ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA
INFORMÁTICA

Grado en Ciencia e Ingeniería de Datos

Trabajo Fin de Grado

Predicción de Datos Oceanográficos con Redes Neuronales de Grafos y Aprendizaje por Conjuntos

Alejandro Jesús González Santana

Tutores

Javier Sánchez Pérez

Giovanny Alejandro Cuervo Londoño

Las Palmas de Gran Canaria
25 de julio de 2025

ULPGC Universidad de Las Palmas de Gran Canaria | EII ESCUELA DE INGENIERÍA INFORMÁTICA

TFT04

SOLICITUD DE DEFENSA DE TRABAJO DE FIN DE TÍTULO

D./Dª **Alejandro Jesús González Santana**, autor del trabajo **Predicción de Datos Oceanográficos con Redes Neuronales de Grafos y Aprendizaje por Conjuntos**, correspondiente a la titulación **Grado en Ciencia e Ingeniería de Datos**.

SOLICITA

que se inicie el procedimiento de defensa del mismo, para lo que se adjunta la documentación requerida, haciendo constar que

[X] se autoriza / [ ] no se autoriza la grabación en audio de la exposición y turno de preguntas.

Asimismo, con respecto al registro de la propiedad intelectual/industrial del TFT, declara que:
[ ] Se ha iniciado o hay intención de iniciarlo (defensa no pública).
[X] No está previsto.

Y para que así conste firma la presente. (fecha en firma electrónica)

El/La estudiante

Fdo. Alejandro Jesús González Santana

A rellenar y firmar **obligatoriamente** por el/la/los/las tutores

En relación con la presente solicitud, se informa:(firmar donde corresponda)

Positivamente (en caso de detección de copia, esta firma quedará invalidada)	Negativamente (justificación en TFT05)
Fdo. Javier Sánchez Pérez, Giovanny Alejandro Cuervo Londoño	Fdo. Javier Sánchez Pérez, Giovanny Alejandro Cuervo Londoño

DIRECTORA DE LA ESCUELA DE INGENIERÍA INFORMÁTICA

Predicción de Datos Oceanográficos con Redes Neuronales de Grafos y Aprendizaje por Conjuntos

Trabajo Fin de Grado

Alejandro Jesús González Santana

Tutores

Javier Sánchez Pérez

Giovanny Alejandro Cuervo Londoño

Grado en Ciencia e Ingeniería de Datos

Escuela de Ingeniería Informática

Universidad de Las Palmas de Gran Canaria

Las Palmas de Gran Canaria

25 de julio de 2025

Índice general

Índice de figuras

Índice de tablas

Agradecimientos

Quiero comenzar expresando mi agradecimiento a los tutores de este trabajo, Javier y Giovanny, quienes me han guiado durante todo este proyecto, ayudándome con paciencia en cada duda que surgía y brindándome su constante disponibilidad y apoyo.

Extiendo también mi agradecimiento a todos los profesores que me han acompañado a lo largo de este proceso de aprendizaje durante la carrera. Gracias a ellos, hoy me llevo grandes recuerdos, conocimientos y habilidades.

A mi familia y amigos, gracias por estar siempre ahí. Su apoyo incondicional, su confianza en mí y su compañía constante han sido fundamentales para seguir adelante, especialmente en los momentos más exigentes del camino.

Finalmente, quiero dedicar un agradecimiento especial a quienes han hecho posible que haya llegado hasta aquí: mis padres, Victoria y Jesús. Gracias por su amor, por todo lo que han hecho por mí, y por darme siempre las oportunidades para crecer como persona y alcanzar esta meta.

Resumen

El objetivo principal del trabajo es probar la predicción de variables oceanográficas combinando la salida de varios modelos (aprendizaje por conjuntos) basados en redes neuronales de grafos. Se parte de un modelo existente (SeaCast), que ha sido adaptado a la zona de Canarias en el océano Atlántico. La variante de aprendizaje por conjuntos empleada es Bagging, donde la diversidad en los modelos se obtiene introduciendo ruido a la entrada. La combinación de modelos suele ser especialmente útil a largo plazo, donde debería mejorar la predicción de un único modelo. Los datos que alimentan al modelo no se limitan solo a variables oceanográficas, sino que también se emplean condiciones externas atmosféricas y temporales.

Abstract

The main purpose of the project is to predict oceanographic variables by merging the outputs of several models (ensemble learning) derived from a graph neural network. The base model (Seacast) has been adjusted to the Canary Islands area in the Atlantic Ocean. The ensemble learning variant used is Bagging, where the model diversity is achieved by adding noise to the input. Benefits of model combination are usually noticeable in long-range predictions, where one model predictions should be less accurate. The data used to feed the model not only include oceanographic variables, but also external forcing factors such as atmospheric conditions or timestamps.

