino que se ajustan externamente por parte del usuario. La elección de estos hiperparámetros influye en gran medida en el comportamiento y rendimiento del modelo. Para evitar el sobreajuste en el conjunto de entrenamiento, se crea el conjunto de validación, en el cual se pueden modificar los hiperparámetros y asegurar una buena generalización.

Existen dos enfoques para seleccionar los hiperparámetros en los algoritmos de aprendizaje profundo: la selección manual y la selección automática. La selección manual implica comprender cómo afectan los hiperparámetros al rendimiento del modelo y requiere experiencia en el dominio. En cambio, los enfoques automáticos buscan encontrar los mejores hiperparámetros de manera eficiente. Estos métodos exploran el espacio de hiperparámetros para encontrar la mejor combinación que maximice la calidad del modelo o minimice alguna métrica de evaluación.

Dos métodos comunes dentro de la selección automática son la búsqueda en cuadrícula (***grid search***) y la búsqueda aleatoria (***random search***). La búsqueda en cuadrícula prueba todas las combinaciones posibles de valores de los hiperparámetros, mientras que la búsqueda aleatoria muestrea valores de forma aleatoria.

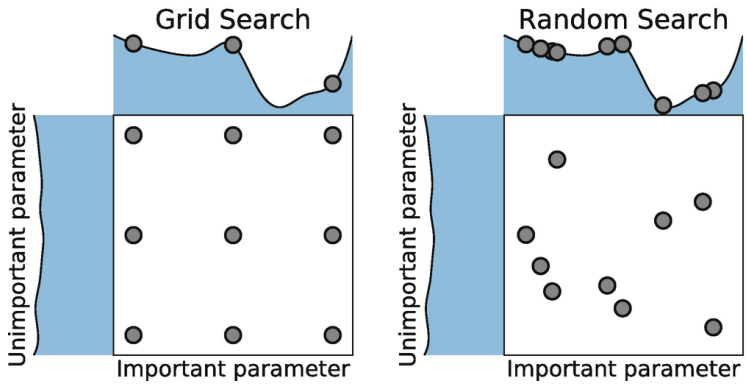

Figura 2.9: Grid Search y Random Search

Teniendo en cuenta lo anterior, la búsqueda en cuadrícula es adecuada cuando hay pocos hiperparámetros y se dispone de un conocimiento previo para seleccionar los valores a explorar. Sin embargo, su principal inconveniente es el alto costo computacional, que crece exponencialmente con el número de parámetros. Por otro lado, cuando se dispone de muchos hiperparámetros, la mejor opción es la búsqueda

aleatoria, ya que evita experimentos innecesarios y converge más rápidamente hacia buenos valores en los hiperparámetros. Además del número de hiperparámetros, también es importante considerar el conjunto de valores que se quiere probar para cada uno a la hora de elegir entre *grid search* y *random search*.

Capítulo 3

Redes neuronales convolucionales

Las redes neuronales convolucionales (CNN, del inglés *Convolutional Neural Network*) representan un tipo especializado de red neuronal, caracterizado por el uso de la **operación de convolución** en al menos una de sus capas. Esta operación puede ser utilizada para aprovechar simetrías y patrones locales, tanto sobre datos unidimensionales como en datos en forma de matriz bidimensional, lo que hace que este tipo de redes sea adecuado para el procesamiento y análisis de imágenes. No obstante, su campo de aplicación también incluye el procesamiento de datos secuenciales como pueden ser los correspondientes a audio, texto, series temporales o señales biomédicas como el electrocardiograma.

Cabe mencionar que la operación de convolución encuentra múltiples usos fuera del ámbito de las redes neuronales. Dejando a un lado aplicaciones en matemáticas puras o física teórica, esta es empleada en teoría de probabilidad o resolución de ecuaciones diferenciales lineales, además de estar estrechamente relacionada con la transformada de Fourier. Así es que ha sido usada para el filtrado de frecuencias en señales analógicas, como en circuitos electrónicos o acústica. Además, se ha empleado para describir fenómenos ópticos, interpretar señales de radar y para procesar señales en telecomunicaciones mediante filtros digitales.

En este capítulo se presentan las redes CNN. Se comienza con unas notas históricas, seguidas de un tratamiento formal de la operación de convolución, para después presentar la versión aplicada en la práctica junto a sus ventajas de uso. A continuación, se discute la capa convolucional y aspectos relevantes, la capa de ***pooling*** y el ***data augmentation***, un tipo de regularización específica a nuestra aplicación de este tipo de red. Finalmente, se exploran adaptaciones computacionales empleadas para aplicar estas redes en la práctica.

Contextualización histórica

En 1989, Yann LeCun introdujo *LeNet-5* [9], la primera red neuronal convolucional que tuvo un impacto significativo. LeNet-5 fue desarrollada para reconocer dígitos escritos a mano en cheques bancarios, marcando así un primer paso importante en la aplicación de las CNN para el reconocimiento de patrones.

Durante los años 90 y 2000, las CNN continuaron avanzando, surgiendo nuevas arquitecturas y técnicas. Sin embargo, el verdadero impulso para las CNN llegó en 2012 con el desarrollo de *AlexNet* [7] por parte de Alex Krizhevsky, Geoffrey Hinton e Ilya Sutskever. Este modelo revolucionario participó en el desafío *ImageNet Large Scale Visual Recognition Competition (ILSVRC)* [17], superando notablemente a los modelos anteriores en términos de precisión y rendimiento. AlexNet introdujo una arquitectura más profunda y técnicas innovadoras, como el **batch normalization** y la **función de activación ReLU**, lo que marcó un hito clave en la historia de las CNN.

Después del éxito de AlexNet, se produjo un rápido desarrollo de nuevas arquitecturas de CNN. Modelos como *VGGNet*, *GoogLeNet (Inception)*, *ResNet* y *DenseNet* surgieron, cada uno aportando mejoras en términos de profundidad, capacidad de aprendizaje y rendimiento en tareas de visión por computadora. Estos modelos se beneficiaron del aumento de datos disponibles y del crecimiento en el poder de cálculo, consolidando aún más el papel de las CNN en el campo del aprendizaje profundo.

Es importante destacar que, en la actualidad, han surgido modelos más avanzados en áreas específicas. Por ejemplo, en la detección de objetos, los modelos más punteros incluyen *YOLO* (*You Only Look Once*) [15] y *GroundingDino* [14]. Para la segmentación, el modelo SAM(*Segment Anything Model*) [21] de Meta ha mostrado excelentes resultados. En cuanto a la generación de imágenes, modelos como *Stable Diffusion* [10] y *DALL-E* [16] han logrado avances significativos en el campo de la inteligencia artificial generativa, rama de la inteligencia artificial que se enfoca en la generación de contenido original a partir de datos existentes.

Estos modelos de vanguardia representan el estado actual de la investigación en sus respectivas áreas, y continúan impulsando el campo de las CNN hacia nuevas fronteras. Las CNN han encontrado aplicaciones en diversos campos, como la medicina, la robótica, la conducción autónoma y la realidad aumentada, demostrando su versatilidad y capacidad para abordar problemas complejos.

3.1. Operación de convolución

La convolución es una operación matemática que transforma dos funciones f y g en una tercera función s, que representa la superposición de ambas después de invertir y trasladar la función g. La convolución de las funciones f y g se denota como:

$$s(t) = (f * g)(t) \tag{3.1}$$

y se define como la integral del producto de ambas funciones después de desplazar una de ellas a una distancia t.

$$s(t) = (f * g)(t) = \int_{-\infty}^{\infty} f(\eta) g(t - \eta) d\eta \tag{3.2}$$

Para funciones discretas, también se puede emplear una versión discreta de la convolución:

$$s(t) = (f * g)(m) = \sum_{n=-\infty}^{\infty} f(n) g(m - n) \tag{3.3}$$

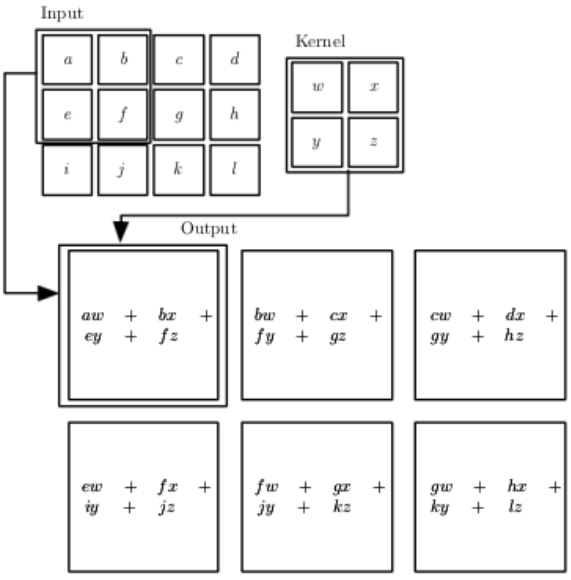

Figura 3.1: Representación esquemática de la operación de convolución en una CNN.

En la figura 3.1, un kernel de 2x2 se aplica sobre una imagen 3x4, produciendo otra imagen 2x3 al calcular para cada píxel del output la suma de las multiplicaciones elemento a elemento de las matrices de input y de kernel.

En el caso de las redes CNN, la primera función corresponderá a una matriz multidimensional o tensor de datos, como podría ser una imagen multicanal. La segunda función será un tensor cuyos elementos son parámetros, y que será llamado **kernel**. Dado que ambas serán finitas, se pueden ver estos tensores como funciones que son nulas en casi todo punto, excepto en un número finito de posiciones. Así, por ejemplo, si se tuviera una imagen bidimensional de un solo canal (en la que cada posición correspondería a un píxel recogiendo un valor de intensidad), se le aplicaría un kernel de dos dimensiones, y se tendría la siguiente expresión:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{3.4}$$

Y como la convolución es una operación conmutativa, también se puede expresar:

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{3.5}$$

con los sumatorios recorriendo las posibles posiciones del kernel, conjunto finito de índices. Para una posición (i,j) de la imagen inicial, la operación consiste en la multiplicación de cada elemento del kernel por su correspondiente en la entrada, sumándose después estos productos. Así se calcula el valor de una posición de la salida, y al recorrer todos los puntos de la imagen inicial se obtiene lo que se conoce como un **mapa de características**. Este proceso se representa en la Figura 3.1, donde se aprecia cómo el kernel actúa como una ventana deslizante que se desplaza sobre la imagen. Eso sí, conviene aclarar lo siguiente: en la definición anterior de $S(i,j)$, las posiciones del kernel están invertidas con respecto a las de los elementos de la imagen. En la práctica, se deshace esta inversión del kernel para definir una operación similar llamada correlación cruzada (la representada en la figura mencionada), cuya definición es:

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{3.6}$$

La mayoría de paquetes o librerías de *deep learning* utiliza esta operación, que es informalmente llamada convolución. De todas maneras, independientemente de esta distinción, una red neuronal aprenderá los valores adecuados de los parámetros del kernel en su posición correcta.

3.1.1. Ventajas de uso

Estas son algunas de las ventajas de usar capas de convolución en redes neuronales:

- **Interacciones dispersas**

  Las redes neuronales convolucionales aprovechan la estructura local inherente de la entrada para reducir el número de conexiones utilizadas al procesar la información. En comparación con la capa típica de una red neuronal, en la que cada unidad de salida está conectada a cada unidad de entrada a la capa, las capas de convolución restringen estas interacciones a pequeñas regiones del mapa de activaciones de entrada. De todas formas, una neurona en una capa lo suficientemente profunda tendrá un campo receptivo lo suficientemente grande como para abarcar todo el input de la red (imaginando una neurona que estuviera conectada con 8 de la capa anterior, y cada una de estas otras 8, uno se puede hacer una idea de cómo aumenta el campo receptivo con el número de capas).

- **Compartición de parámetros**

  Al contrario que en una capa de red neuronal tradicional, donde cada neurona tiene sus propios parámetros para producir una activación, en las capas convolucionales se tendrá un conjunto pequeño de parámetros que se aplica a cada posición de la entrada para producir un mapa de activaciones en la siguiente capa. Este conjunto de parámetros, el cual se menciona en el apartado 3.1, se denomina kernel, que será un filtro aplicado en una operación de convolución con una imagen o mapa de características (una capa puede tener varios kernels). Esto conlleva una reducción significativa en el número de parámetros, que no solo implica que el modelo sea más eficiente en términos de cálculo y memoria, sino también se reduce el riesgo de sobreajuste.

- **Representaciones equivariantes**

  La equivariancia es la capacidad de producir, bajo cierta transformación en la entrada, una salida transformada de la misma manera. La convolución presenta la propiedad de equivariancia bajo traslaciones, lo que permite al modelo CNN aprender a capturar características relevantes independientemente de su posición en la imagen de entrada. Sin embargo, la operación de convolución no es equivariante a otras transformaciones como pueden ser los cambios de escala o rotaciones de la imagen. Para solucionar esto, se emplean otras herramientas como el *pooling* o estrategias de regularización como el ***data augmentation***, ambos descritos más adelante.

3.2. Consideraciones prácticas

En el contexto del uso de redes CNN sobre imágenes, la convolución se puede entender como la aplicación de un filtro lineal. De la misma forma que en el procesamiento de imágenes tradicional, esta acción resaltará ciertas características específicas del mapa de entrada como pueden ser bordes, texturas o formas. Una capa convolucional está formada por múltiples filtros operando en paralelo, obteniéndose como output un mapa de características por filtro, en el que se recogen las activaciones correspondientes a la matriz de parámetros compartidos que constituye el kernel. La red neuronal irá aprendiendo durante el entrenamiento a resaltar patrones relevantes en la capa, que serán utilizados en etapas posteriores, ya sea para realizar tareas como clasificación o segmentación, o ya sea para servir como input para otras capas convolucionales. En el caso de concatenar varias capas convolucionales, como se puede ver en la figura 3.2, primeramente se aprenden características locales, como bordes horizontales, verticales y diagonales; en una segunda aplicación utilizando estas características locales como base, la combinación de estos bordes permite detectar un ojo, una boca, una ceja..., y posteriormente, es capaz de reconocer objetos, en este caso caras completas.

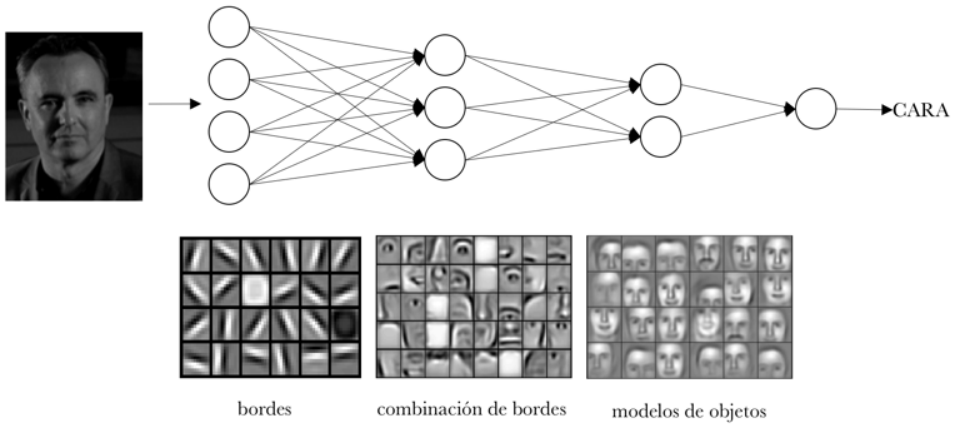

Figura 3.2: Ejemplificación de las fases de una red neuronal de clasificación de imágenes.

Se debe tener en cuenta que, a diferencia de la definición formal presentada para la convolución, esta operación no se suele realizar sobre una imagen bidimensional con un solo canal, sino que en la mayoría de los casos los canales son de color (RGB, por ejemplo). Es decir, la convolución suele ser aplicada sobre mapas de características multicanal, como puede ser un parche del cielo con canales correspondientes a diferentes frecuencias, o una "imagen" compuesta por múltiples mapas

de características apilados. Por lo tanto, los filtros utilizados no son matrices bidimensionales (por ejemplo, 3x3 píxeles), sino que también tienen una dimensión adicional que está vinculada al número de canales del input de la capa de convolución. Estos filtros actuarán como una ventana deslizante, moviéndose solamente en las dimensiones de anchura y altura, a diferencia de lo que se conoce como convolución 3D, que también se desplaza a lo largo de la dimensión de canal, produciendo un resultado tridimensional por filtro.