Capítulo 1

Introducción

La monitorización del océano ha adquirido una gran importancia en las últimas décadas. El auge de la economía azul [Novellino et al., 2024], el cambio climático y la definición de los Objetivos de Desarrollo Sostenible (ODS) [Veitch et al., 2025] han aumentado la necesidad de modelos y soluciones climáticas que permitan dar respuesta a retos futuros. Típicamente los modelos de predicción han sido numéricos, empleando leyes físicas que resuelven ecuaciones complejas. Este paradigma de predicción, pese a ser robusto, es solo utilizado por las grandes agencias y organizaciones climáticas debido a sus altos costes computacionales.

En los últimos años, el aprendizaje automático ha impulsado una nueva generación de modelos más eficientes y accesibles. Estos modelos, al no depender de simulaciones físicas, permiten realizar inferencias rápidas en equipos convencionales, facilitando la democratización de la investigación en oceanografía.

No obstante, muchos desarrollos están liderados por grandes compañías o grupos de investigación que elaboran modelos a una escala global. Estos modelos se caracterizan por tener una menor resolución espacial, ya que están diseñados para abarcar regiones más extensas. Como consecuencia, se pierde detalle en los complejos procesos a pequeña escala. Una estrategia comúnmente usada para compensar los errores es el uso de múltiples modelos para realizar predicciones, cuyos resultados se combinan posteriormente. Esta metodología, conocida como aprendizaje por conjuntos (*ensemble*), se caracteriza por la diversidad en el conjunto de modelos, ya sea por su arquitectura, parámetros o datos de entrada. En este proyecto se adapta SeaCast [Holmberg et al., 2024], un modelo basado en la arquitectura de redes neuronales de grafos (GNN), a la zona del Atlántico Norte. Este tipo de redes son capaces de adaptarse a la geometría irregular del océano, que suele estar influenciada por la presencia de islas y regiones de tierra. Sobre esta base, se construyen conjuntos de predicciones para explorar las ventajas de esta metodología en la predicción de variables oceanográficas, así como para comparar el desempeño entre distintas configuraciones. La generación de estos conjuntos se realiza mediante la aplicación de perturbaciones iniciales a los datos de entrada de un mismo modelo en fase de inferencia.

A continuación, se exponen las motivaciones que dieron origen a este proyecto, los objetivos planteados durante su planificación, así como su alcance y limitaciones, y finalmente, la estructura del documento.

1.1. Motivaciones

Actualmente existen una gran cantidad de repositorios que ofrecen productos de variables oceanográficas con alta resolución. Aunque los modelos más potentes suelen operar a escala global, el creciente aumento del desarrollo de modelos de aprendizaje automático ha facilitado el desarrollo de modelos regionales. Estos modelos permiten trabajar con datos de mayor resolución y precisión, abriendo nuevas posibilidades en el estudio de fenómenos locales.

La zona de estudio en este trabajo corresponde a la costa atlántica que rodea a las Islas Canarias y al noroeste de África. Esta región se distingue por un fenómeno denominado afloramiento costero (*upwelling*), que transporta nutrientes desde las profundidades del océano hasta la superficie. El afloramiento costero es el resultado de procesos complejos de flujo oceánico y la acción de los vientos Alisios que desplazan la superficie oceánica [NOAA, 2025b]. Debido a su complejidad, la identificación de este proceso requiere del uso de datos de alta resolución y técnicas capaces de capturar los complejos patrones espaciales.

El uso de GNNs aparece como una solución prometedora, ya que se alejan del paradigma tradicional de redes neuronales basada en estructuras regulares (como las convolucionales o recurrentes). Las GNNs trabajan con una estructura de grafos, donde los nodos representan puntos de datos y las conexiones representan relaciones espaciales entre ellos. Esta estructura adaptable permite modelar datos de dominios irregulares como los océanos, que están rodeados de islas y continentes, facilitando el aprendizaje de patrones locales y espaciales.

Además, el uso de conjuntos de predicciones permite representar la incertidumbre de los procesos oceánicos. En zonas donde los fenómenos a pequeña escala son importantes, como en la región de estudio, el aprendizaje por conjuntos ayuda a realizar predicciones más robustas mediante la combinación de múltiples modelos o la perturbación de los datos de entrada. Asimismo, en el futuro, los conjuntos de modelos serán el estándar operacional para las predicciones oceanográficas [Hoteit et al., 2025].

El uso de conjuntos también está relacionado con una de las críticas a los modelos de aprendizaje automático en predicción climática. Estos se caracterizan por producir una única predicción para un estado, por lo que tienen una capacidad limitada para representar la incertidumbre propia de los procesos físicos. En la predicción numérica, es muy frecuente utilizar distintos datos de entrada o parámetros para realizar simulaciones más precisas. Por lo tanto, la aplicación de técnicas de aprendizaje por conjuntos puede mejorar la capacidad de captar la incertidumbre de estos modelos, así como generar conjuntos de predicciones más robustas.

Por otra parte, la comparación de técnicas de aprendizaje por conjuntos es clave para identificar cuáles aportan una mejor representación de la incertidumbre en procesos oceánicos. Esto es especialmente relevante en contextos con recursos computacionales limitados, como en este proyecto donde se trabaja con conjuntos en fase de inferencia. Una de las características de esta metodología es la gran variedad de posibilidades para generar conjuntos de modelos o predicciones. Por lo tanto, la evaluación de estos parámetros específicos y de distintas configuraciones de ruido resulta fundamental para optimizar el desempeño y eficiencia de estas técnicas.