Otro elemento importante en la implementación de la operación de convolución es el parámetro ***stride***, que indica el número de posiciones (o píxeles) que se desplaza el kernel entre operaciones de convolución. Un valor de *stride* mayor que 1 implica un salto más grande entre cada aplicación del kernel, lo que resultará en una reducción de la resolución espacial del mapa de salida. De esta manera se realizaría un muestreo más "agresivo", capturando características generales en lugar de detalles finos, lo que puede ser útil para reducir el sobreajuste o el coste computacional, que puede ser beneficioso según las necesidades del modelo en cuestión.

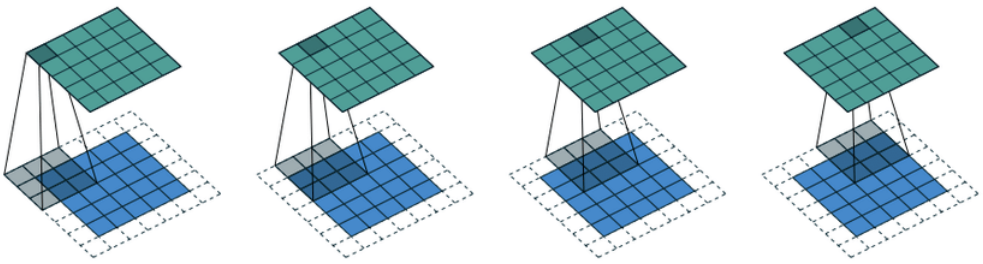

Figura 3.3: Representación esquemática de la operación de convolución en la que se aplica un filtro 3x3 con stride 1 sobre un mapa de entrada 5x5 con suficiente *zero-padding* como para obtener un mapa de salida 5x5

Finalmente, en la práctica también debe de ser considerado el concepto de ***zero-padding***. Este consiste en agregar ceros alrededor de los bordes de una imagen o mapa de características antes de aplicar la convolución, con el fin de controlar el tamaño del tensor de salida. Se distinguen tres tipos: *valid*, en el que no se utiliza el rellenado de ceros y el kernel sólo puede visitar posiciones completamente contenidas en el input (se reduce el tamaño de la salida); *same*, en el que se agrega el suficiente relleno de ceros para mantener el tamaño del tensor de salida igual al tamaño del tensor de entrada; y *full*, en el que se añaden ceros en todos los bordes de una imagen o mapa de características hasta que el tamaño del output sea el deseado.

3.2.1. Pooling

En las redes CNN, es una práctica común aplicar una función de activación no lineal a cada una de las activaciones lineales obtenidas después de una convolución. A continuación, se suele realizar una operación de ***pooling*** en el mapa resultante.

El *pooling* es una operación utilizada para resumir la información de un mapa de características. Se agrupan las salidas de la capa de entrada en regiones cuadradas (o rectangulares, según convenga) y se define una estadística resumen de las regiones. De nuevo, según se elija el tamaño de la ventana y el espaciado, cambiará el tamaño del mapa resultado, lo que puede ser útil desde el punto de vista de la eficiencia estadística o computacional de la red. Por ejemplo, si la siguiente capa es una capa convencional o completamente conectada será útil emplear el *pooling* para submuestrear y así utilizar un menor número de parámetros. Además, el empleo de esta técnica asegurará invarianza a pequeñas traslaciones de la entrada, aumentando la robustez de las predicciones de la red (desde el punto de vista bayesiano: se expresará una preferencia fuerte porque la red neuronal presente la propiedad de obtener la misma salida independientemente de una pequeña traslación en la entrada).

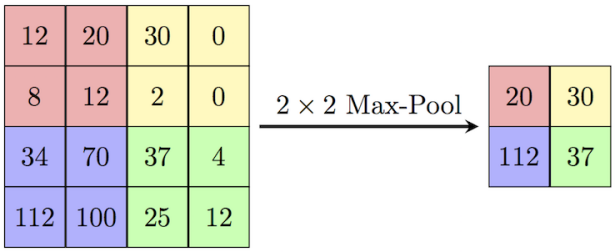

Figura 3.4: Representación esquematica del *max-pooling* aplicando un *stride* igual a 2

Dentro de las estadísticas resumen, es mayormente utilizada la de ***max pooling***, que toma como resultado el máximo de cada región seleccionada. Otras funciones de *pooling* incluyen el promedio de la región, la norma L^2 o un promedio ponderado en base a la distancia al píxel central.

3.2.2. Regularización

Uno de los grandes inconvenientes consiste en la dificultad que supone la construcción y el etiquetado de un conjunto de datos de calidad. En el contexto de la aplicación de redes CNN para el tratamiento de imágenes, existe un método de regularización comúnmente utilizado: el aumento de datos (***data augmentation***).

La idea detrás del aumento de datos es aumentar la diversidad y la cantidad de datos de entrenamiento sin recopilar nuevas imágenes. Esto se logra aplicando una serie de transformaciones a las imágenes originales, o bien geométricas como rotaciones, traslaciones, cambios de escala, recortes y volteos, o bien relativas a cambios en el contraste, saturación, tono o iluminación de las imágenes. Estas transformaciones introducen variabilidad y diversidad en los datos, lo que permite que la red CNN aprenda características más robustas e invariantes a las variaciones aplicadas, mejorando así la capacidad de generalización del modelo.

Figura 3.5: Ejemplo de aplicación de *data augmentation* sobre una imagen

3.2.3. Adaptaciones computacionales

Las redes CNN son computacionalmente intensivas y requieren un gran poder de procesamiento. Para acelerar su ejecución, se utilizan técnicas de paralelismo a nivel de hardware, como el uso de unidades de procesamiento gráfico (GPU) y unidades de procesamiento tensorial (TPU) que pueden realizar múltiples cálculos (como convoluciones) en paralelo. Estos dispositivos especializados se han convertido en componentes clave para entrenar y ejecutar estas redes de manera eficiente.

También existen diferentes enfoques utilizados para mejorar el rendimiento y eficiencia de las redes CNN. Se encuentran con adaptaciones de estas redes neuronales diseñadas para tener un menor coste computacional y, por tanto, un menor tiempo de inferencia, llegando a analizar imágenes en milisegundos. Otra posible ventaja de estos enfoques es la implementación en dispositivos móviles y sistemas embebidos sin comprometer excesivamente la calidad de los resultados. Por un lado, aparecen ciertas arquitecturas especialmente ligeras, como en el caso de *MobileNet* y *EfficientNet*, con un número reducido de parámetros. Por otro lado, se presenta la compresión de redes, que consigue un aumento notable en la capacidad de computación requerida mediante eliminación de conexiones redundantes o cuantización de los pesos. Esta técnica consiste en una reducción de la precisión numérica (de 32 bits a 8 bits por ejemplo) en los valores de los parámetros de la red, y puede ser aplicada o bien a posteriori o bien ser exigida durante el propio entrenamiento.

Otra forma de agilizar el trabajo con redes CNN es el aceleramiento del proceso de entrenamiento mediante el uso de redes preentrenadas. En lugar de entrenar un modelo desde cero, se puede aprovechar una red CNN previamente entrenada en grandes bases de datos para resolver problemas similares. Es denominado como ***fine-tuning*** y permite ajustar la red preentrenada a una tarea específica, lo que acelera significativamente el proceso al requerir menos epochs de entrenamiento y un conjunto de datos más reducido. Al utilizar el conocimiento previo aprendido por la red en la tarea original, se logra una adaptación más rápida y en muchos casos más eficiente a la nueva tarea.

Capítulo 4

Descripción del problema

En los apartados anteriores se ha expuesto la motivación detrás del presente proyecto. En esta sección, se presentará una descripción detallada del problema que se acomete en este trabajo, comenzando por el planteamiento del desarrollo. A continuación, se describirá el conjunto de datos utilizado, así como la estimación de los parámetros relevantes. Posteriormente, se discutirá el diseño de las redes y la evaluación de los resultados obtenidos, seguido de la implementación práctica del proyecto. Finalmente, se explorará la etapa de postprocesamiento, con el objetivo de mejorar la calidad de los resultados obtenidos.

4.1. Planteamiento

Para abordar el desafío planteado acerca de la detección de icebergs en territorio ártico a través de imágenes satelitales, se decide proponer un enfoque que involucra dos etapas principales: segmentación de las zonas marítimas y señalización de las detecciones de icebergs. En este caso, se plantea el uso de redes neuronales convolucionales para ambos procesos.

El propósito radica en la detección de bloques de hielo en las imágenes generadas por el satélite Sentinel-1, satélite que captura imágenes por radar (SAR, *Synthetic Aperture Radar)*, en vez de imágenes ópticas, como sí que hace el satélite Sentinel-2. Sentinel-1 proporciona imágenes de alta resolución de la superficie terrestre, independientemente de las condiciones meteorológicas o de iluminación. Por consiguiente, estas imágenes no solo muestran la superficie marina, sino que también puede aparecer superficie terrestre o banquisa, capas de hielo marino que se forma en las regiones polares.

Teniendo claro este punto, parece lógico que solo se deseen detectar los témpa-

nos de hielo que se encuentran en el mar, ya que son los que pueden dificultar la navegación de los buques y tienen mayor riesgo, por ser menos visibles. Por eso, una primera fase trata de la extracción de las zonas marinas dentro de las imágenes, lo que en el proyecto es denominado como segmentación. Al final de la primera etapa, se tendría una red entrenada capaz de distinguir categóricamente entre zonas de tierra, zonas de mar y zonas de banquisa.

Una vez que se ha realizado la segmentación del entorno marítimo, se puede pasar a la detección y señalización de los bloques de hielo, es decir, identificar y resaltar las áreas que contienen icebergs, entrenando una segunda red neuronal. En este caso, se trataría de un problema de clasificación binaria, en la que la red es capaz de distinguir entre la presencia o no de hielo.

Para finalizar, se realiza una fase de refinamiento denominada post-procesamiento, que consiste en la agrupación de ventanas, de forma que cada una de ellas se corresponda con un bloque de hielo.

En los siguientes apartados que forman la presente sección, se describirá de forma detallada cada una de las fases, pasando por la descripción de cada uno de los conjuntos de datos que se utilizan para el entrenamiento de los modelos, la descripción del diseño de la red neuronal y la evaluación de los resultados.

Para la realización de todo este procedimiento se emplean las librerías Keras [1] y TensorFlow [2]. TensorFlow es una librería de calculo tensorial, utilizada para definir modelos de redes neuronales, ya que se definen en su mayoría como operaciones entre tensores. Keras, incluida dentro de TensorFlow a partir de la versión 2, proporciona implementaciones de alto nivel de los modelos más utilizados, como pueden ser capas, neuronas o funciones de activación.

4.2. Conjunto de datos

Se cuenta con dos conjuntos de datos independientes para el entrenamiento de los modelos, cada uno de ellos esta formado por trozos de imágenes Sentinel-1. Cada uno de estos trozos, que serán denominados como *patches*, son figuras de dimensiones 75x75 en formato ".json".

Para la segmentación, el conjunto de datos consta de 3000 imágenes debidamente

[1] https://keras.io/
[2] https://www.tensorflow.org/

etiquetadas como tierra, mar y banquisa. Es importante destacar que se trata de una muestra balanceada, ya que se cuenta con 1000 elementos para cada una de las clases mencionadas. Este conjunto de datos ha sido creado de forma semiautomática a través de la generación de patches aleatorios en las imágenes previamente descargadas y siendo clasificados cada uno de ellos mediante un proceso manual.

En el caso de los datos para la detección, se cuenta también con una muestra equilibrada formada por 2404 imágenes. En este caso, 851 están identificadas como iceberg, 753 como buques y 800 como "nada", imágenes que solo contienen mar. Para la consecución de este conjunto de datos, se utilizaron datos proporcionados por Statoil [3], empresa energética internacional, en el que estaban etiquetados elementos clasificados como hielo y buque, y al que se vio necesario añadir 800 datos de patches clasificados como nada.

En cada una de las colecciones se distribuirán los datos en tres grupos, garantizando que las clases siguen balanceadas: entrenamiento, formado por el 72 % de la muestra, validación, el 18 % y test, el 10 % restante. El conjunto de entrenamiento (***train set***), se utiliza para el entrenamiento y aprendizaje del modelo, el de validación (***validation set***), para el ajuste de los hiperparámetros, y el de prueba (***test set***), que se utiliza para evaluar el modelo.

4.3. Estimación de los hiperparámetros

Tal y como se ha explicado en la subsección 2.2.3 la estimación de los hiperparámetros juega un papel crucial en el proceso de ajuste y optimización de los modelos de aprendizaje automático. Son variables que afectan al comportamiento y rendimiento del modelo.

Para la resolución, tanto del problema de la primera fase como el de la segunda fase, se ha decidido emplear técnicas de estimación hiperparamétrica automática. Tras una documentación previa acerca de las técnicas existentes, algunas resumidas en la sección mencionada en el anterior párrafo, y observando las características de los datos, se ha decidido aplicar un *random search*, debido a la carga computacional que se generaba con la utilización de la búsqueda en cuadrícula.

Para ambos modelos se busca lo siguiente:

- El número de capas para el modelo.

[3] https://www.kaggle.com/c/statoil-iceberg-classifier-challenge

- El tamaño del kernel utilizado en las capas convolucionales.
- La longitud del paso en las operaciones de convolución.
- El tamaño de la ventana utilizada en las operaciones de *pooling*.
- La tasa de aprendizaje utilizada en el proceso de optimización del modelo.
- El tamaño del lote utilizado durante el entrenamiento del modelo.

Tanto para la parte de búsqueda de hiperparámetros como para el entrenamiento del modelo se utiliza la técnica de regularización de *Early Stopping*, que detiene el entrenamiento del modelo si no se observa una mejora en la función de pérdida durante un número determinado de épocas consecutivas.

4.4. Diseño de la red neuronal convolucional

Obtenidos los resultados del proceso de estimación hiperparamétrica, se pasa a la fase de diseño de la red neuronal. En esta sección, debido a ciertas diferencias entre la estructura de las redes, se ha decidido hablar por separado del modelo de cada fase.

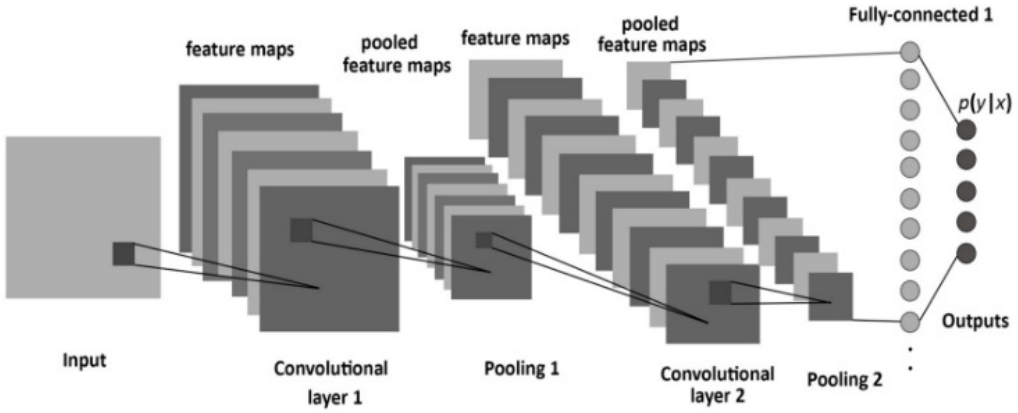

Figura 4.1: Capas de una Red Neuronal Convolucional

Segmentación del entorno marítimo

Para la red neuronal del problema de segmentación tenemos un problema de clasificación multiclase (mar, tierra y banquisa), por lo que la función de activación

que se utilizará en la capa de salida será softmax, que produce probabilidades para cada clase de salida.