1.2. Objetivos

El objetivo principal de este proyecto es evaluar la predicción de variables oceanográficas mediante la combinación de las salidas de varios modelos basados en redes neuronales de grafos.

El primer objetivo consiste en adaptar el modelo de SeaCast, desarrollado para la región del mar Mediterráneo, a la zona del Atlántico Norte. Este proceso implica comprender y trabajar con datos oceanográficos multidimensionales, así como recopilar datos de mayor resolución para la zona de estudio.

Otro objetivo es comprender la red neuronal de grafos propuesta por SeaCast, la cual se basa en otros trabajos como Neural-LAM [Oskarsson et al., 2023] o GraphCast [Lam et al., 2023]. Se estudia su proceso de entrenamiento y su capacidad para mejorar la predicción de variables oceanográficas.

La exploración de los métodos de aprendizaje por conjuntos disponibles es otro aspecto fundamental de este proyecto. Se investigan diferentes técnicas, tanto generales como específicas para la predicción climática, destacando especialmente la alteración de datos de entrada mediante el uso de ruido. El objetivo de aplicar conjuntos de predicciones es desarrollar predicciones más robustas a medio y largo plazo, al menos alcanzando el rendimiento de un único modelo. Debido a la particularidad de este enfoque, también se busca comprender y aplicar métricas de evaluación especializadas, conocidas como métricas probabilísticas.

Por último, se pretende realizar un análisis de los resultados obtenidos y de los puntos débiles identificados, con el fin de establecer futuras líneas de investigación que mejoren este enfoque. Estos resultados se centran en la identificación de configuraciones de ruido con mejor rendimiento en conjuntos. Además, se busca optimizar la eficiencia computacional del proceso, introduciendo las perturbaciones durante la fase de inferencia, lo que permite la ejecución de entrenamientos durante el desarrollo de pruebas.

Desde el punto de vista académico, se busca afianzar conceptos abordados a lo largo de la carrera. Entre ellos, se encuentran el análisis y procesamiento de datos, el uso de modelos de aprendizaje profundo y el diseño de pruebas, especialmente en entornos que manejan grandes volúmenes de datos similares a los del mundo real. El enfoque también tiene como objetivo ampliar los conocimientos sobre redes neuronales de grafos, una temática que no se ha tratado de forma detallada durante la carrera.

1.3. Alcance y limitaciones

Este proyecto presenta una serie de limitaciones que deben tenerse en cuenta a la hora de interpretar los resultados. En primer lugar, el aprendizaje por conjuntos se aplica únicamente en la fase de inferencia. Esto implica que el modelo base fue entrenado con datos no perturbados, lo que podría afectar negativamente al rendimiento del conjunto en comparación con enfoques donde la diversidad se introduce en la fase de entrenamiento.

Asimismo, el entrenamiento del modelo base debió ajustarse para adaptarse a las restricciones computacionales disponibles, lo que puede haber limitado su capacidad de generalización. Aunque los resultados obtenidos pueden servir como referencia para estudios futuros, se debe tener en cuenta el contexto específico en el que han sido generados, especialmente en cuanto a las variables empleadas. Diferentes configuraciones de entrada podrían producir resultados distintos, por lo que los hallazgos deben interpretarse con cierta cautela.

Por otro lado, debido a la gran cantidad de posibles configuraciones en el aprendizaje por conjuntos, este proyecto se ha delimitado en el análisis de dos tipos de ruido (ruido Gaussiano y ruido de Perlin), seleccionados por su capacidad de ser comparables entre sí. Esta elección también permite evaluar el efecto de parámetros específicos sobre la generación de ruido y su impacto en las predicciones.

Pese a estas limitaciones, este proyecto es un primer paso en la evaluación de técnicas de perturbación para la creación de conjuntos de predicciones, y puede servir como referencia para futuras investigaciones que busquen mejorar la representación de incertidumbre en entornos oceánicos.

1.4. Organización del documento

Este documento se compone de un total de ocho capítulos, siendo el primero de ellos esta Introducción, donde se presenta el contexto necesario para explicar las motivaciones y objetivos del trabajo.

El capítulo 2 corresponde al estado del arte. En él se recorren los diferentes componentes del modelo de predicción oceanográfica aplicado, comenzando por una explicación de los modelos numéricos climáticos tradicionales, y su evolución hacia los modelos de aprendizaje automático, que se ilustra con ejemplos de modelos y tendencias actuales. Tras ello, se presentan los conceptos clave acerca de redes neuronales de grafos, con el objetivo de introducir su funcionamiento, así como su aplicación en modelos oceanográficos para la predicción. En la siguiente sección, se exploran las técnicas de aprendizaje por conjuntos desde una perspectiva general, repasando las estrategias principales y casos de uso. Como siguiente paso, se combinan los dos temas anteriores para revisar el uso del aprendizaje por conjuntos aplicado a GNN. Por último, se realiza una primera mención a SeaCast de forma introductoria y se consideran las limitaciones actuales de las GNNs y de las técnicas de aprendizaje por conjuntos.