La estructura de la red se puede describir de la siguiente manera:

- **Capa de convolución**. Aplica 32 filtros a la entrada, cada filtro tiene un tamaño de 7x7 y se mueve por la imagen con un paso de 1x1. La función de activación utilizada es ReLU. La entrada de esta capa es una imagen en escala de grises de tamaño 75x75.
- ***Max-Pooling***. Se le aplica con una ventana de tamaño 2x2. Esta capa reduce la dimensionalidad de los mapas de características obtenidos en la capa anterior.
- **Repetición de capa de convolución y *pooling***. El patrón de una capa convolucional seguida de una capa de *pooling* se repite dos veces más. La segunda capa convolucional utiliza 64 filtros y el mismo tamaño de kernel. La tercera capa convolucional utiliza 128 filtros.
- **Capa de aplanamiento**. Esta capa transforma la salida de la capa anterior en un vector unidimensional, preparándolo para la capa completamente conectada.
- **Capa densamente conectada**. Se agrega a continuación una capa densamente conectada con 64 neuronas. La función de activación utilizada en esta capa es la ReLU.
- **Capa densamente conectada de salida**. Como se mencionó al comienzo, se utilizó la función de activación softmax con 3 neuronas.

A la hora de optimizar la red para ajustar sus pesos durante el entrenamiento, minimizando la función de perdida, se utiliza el optimizador Adam, con una tasa de aprendizaje del 0.0001.

Detección de los bloques de hielo

En el caso de la fase de detección de hielo, se trata de un problema de clasificación binaria, en el cual se representan las variables categóricas como vectores numéricos binarios, en la que 1 es la presencia de hielo y 0 la no presencia. Debido a esto, la capa densamente conectada de salida utilizará la función sigmoidal, que genera probabilidades independientes para las dos clases posibles.

En cuanto a la estructura elegida:

- **Capa de convolución**. Aplica 32 filtros a la entrada, un tamaño de kernel de 3x3 y un paso de 1x1. La función de activación utilizada es ReLU. La capa toma una entrada con una forma de (75, 75, 1), que representa una imagen en escala de grises.

- ***Max-Pooling***. Se le aplica con una ventana de tamaño 2x2.

- **Repetición de capa de convolución y *pooling***. Se repite dos veces más. La segunda capa convolucional utiliza 64 filtros y el mismo tamaño de kernel, y la tercera 128 filtros.

- **Capa de aplanamiento**.

- **Capa densamente conectada**. Función de activación ReLU con 64 neuronas.

- **Capa densamente conectada de salida**. Función de activación sigmoidal con 2 neuronas.

4.5. Evaluación de los resultados

La función de pérdida es una medida utilizada para evaluar la discrepancia entre las predicciones del modelo y los valores reales en un problema de aprendizaje automático. Su objetivo es minimizarse, lo que indica que el modelo está haciendo predicciones más precisas.

En el caso de la clasificación, una función de pérdida comúnmente utilizada es la entropía cruzada (***cross-entropy***). El *cross-entropy* cuantifica la diferencia entre la distribución de probabilidad predicha por el modelo y la distribución real de las clases. Se utiliza ***categorical cross-entropy*** para la clasificación multiclase por lo tanto, es la que se utiliza en la primera red; y ***binary cross-entropy*** para la clasificación binaria, utilizada en el segundo modelo.

No obstante, aunque la función de pérdida es crucial para el proceso de entrenamiento, es igualmente importante contar con herramientas de evaluación para medir y analizar el rendimiento del modelo sobre un conjunto de datos de prueba. La evaluación de los resultados permite entender cómo se comporta el modelo frente a datos no vistos y es esencial en el proceso de construcción y selección de modelos de aprendizaje automático.

4.5.1. Métricas generales

Antes de abordar las métricas específicas de los modelos construidos, es importante definir las métricas generales que se utilizan para evaluar cualquier problema de clasificación:

Matriz de confusión

Una de las herramientas más utilizadas para evaluar el rendimiento de modelos de clasificación es la matriz de confusión. Es una tabla cuadrada que muestra el número de instancias clasificadas correctamente e incorrectamente para cada clase. Las filas representan las clases reales y las columnas representan las clases predichas.

La estructura de la matriz de confusión depende del número de clases (N) en el problema de clasificación y se muestra de la siguiente manera:

	Clase Real 1	**Clase Real 2**	$\cdots$	**Clase Real N**
Clase Predicha 1	C_{11}	C_{12}	$\cdots$	C_{1N}
Clase Predicha 2	C_{21}	C_{22}	$\cdots$	C_{2N}
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
Clase Predicha N	C_{N1}	C_{N2}	$\cdots$	C_{NN}

Tabla 4.1: Matriz de confusión

Donde:

- C_{ii}: Número de instancias clasificadas correctamente en la clase i cuando su clase real es i.
- C_{ij} (donde $i \neq j$): Número de instancias clasificadas incorrectamente en la clase i cuando su clase real es j.

En la matriz de confusión, los elementos en la diagonal principal representan las muestras clasificadas correctamente y los elementos fuera de la diagonal principal; las clasificadas incorrectamente. A partir de la matriz de confusión, se pueden calcular varias métricas de evaluación, como ***precision***, ***recall***, ***f1-score*** y ***accuracy***.

Exactitud (*Accuracy*)

La exactitud es una métrica que proporciona una medida general del rendimiento del modelo. Es la proporción de instancias clasificadas correctamente en relación

con el total de instancias en el conjunto de datos de prueba.

$$\text{Precisión} = \frac{C_{11} + C_{22} + \cdots + C_{NN}}{C_{11} + \cdots + C_{1N} + C_{21} + \cdots + C_{2N} + \cdots + C_{N1} + \cdots + C_{NN}}$$

Un caso específico de los problemas multiclase es la clasificación binaria. En este escenario, la matriz de confusión adopta una estructura de 2x2, que se muestra en la siguiente tabla:

	Predicción Positiva	**Predicción Negativa**
Observación Positiva	Verdadero Positivo (VP)	Falso Negativo (FN)
Observación Negativa	Falso Positivo (FP)	Verdadero Negativo (VN)

Figura 4.2: Matriz de confusión binaria

Definidas para este caso [5], de esta matriz se extraen las siguientes métricas, que brindan información sobre el rendimiento del modelo en diferentes aspectos:

- **Precision** (Precisión): Es la proporción de instancias clasificadas correctamente como positivas en relación con el total de instancias clasificadas como positivas. Mide la capacidad del modelo para evitar falsos positivos.

$$\text{Precisión} = \frac{\text{VP}}{\text{VP} + \text{FP}}$$

- **Recall** (Exhaustividad): Es la proporción de instancias clasificadas correctamente como positivas en relación con el total de instancias que son realmente positivas. Mide la capacidad del modelo para encontrar todos los casos positivos.

$$\text{Recall} = \frac{\text{VP}}{\text{VP} + \text{FN}}$$

- **F1-Score** (Puntuación F1): Es una medida que combina la precisión y la exhaustividad en un solo valor. Es el promedio armónico de ambas métricas y proporciona un equilibrio entre ellas.

$$\text{F1 Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **Exactitud** (*Accuracy*): Definida anteriormente, se trata de una métrica que mide la proporción de instancias que el modelo clasificó correctamente en la clase positiva y la clase negativa.

$$\text{Exactitud} = \frac{\text{VP} + \text{VN}}{\text{VP} + \text{FP} + \text{VN} + \text{FN}}$$

4.5.2. Evaluación de los modelos construidos

En el caso de la segmentación, en el que se usa clasificación multiclase, se utiliza la exactitud del conjunto de pruebas para evaluar la capacidad del modelo para predecir correctamente, sumado con la matriz de confusión, de tamaño 3x3, para obtener una visión más detallada de cómo el modelo clasifica las tres diferentes clases.

En el caso de la detección, clasificación binaria, el objetivo principal es evitar choques de buques contra icebergs, por lo que se prioriza la detección de elementos en lugar de la no detección. Es decir, es preferible que el modelo detecte un iceberg incluso si no está presente, a que no lo detecte estando realmente presente. Para evaluar la capacidad de detección, se utiliza *recall* (sensibilidad). En este contexto, es importante minimizar los falsos negativos para asegurar una detección adecuada. Un alto *recall* indica que el modelo está capturando la mayoría de los elementos positivos y evitando la no detección de icebergs. Sin embargo, también es importante tener en cuenta otros aspectos, como el equilibrio entre el *recall* y la precisión (*f1 Score*).

4.6. Implementación

Una vez se han obtenido los modelos con los resultados óptimos considerados, se procede a la implementación de los mismos. El objetivo es detectar los bloques de hielo en una imagen completa utilizando un enfoque de ventana deslizante.

La implementación propuesta consiste en aplicar el modelo de detección de mar a imágenes completas. Se utiliza una ventana deslizante de tamaño 75x75, el mismo utilizado durante el entrenamiento de la red con parches. La ventana se mueve de un numero variable de 36 píxeles, la mitad del tamaño de la ventana; se ha decidido así para mejorar la eficiencia computacional.

En cada posición de la ventana, se utiliza el modelo de detección para determinar si esa región se clasifica como mar. El modelo proporciona tres posibilidades de clasificación: mar, tierra y banquisa. Si la probabilidad de que sea mar es la más

alta entre las tres posibilidades, se guarda la coordenada de la ventana en un vector.

Una vez obtenidas todas las coordenadas de las ventanas clasificadas como mar por el modelo, se obtiene un vector con la información necesaria para localizar las regiones marinas en la imagen completa. Estas coordenadas representan las ubicaciones donde se encontraron regiones marinas según las predicciones del modelo.

A continuación, se procede a aplicar el modelo de detección de hielo a las regiones marinas obtenidas utilizando el mismo enfoque de ventana deslizante. Se desliza una ventana de tamaño 75x75, moviéndose en cada iteración a lo largo de cada región marina.

En cada posición de la ventana, se aplica el modelo de detección de hielo para determinar si se detecta la presencia de hielo. Si la detección es positiva, se marca la región con una cuadrícula del mismo tamaño que la ventana.

Es importante tener en cuenta que este enfoque de ventana deslizante implica analizar cada subregión de forma independiente, lo que puede generar cierta superposición entre las ventanas adyacentes. Sin embargo, este problema será corregido y explicado en el siguiente apartado de postprocesamiento.

4.7. Etapa de postprocesamiento

Durante la fase de postprocesamiento, se lleva a cabo un análisis de superposición entre las cuadrículas generadas. En caso de que se detecte la superposición de dos o más cuadrículas, se procede a fusionarlas en una sola cuadrícula que abarca el área combinada.

La aplicación de esta técnica de fusión de cuadrículas ofrece ventajas notables, ya que contribuye a mejorar la calidad de la detección al evitar la duplicación de resultados. Además, proporciona una representación más precisa de la ubicación del hielo en las imágenes analizadas. Al reducir las detecciones redundantes y fusionar las áreas superpuestas, se obtiene una visión más coherente y precisa de la presencia de hielo en las distintas regiones de interés.

Después de completar el proceso de anotación y postprocesamiento, se obtendrán imágenes o representaciones visuales que mostrarán recuadros de diferentes tamaños en las áreas marinas donde se ha detectado la presencia de bloques de hielo. Estos re-

cuadros representan las regiones identificadas como hielo en las imágenes analizadas.

Como se menciona en la introducción 1, esta tecnología será incluida en una Sala de Control Virtual, por lo que estas detecciones serán enviadas a través de *Apache Kafka* [4], plataforma de *streaming* de eventos de alto rendimiento, utilizada para la transmisión de datos en tiempo real, para que los buques puedan contar con la información acerca de las posiciones de los bloques de hielo.

[4]https://kafka.apache.org/

Capítulo 5

Resultados y conclusiones

En esta sección se presentarán los hallazgos obtenidos en los procesos detallados en la sección 4. Se tratará de un apartado descriptivo, donde se plasmarán los resultados y, a través de un ejemplo, se mostrarán las fases del procedimiento. Todo ello anidado con la discusión pertinente acerca de los hallazgos encontrados.

5.1. Punto de partida

El objetivo final de este proceso es la detección de bloques de hielo en imágenes Sentinel-1, como se ha mencionado en las secciones anteriores. A continuación se presenta una figura de ejemplo que ilustra este tipo de imágenes:

Figura 5.1: Imagen sentinel-1

En esta figura, se puede observar que en la esquina superior derecha aparece un

segmento de tierra, mientras que en la esquina inferior izquierda se encuentra la banquisa, áreas donde no se desea localizar icebergs.

5.2. Rendimiento

En cuanto al rendimiento de los modelos de redes neuronales convolucionales, se han entrenado varios modelos con diferentes configuraciones de hiperparámetros. Los resultados obtenidos son los siguientes:

Segmentación del entorno marítimo

En la sección 4.4, se describe en detalle la estructura del modelo de segmentación. Por lo tanto, se presenta un resumen de dicho modelo:

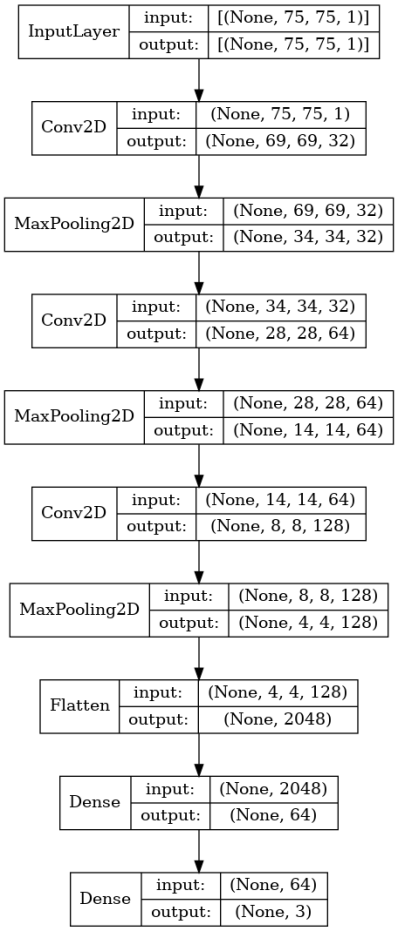

Figura 5.2: Resultados modelo CNN segmentación marina

Los resultados obtenidos para el modelo son los siguientes: en el conjunto de entrenamiento, se obtuvo una exactitud de 0.79 y una función de pérdida de 0.6; en el de validación, se logró un accuracy de 0.53 y una función de pérdida de 1.03; y en el conjunto de prueba, se alcanzó un accuracy de 0.58 y una función de pérdida de 0.99.

Además, se generaron matrices de confusión para cada conjunto, las cuales proporcionan información adicional sobre el rendimiento del modelo, pueden visualizarse en la figura 5.4.

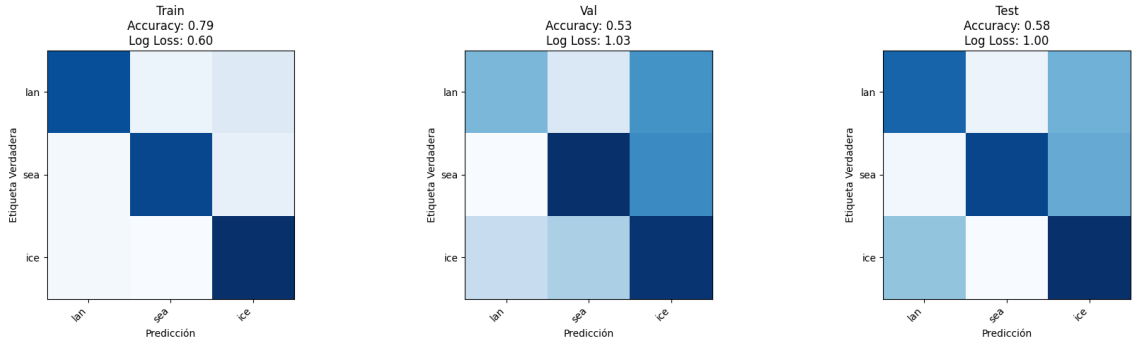

Figura 5.3: Matrices de confusión sobre el rendimiento del modelo de segmentación marina

A la vista de los resultados, se pueden extraer dos conclusiones significativas. En primer lugar, es posible que el modelo esté ligeramente sobreajustado a los datos de entrenamiento, ya que muestra una mejora en la predicción en comparación con el conjunto de test. Esto plantea la preocupación de que el modelo pueda tener dificultades para generalizar adecuadamente a nuevos datos.

En segundo lugar, es crucial tener en cuenta que los resultados del conjunto de test, que son los más relevantes para evaluar el rendimiento del modelo en datos desconocidos, y estos muestran un nivel de exactitud bastante modesto. Esto sugiere que la segmentación generada por el modelo no es lo suficientemente sólida como se esperaba.