En el capítulo 3 se detalla la planificación del proyecto, donde se explica la metodología de ciencia de datos empleada para su estructuración en fases. También se mencionan los cambios realizados en las tareas respecto a lo planificado en documentos anteriores. Por último, se efectúa una recopilación de los recursos empleados para el desarrollo del proyecto.

En el capítulo 4 se describen en detalle las fuentes de datos empleadas, junto con las decisiones de diseño tomadas durante la adaptación del proyecto original. De la misma forma, se detallan las características de los distintos datos utilizados y cómo se realizó la división de datos para el entrenamiento.

En el capítulo 5 se realiza un recorrido por el funcionamiento del modelo base SeaCast y de las redes neuronales de grafo. Cada subsección de este capítulo trata un componente específico del modelo, así como también se estudian algunos de los modelos previos sobre los que se construyó SeaCast. Finalmente, se hace referencia a todos los aspectos relacionados con el flujo de trabajo operativo del modelo. Se realiza una explicación general del funcionamiento de SeaCast hasta la generación de conjuntos de predicciones, junto los cambios realizados en los distintos módulos, tanto para su adaptación como para la aplicación del aprendizaje por conjuntos.

En el capítulo 6 se aborda la configuración específica de las pruebas y entrenamientos. En primer lugar, se presentan las estrategias de aprendizaje por conjuntos consideradas a lo largo del proyecto, prestando especial atención a diversos tipos de ruido. También se presenta el mecanismo de unificación de conjuntos implementado y se definen las métricas

de evaluación utilizadas. Finalmente, se revisan las estrategias de entrenamiento y el diseño de pruebas para comparar las técnicas de perturbación inicial empleadas.

En el capítulo 7 se analizan en detalle los resultados de las pruebas descritas en el apartado de configuración de experimentos. En primer lugar, se comparan los rendimientos de los métodos de conjuntos frente a las predicciones deterministas. Después, se analizan las diferencias entre los distintos tipos de ruido aplicados en las perturbaciones iniciales, tanto a nivel individual como global.

Por último, el capítulo 8 presenta una revisión general del desarrollo del proyecto. Se realiza un análisis global de los resultados obtenidos, tanto de la adaptación de SeaCast como de su aplicación mediante conjuntos. A continuación, se valora la consecución de los objetivos académicos y propios del proyecto, señalando también aquellos que no fueron alcanzados. Por último, se consideran las limitaciones y retos enfrentados durante el desarrollo del trabajo y se plantean posibles líneas futuras de investigación derivadas desde este.

Para finalizar el documento, se incorporan varios apéndices que aclaran ciertos aspectos relevantes. En el apéndice A se presentan resultados adicionales sobre los distintos tipos de ruido empleados para conformar la diversidad del conjunto de predicciones, relacionados con la intensidad y distribución del ruido añadido. En el apéndice B se abordan las competencias específicas del grado en Ciencia e Ingeniería de Datos que este proyecto cubre. El apéndice C presenta la relación del proyecto con los Objetivos de Desarrollo Sostenible, determinando su posible impacto medioambiental y en la sociedad. Por último, se incluye una lista de acrónimos y un glosario, así como las fuentes consultadas para el desarrollo del proyecto en la bibliografía.

Capítulo 2

Estado del arte

El crecimiento emergente de la economía azul ha centrado la atención en el análisis de datos oceánicos para la toma de decisiones en tiempo real. Los modelos de predicción se emplean en actividades como la protección del ecosistema marino, el transporte y comercio marítimo, la producción alimentaria, las energías renovables y el desarrollo de actividades turísticas y portuarias, entre otras. El potencial financiero, laboral y sostenible de estos sectores demanda el desarrollo de herramientas cada vez más precisas que faciliten soluciones confiables, innovadoras y rápidas [Novellino et al., 2024].

La eficaz predicción de las variables oceánicas es considerada también una herramienta cuya explotación podría ser beneficiosa para la consecución de los ODS propuestos por la Organización de las Naciones Unidas (ONU). El progreso requiere que la comunidad investigadora sea capaz de dar soporte al desarrollo de aplicaciones dirigidas a usuarios finales, de manera que la información extraída de las salidas de los modelos sea asimilada adecuadamente. De esta forma, la sociedad contará con medios accesibles que fomentarán una mayor concienciación e implicación [Veitch et al., 2025].

El propósito de este capítulo es presentar la situación actual de las técnicas y procedimientos empleados en este trabajo con el fin de desarrollar un modelo basado en aprendizaje automático aplicado a la predicción oceanográfica. En primer lugar, se examinan las técnicas de predicción climáticas tradicionales y su evolución a lo largo del tiempo, así como los principales proyectos e instituciones que han impulsado este cambio. Posteriormente, se exploran los temas centrales del estudio con el propósito de ofrecer una primera aproximación a los conceptos y analizar su uso en el panorama actual de la investigación. Estos ejes fundamentales son tanto las GNNs como los métodos de aprendizaje por conjuntos. Por último, se realiza una breve introducción al modelo base de este proyecto, SeaCast, y se explican algunas de las limitaciones a las que se enfrentan las GNNs.