Teniendo en cuenta estas observaciones, es esencial explorar posibles alternativas y mejoras en fases posteriores. En el apartado correspondiente a futuras propuestas 6, se proponen enfoques adicionales que podrían ayudar a abordar las limitaciones actuales. Dado que la detección del mar desempeña un papel crucial en el proceso, es fundamental considerar opciones adicionales para fortalecer la robustez y precisión

de la segmentación.

Detección de los bloques de hielo

Al igual que en el modelo anterior, el resumen del modelo aparece a continuación.

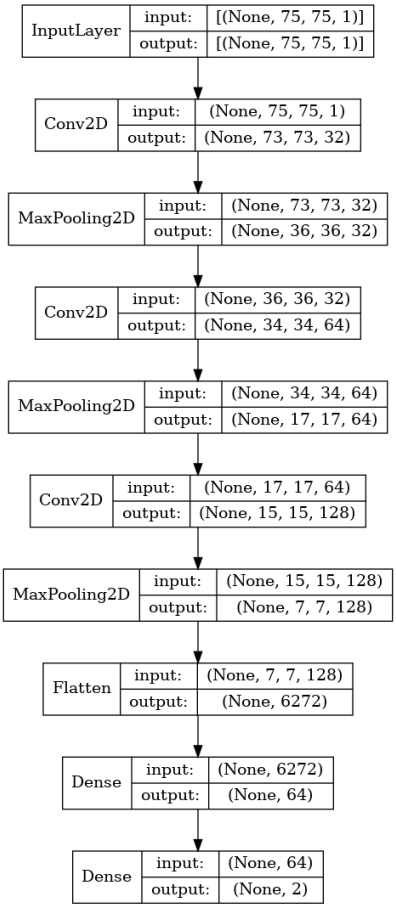

Figura 5.4: Red neuronal para la detección de hielo

A continuación, se presentan los resultados del modelo de detección de bloques de hielo. Es importante destacar que, en este contexto, lo que se busca es la detección precisa de icebergs y minimizar los casos de no detección. Por lo tanto, las métricas más relevantes son el *recall* y el *f1-score*.

	Precision	Recall	f1-score	support
Train	0.94	0.95	0.95	551
Val	0.82	0.84	0.83	136
Test	0.92	0.82	0.86	66

Tabla 5.1: Resultados modelo CNN detección de hielo

En este caso, los resultados del modelo se consideran aceptables, ya que en ningún caso descienden por debajo de 0.8. Siendo centrado el análisis en la columna de *recall*, se puede observar que en el conjunto de entrenamiento es ligeramente superior a la precisión, mientras que en el conjunto de prueba, el *recall* disminuye considerablemente de 0.95 a 0.82. A pesar de esta disminución, se considera un resultado satisfactorio.

Estos resultados indican que el modelo es capaz de detectar correctamente la mayoría de los bloques de hielo presentes en las imágenes, aunque existe un margen de mejora en el conjunto de prueba.

5.3. Implementación de la solución propuesta

Una vez que los modelos elegidos han sido entrenados, se puede proceder a la implementación y marcación de las imágenes. En primer lugar, se realiza una prueba utilizando únicamente el modelo de detección de bloques de hielo, sin aplicar el modelo de segmentación previo. Esta prueba se muestra en la Figura 5.5.

Esta prueba se realiza considerando los resultados obtenidos por el modelo de segmentación, que no han sido satisfactorios. En la imagen, se puede observar la presencia de ciertos bloques de hielo en el mar, mientras que la mayoría de las detecciones se encuentran sobre la tierra, lo cual es lógico. También se pueden ver algunas detecciones en la banquisa, lo cual puede ocurrir debido a su similitud con los bloques de hielo independientes.

Tras observar cómo se ve la detección de bloques de hielo en toda la imagen, se puede suponer cómo debería ser la detección únicamente en las zonas marítimas. Esto se muestra en la Figura 5.6, donde se evidencia cómo el modelo de segmentación realmente realiza la segmentación del entorno marítimo, siguiendo las conclusiones extraídas de los resultados de ajuste y la matriz de confusión.

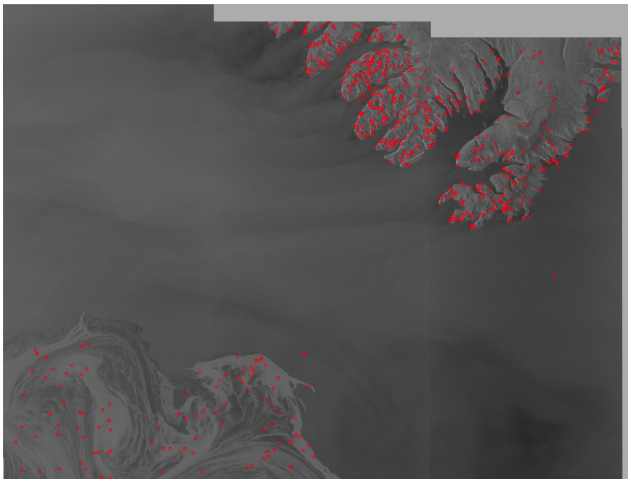

Figura 5.5: Implementación única del modelo de detección de hielo

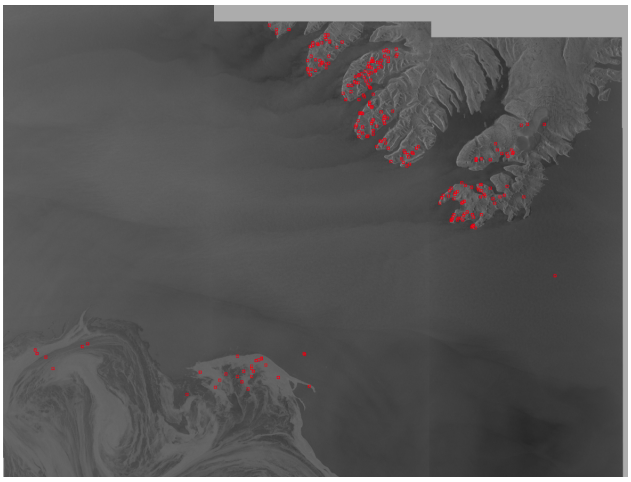

Figura 5.6: Implementación de los dos modelos

En las dos imágenes anteriores se puede apreciar cierto solapamiento entre algunos de los rectángulos de detección. Este solapamiento se resuelve en una fase de refinamiento conocida como postprocesamiento, que garantiza que los bloques de hielo sean marcados sin que haya intersecciones entre las detecciones. El resultado final de la imagen de ejemplo se muestra a continuación.

En la imagen se observan los rectángulos de diferentes tamaños que ayudan a mejorar la visualización de la detección de los bloques de hielo.

A través del ejemplo se han ido mostrando cómo son los diferentes resultados de

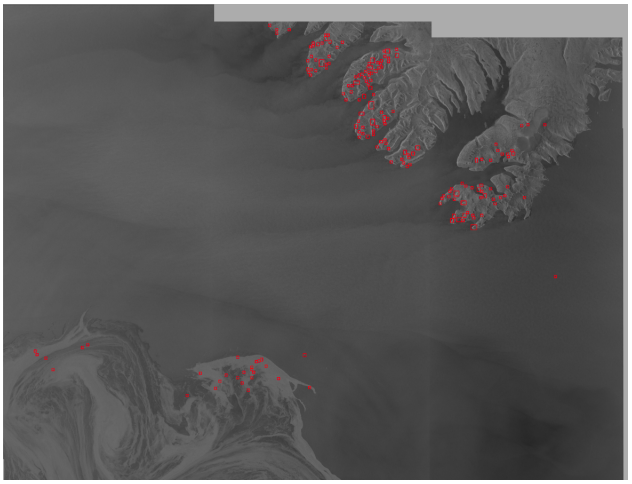

Figura 5.7: Implementación completa sobre la imagen ejemplo

las distintas fases al ser implementadas sobre la imagen propuesta. Se ha de tener en cuenta que el programa está diseñado para que a través de un input, imagen Sentinel-1 (Figura 5.1), devuelva el output de la imagen final anterior (Figura 5.7).

Se observa que en ciertas zonas de la tierra y de la banquisa tiene problemas para detectar que no se trata de área correspondiente al mar. En la siguiente sección se verá como de precisa es la mejora propuesta para líneas futuras.

Capítulo 6

Conclusiones y líneas futuras

Esta sección presenta una síntesis de los hallazgos y su relevancia en el contexto del estudio, así como posibles aplicaciones y beneficios de los resultados obtenidos para la continuación futura de la investigación.

En este proyecto, se ha logrado desarrollar un algoritmo para la detección de hielo marino en imágenes satelitales del Ártico. A través de la clasificación de zonas navegables y no navegables, así como la detección de bloques de hielo, se ha buscado mejorar la toma de decisiones y la seguridad para los actores marítimos en estas regiones. Esta tecnología, que será incluida en la sala de control virtual de AI-ARC es capaz de enviar las posiciones de los icebergs detectados en áreas árticas. En términos generales, los resultados obtenidos son aceptables y se ha cumplido con el objetivo principal. Sin embargo, siempre hay margen de mejora y se identifican posibles líneas futuras para continuar perfeccionando el proceso.

Líneas futuras

Una posible línea futura para mejorar el modelo sería optimizar el proceso de segmentación aplicando técnicas de aumento de datos (*data augmentation*), descrito en la subseccion 3.2.2 al modelo de CNN. Al aumentar la cantidad de datos de entrenamiento, es posible mejorar la precisión en el conjunto de prueba sin incurrir en un tiempo excesivo de procesamiento.

Además, se ha explorado el uso del modelo SAM [1] (Segment Anything Model). SAM es un enfoque de segmentación desarrollado por Meta que tiene la capacidad de segmentar cualquier tipo de imagen en diferentes regiones. La segmentación de

[1] https://github.com/facebookresearch/segment-anything

imágenes es el proceso de dividir una imagen en segmentos significativos con el objetivo de identificar y aislar objetos o áreas de interés en la imagen, se basa en la detección de límites y características distintivas en la imagen, con el fin de agrupar píxeles o regiones similares. Este modelo utiliza técnicas avanzadas de aprendizaje automático y procesamiento de imágenes para generar máscaras que delimitan las diferentes partes de la imagen. El código fuente y los archivos relacionados están disponibles para su consulta y, por lo tanto, para su posible adaptación para abordar el problema específico que concierne al proyecto.

Al realizar pruebas con este modelo, se obtuvieron resultados precisos y más exactos que con el enfoque manual por parches. Este modelo genera una máscara que podría utilizarse para aplicar el segundo modelo de CNN, lo cual podría mejorar significativamente la detección de icebergs.

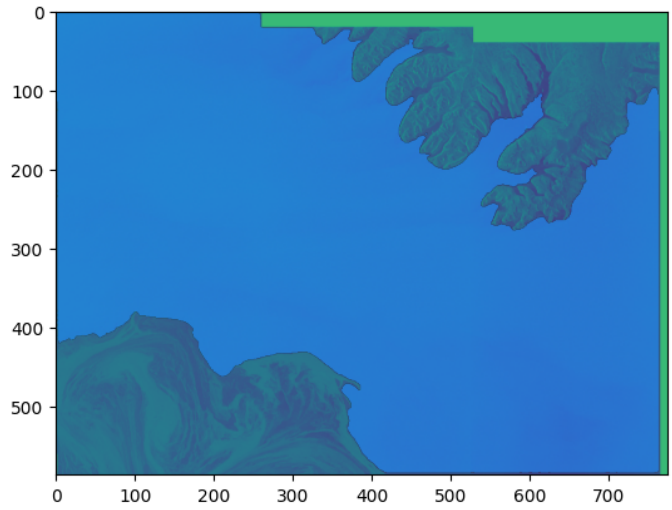

Figura 6.1: Imagen tras aplicar el modelo SAM, donde aparece dibujada la mascara de la detección de mar en color azul

En conclusión, y puesto a que la búsqueda de conocimiento y la innovación continuada son cruciales para preservar nuestros ecosistemas polares y garantizar un futuro sostenible para las generaciones venideras, este proyecto de detección de hielo marino en imágenes satelitales del Ártico, representa un paso hacia la mejora de la seguridad y la toma de decisiones en las regiones polares.

A través de la combinación de modelos de segmentación y detección de bloques de hielo, se ha logrado obtener resultados que pueden llegar a ser prometedores. Sin

embargo, este proyecto no marca el final de la investigación, sino que, sumado a las tecnologías implementadas por el resto de los participantes del programa AI-ARC, seguirá en busca de soluciones cada vez más precisas y eficientes.

Bibliografía

[1] Bottou, L., Curtis, F. E., Nocedal, J. (2018). Optimization Methods for Large-Scale Machine Learning. *SIAM Review*, 60(2), 223-311.

[2] Duchi, J., Hazan, E., Singer, Y. (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research*, 12, 2121-2159.

[3] Goodfellow, I., Bengio, Y., Courville, A. (2016). *Deep Learning*. The MIT Press.

[4] Hastie, T., Tibshirani, R., Friedman, J. (2001). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer.

[5] Hand, D. J., Till, R. J., Duda, R. O. (2001). A note on the ROC analysis of probabilistic classification algorithms. *Pattern Recognition Letters*, 22(12), 1289-1294.

[6] Han, J., Kamber, M., Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann.

[7] Krizhevsky, A., Sutskever, I., Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems* (pp. 1097-1105).

[8] LeCun, Y., Bengio, Y., Hinton, G. (2015). Deep learning. *Nature*, 521(7553), 436-444.

[9] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation Applied to Handwritten Zip Code Recognition, Neural Computation, 1(4) (1989) pp.541-551

[10] Li, Y., Wu, J., Li, C., Lu, W., Wu, Y. N., Lu, C. (2021). Stable Diffusion: Learning Stable Latent Dynamics for Motion Forecasting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 4480-4489).

[11] Maas, A. L., Hannun, A. Y., Ng, A. Y. (2013). Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)* (pp. 3-11).

[12] McCorduck, P. (2004). *Machines Who Think: A Personal Inquiry into the History and Prospects of Artificial Intelligence* (2nd ed.). Natick, MA: A.K. Peters Ltd.

[13] McCarthy, J., Minsky, M. L., Rochester, N., Shannon, C. E. (1956). A proposal for the Dartmouth Summer Research Project on Artificial Intelligence. *AI Magazine*, 27(4), 12-14.

[14] Peng, B., Li, J., Cheng, C., Smith, J. R., Luo, J. (2021). GroundingDino: Semantic Image Synthesis from Text with Discriminators. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 11063-11073).

[15] Redmon, J., Divvala, S., Girshick, R., Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 779-788).

[16] Ramesh, A., Radford, A., Sutskever, I., Kolesnikov, A. (2021). Scaling Laws for Neural Language Models. In *Proceedings of the International Conference on Learning Representations (ICLR).*

[17] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A. C., Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3), 211-252.

[18] Rumelhart, D. E., Hinton, G. E., Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323, 533-536.

[19] Sutton, R. S., Barto, A. G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.

[20] Wang, M., Overland, J. E., Stroeve, J. C. (2018). Future climate change and Arctic sea ice decline. *Nature*, 520(7549), 495-504.

[21] Zhang, H., Lu, Y., Ban, X., Zhu, X. (2021). SAM: Segment Anything You Want. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 8969-8978).

De acuerdo con lo expresado en el *artículo 8.3 del Reglamento para la elaboración y defensa del Trabajo Fin de Máster de la Universidad de Oviedo*, aprobado por su Consejo de Gobierno el 17 de julio de 2020 (BOPA de 7 de agosto de 2020), quiero expresar lo siguiente:

Yo, ALICIA LOJO IGLESIAS, con DNI , en relación a la memoria que presento ante el Tribunal, para su valoración como ***Trabajo Final en el Máster Universitario en Análisis de Datos para la Inteligencia de Negocios (MANADINE),*** quiero ***DECLARAR*** que ***s***oy el autor de la misma, habiendo citado debidamente las fuentes utilizadas en su desarrollo.

Para que conste, firmo el presente documento.

Oviedo, 16 de julio de 2023

Fdo.- Alicia Lojo Iglesias