2.1. Modelos climáticos numéricos

Los modelos climáticos globales son fundamentales para la predicción del clima, la emisión de avisos por fenómenos adversos y la monitorización de los efectos del calentamiento global. Estos modelos se basan en ecuaciones diferenciales complejas que se resuelven sobre una cuadrícula global, proporcionando como resultado una representación de las dinámicas climáticas. El globo terráqueo se divide en celdas definidas por latitud, longitud y elevación, sobre las cuales se aplican ecuaciones específicas para los componentes atmosféricos, terrestres, oceánicos y de hielo. Este enfoque es empleado por los

modelos de predicción numérica del tiempo (NWP - *Numerical Weather Prediction*).

El tamaño de la cuadrícula desempeña un papel fundamental en la salida de los modelos. Utilizar un mayor número de celdas implica una mejor resolución, pero también un mayor coste computacional. Sin embargo, al sacrificar resolución a favor del rendimiento, se obtienen resultados menos detallados. Los modelos se ejecutan en distintos intervalos temporales determinados, actualizando las tendencias obtenidas con la aparición de nuevos datos y el refinamiento de modelos.

De este modo, los modelos numéricos tradicionales se apoyan en fundamentos físicos y son verificados por una gran comunidad de instituciones e investigadores. Además, la precisión de las predicciones se ha contrastado con observaciones reales, especialmente a gran escala, lo que ha sido de gran ayuda para reducir la incertidumbre sobre las variaciones climáticas y diseñar planes de contención.

No obstante, cuentan con desventajas principalmente relacionadas al cómputo necesario para la ejecución, así como a la inclusión de nuevas fórmulas o componentes para mejorar la precisión. La evolución de estos modelos numéricos depende del poder computacional disponible y de la resolución de los datos [NOAA, 2025a].

2.2. Evolución hacia el aprendizaje automático

La irrupción del aprendizaje automático ha sido acogida por agencias e investigadores debido a su gran potencial para mejorar las predicciones. Este concepto fue definido por primera vez por Arthur Samuel como:

> *El área de estudio que da la habilidad a las computadoras de aprender sin necesidad de ser programadas explícitamente.* – Fuente: [Samuel, 1959]

Desde entonces, el aprendizaje automático ha evolucionado y se ha consolidado como una herramienta estándar para abordar múltiples problemas. Su principal contribución al campo de la predicción climática es que no se basa en conocimiento especializado, a diferencia de los modelos numéricos. Es decir, estos modelos pueden identificar patrones y relaciones en los datos sin la necesidad de definir fórmulas complejas desarrolladas por expertos, lo cual facilita que se puedan desarrollar más aplicaciones por perfiles profesionales no vinculados a la climatología. Otra ventaja importante es la capacidad de aprovechar la potencia computacional de las tarjetas gráficas actuales (GPU - *Graphics Processing Unit*). Los entrenamientos de modelos pueden ejecutarse en paralelo en varias máquinas para reducir los tiempos que suelen ser prolongados.

Una vez entrenados, los modelos de aprendizaje automático se caracterizan por su rapidez en la fase de inferencia y en la generación de predicciones. Aunque los expertos han mostrado cierta preocupación por la falta de interpretabilidad de estos modelos, la aparición de los mecanismos de atención ha ayudado a mitigar dichos temores. Debido a la complejidad e incertidumbre de los procesos climáticos, numerosos modelos de aprendizaje automático emplean técnicas de aprendizaje por conjuntos, que pueden incluir la combinación de distintas arquitecturas o la perturbación de los datos de entrada [de Burgh-Day and Leeuwenburg, 2023].

Esta tendencia se refleja en el reciente lanzamiento por parte del Centro Europeo de Predicción Meteorológica a Medio Plazo (ECMWF) de la versión de conjunto de su Sistema de Predicción de Inteligencia Artificial (AIFS). Esta versión, denominada AIFS ENS, genera predicciones atmosféricas complementarias al modelo numérico tradicional, el

Sistema Integrado de Predicción (IFS). La directora de ECMWF destacó la importancia de este avance, que mejora la eficiencia y capacidad predictiva de los modelos de aprendizaje automático dedicados a la climatología, además de ofrecer un sistema público con soporte continuo y planes de mejora. AIFS ENS produce 51 predicciones distintas que parten de condiciones iniciales perturbadas con el fin de cubrir el mayor número posible de escenarios.

Este conjunto no solo supera a los modelos basados en física para distintas variables, sino que al mismo tiempo permite realizar predicciones hasta diez veces más rápidas. No obstante, su resolución actual es inferior a la de los modelos numéricos, por lo que ECMWF está considerando usar soluciones híbridas que combinen ambos enfoques. Otra de las ventajas de este conjunto es que el consumo energético de las predicciones se reduce aproximadamente mil veces [ECMWF, 2025a].

En la Figura 2.1 se muestra una comparación de la raíz del error cuadrático medio (RMSE) de modelos de aprendizaje automático basados en conjuntos frente a los datos de ERA5 (observaciones atmosféricas), junto con la versión de conjuntos de IFS, llamada IFS ENS.

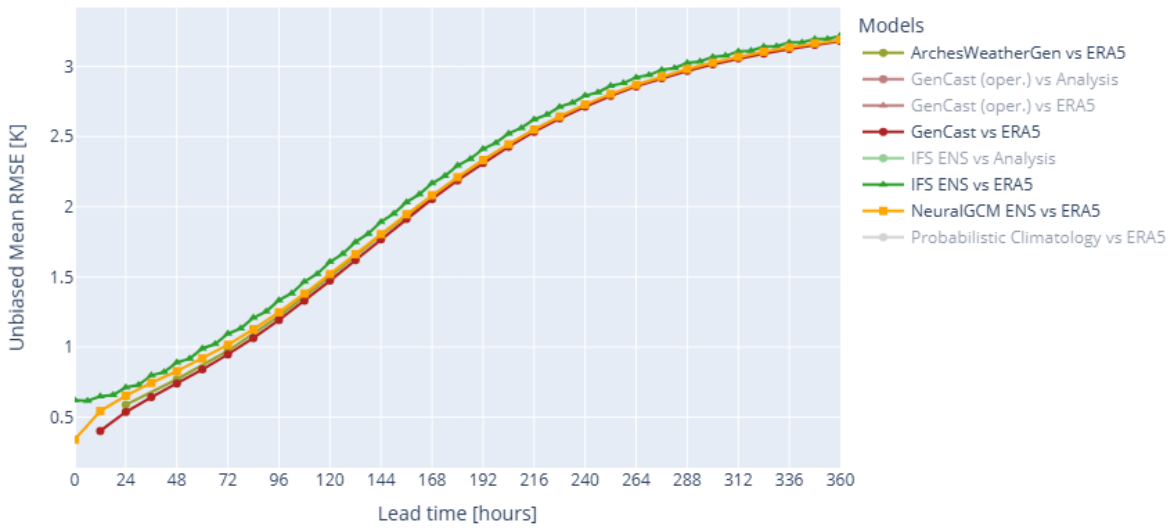


Figura 2.1: Comparación del error cuadrático medio (sin sesgo) de predicciones entre distintos modelos. Fuente: Google Research [Google Research, 2024].

Entre los modelos que participan en la comparación se encuentran:

- **ArchesWeatherGen:** es un modelo desarrollado por el Instituto Nacional Francés de Investigación en Informática y Automatización. Se basa en ArchesWeather, un modelo determinista con una arquitectura transformer diseñado para la predicción de datos del conjunto ERA5. ArchesWeatherGen [Couairon et al., 2024] extiende este enfoque hacia un modelo probabilístico mediante una técnica conocida como *flow matching*. Con esta técnica se entrena un modelo generativo sobre los residuos normalizados obtenidos al promediar cuatro ejecuciones de ArchesWeather. Durante el entrenamiento, se les añade ruido Gaussiano a estos residuos y el modelo aprende a recuperar el residual original. En la fase de inferencia, se parte de ruido Gaussiano puro para generar un residual que se suma a la media de las predicciones de los

modelos deterministas con el fin de predecir el siguiente estado. Se han realizado experimentos con ArchesWeatherGen con 25 y 50 modelos miembros.

- **Gencast:** es un modelo diseñado por DeepMind de Google que realiza predicción probabilística del clima a medio plazo, generando múltiples trayectorias posibles del estado atmosférico. Las predicciones se obtienen mediante una técnica llamada difusión condicional, que ha sido usada especialmente en el tratamiento de audio, vídeo e imágenes en el contexto de la inteligencia artificial generativa. Para su evaluación, GenCast utilizó un conjunto de 50 miembros, comparándose frente al ECMWF ENS. Este modelo será abordado con mayor profundidad, ya que representa un caso de uso relacionado con este trabajo [Price et al., 2023].

- **NeuralGCM:** es un modelo inspirado en los Modelos de Circulación General (GCM), que son simuladores numéricos basados en principios físicos para la predicción climática. NeuralGCM [Kochkov et al., 2024] incorpora componentes de aprendizaje automático que se encargan de modelar los procesos físicos a pequeña escala. Estos componentes calculan las tasas de cambio (tendencias) de diversas variables, integrando sus resultados con los del modelo numérico mediante un solucionador de ecuaciones diferenciales. El conjunto se genera utilizando como entradas ruido extraído de campos Gaussianos aleatorios. El entrenamiento es guiado usando la métrica probabilística *continuous ranked probability score* (CRPS), que es análoga al error medio absoluto determinista. Se presentan resultados con conjuntos que van desde 34 hasta 50 miembros.

Otro modelo de interés en el desarrollo de este trabajo es Pangu-Weather, elaborado por Huawei [Bi et al., 2023a]. Se basa en una arquitectura tridimensional de redes neuronales que incorpora la dimensión de elevación atmosférica. Para mejorar la precisión de las predicciones, emplea un método de agregación temporal que permite reducir errores, obteniendo buenos resultados tanto en predicciones deterministas como probabilísticas, así como en la estimación de temperaturas extremas. Un aspecto especialmente relevante para este trabajo es que puede operar en modo de conjunto con un total de 100 miembros, mediante la perturbación de las condiciones iniciales del modelo añadiendo ruido de Perlin.

Aunque no se trata de un modelo por conjuntos, GraphCast [Lam et al., 2023] es un modelo anterior desarrollado por DeepMind, precursor de GenCast. Toma los dos estados climáticos previos y, con un funcionamiento autorregresivo, las predicciones realizadas por el modelo se emplean como entrada para generar nuevas predicciones. Se implementa con una GNN basada en codificador-procesador-decodificador. La arquitectura de GraphCast se explicará en más detalle dado que es uno de los antecesores de este trabajo.

2.3. Modelos de predicción oceanográfica

Hasta ahora, solo se han mencionado modelos climáticos globales, dado que los modelos oceánicos en gran medida derivan de ellos, los cuales ya incluyen componentes terrestres, oceánicos y atmosféricos.

La referencia en la Unión Europea es el programa Copernicus Marine Service (Servicio Marino de Copernicus), que ofrece predicciones basadas en datos de satélites y sensores, así como un amplio catálogo de productos de libre acceso para los usuarios [Drillet et al., 2025].

Las predicciones de este programa se obtienen mediante modelos numéricos, cuya ejecución está distribuida en distintos centros. Algunos centros se encargan de la asimilación de datos para crear productos refinados y en tiempo real, disponibles para los usuarios, mientras que otros emplean estos datos para obtener predicciones en un horizonte de diez días. Los centros se organizan tanto por áreas geográficas de predicción como según las variables monitorizadas en el tratamiento de datos. Las predicciones se dividen en tres componentes, que son la física, la biogeoquímica y el oleaje. Los dos primeros utilizan el modelo Núcleo para la Modelización Europea del Océano (NEMO), acoplado a otros componentes relacionados con el hielo y el color del océano. El componente de oleaje se predice mediante el Modelo de Oleaje de Météo-France (MFWAM).

Pese a la estructura de Copernicus y sus servicios, también se están comenzando a desarrollar proyectos que aplican métodos de aprendizaje automático. Uno de los ejemplos es la iniciativa Destination Earth (DestinE) [CMEMS, 2023], lanzada por la Unión Europea como parte de la misión para restaurar los océanos, mares y aguas de aquí al año 2030. Esta iniciativa busca desarrollar un gemelo digital de las aguas europeas. El proyecto está siendo liderado por Mercator Ocean International, que se encarga del soporte Copernicus Marine Service (CMEMS). El objetivo es construir un espacio digital alimentado por grandes volúmenes de datos procedentes de los productos de Copernicus. Como resultado, se prevé establecer un marco basado en modelos de aprendizaje automático donde se podrá comprobar el impacto de acciones sobre el ecosistema marino, al tiempo que se monitorizan sus propiedades.

La metodología de conjuntos ha cobrado importancia en la predicción oceanográfica, dado que permite captar la incertidumbre de los procesos oceánicos a pequeña escala. Gran parte de las metodologías de conjuntos existentes provienen del campo de la predicción climática global. Se espera que, en el futuro, los conjuntos de modelos se conviertan en el estándar operacional para la obtención de productos oceanográficos. Estos modelos se caracterizan por mejorar las predicciones a medida que se avanza en el horizonte temporal, en comparación a las predicciones de deterministas, aunque también son claves para obtener estadísticas de error que se integren en los sistemas de análisis oceánicos [Hoteit et al., 2025].

2.4. Introducción a las redes neuronales de grafos

Las GNNs surgen de la necesidad de modelos de aprendizaje profundo que se adapten a la complejidad de datos que no presentan una forma euclídea (como podrían ser texto o imágenes). Es decir, los modelos convencionales de aprendizaje automático están seriamente limitados por haber sido diseñados para funcionar con datos estructurados representados en una rejilla rectangular. Sin embargo, debido a la complejidad del mundo real, existe un número creciente de aplicaciones en las que los datos se representan como grafos, incluyendo ámbitos como el comercio electrónico, las redes sociales, la identificación de moléculas o las citas de artículos científicos. Estos grafos suelen ser irregulares con cantidades variables de nodos, que a su vez poseen distintos atributos y número de conexiones.

A causa de la variedad de configuraciones y aplicaciones de grafos, existen variantes de GNNs con diferentes enfoques. Las principales son las GNNs recurrentes, las GNNs convolucionales, los autoencoders de grafos y las GNNs espaciotemporales [Wu et al., 2020]. La Figura 2.2 muestra la evolución del número de artículos de GNN en los últimos años, lo que indica que se ha convertido en un campo de estudio en auge.

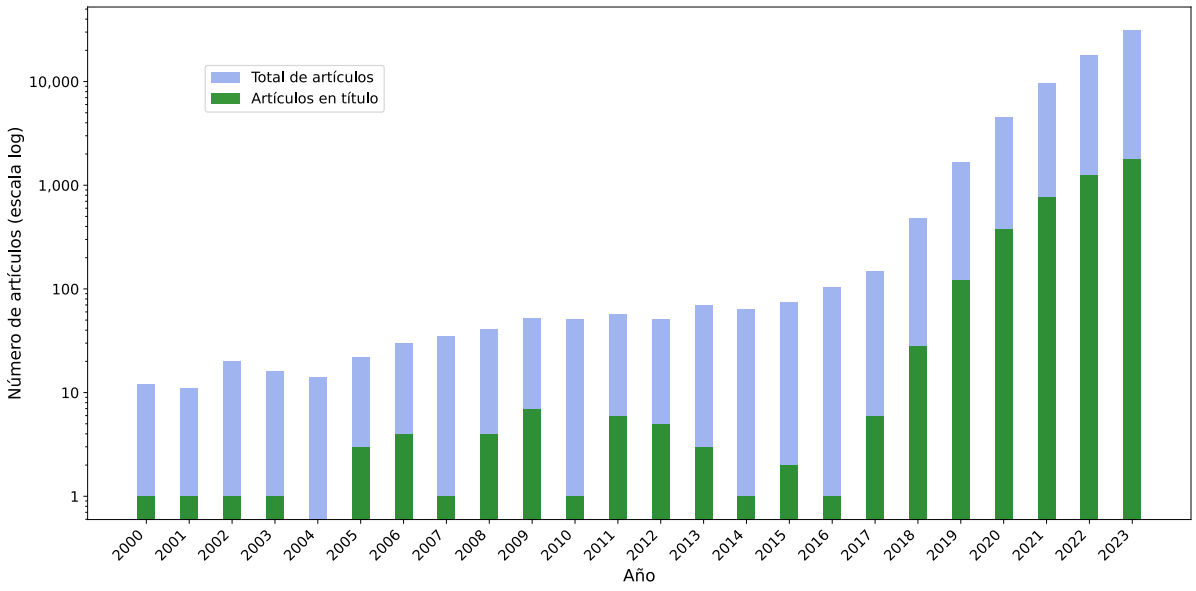


Figura 2.2: Evolución del número de artículos que mencionan "Graph Neural Network" o incluyen el término en el título por año, según Google Scholar.

Una definición sencilla de GNN es la de modelos que aprenden transformaciones sobre los componentes del grafo respetando su simetría estructural, incluyendo la invarianza a permutaciones de los nodos (sin modificar las conexiones y manteniendo las relaciones definidas por la matriz de adyacencia). Los grafos se componen por nodos, aristas (que pueden ser direccionadas o no) y atributos globales, como el número de nodos, el camino más largo y las características del grafo completo. Cada uno de estos componentes implica diferentes tipos de problemas de grafos, que pueden ser a nivel del grafo, de las aristas y de los nodos. La GNN más sencilla consiste en realizar una operación sobre cada componente, por ejemplo, mediante un perceptrón multicapa (MLP), creando una capa de la GNN. De esta forma, se obtiene un nuevo grafo con distintos atributos.

Para crear una red más compleja, es necesario apilar varias capas y recopilar información de los elementos relacionados con un nodo o arista, lo cual se logra mediante la aplicación de funciones de pooling a los atributos de dichos elementos. No obstante, el verdadero potencial de esta arquitectura y la explicación de que su uso sea cada vez más extendido, es el mecanismo de propagación de mensajes. Este mecanismo permite que las representaciones vectoriales (*embedding*) de los nodos o aristas intercambien información durante el procesamiento para actualizarlos.

En su forma más simple, la propagación comienza con la recopilación de las representaciones de los componentes vecinos de un elemento específico, que luego se agregan con una función como podría ser la suma. Este mensaje agregado se introduce en una función de actualización para obtener una nueva representación vectorial para el componente objetivo [Sanchez-Lengeling et al., 2021].

Si el proceso se repite durante varias iteraciones, se consigue que la información de los distintos componentes del grafo se difunda a zonas más alejadas, alcanzando una arquitectura con un contexto mayor. Por ello, la propagación de mensajes se considera el núcleo de las GNNs. Entre las ventajas que ofrece se encuentran la transmisión de información entre distintas regiones a lo largo del grafo, la posibilidad de dividir y paralelizar el problema, así como la capacidad de aplicarse en grafos de distintos tamaños y formas

bajo un procedimiento común [Battaglia et al., 2018].

Modelos específicos a la predicción oceanográfica

Gran parte de las aplicaciones de aprendizaje profundo en oceanografía se basan en redes convolucionales. Sin embargo, estas redes no pueden capturar adecuadamente la geografía terrestre, debido a que utilizan rejillas de tamaño fijo. En todo el mundo existen accidentes geográficos como islas, bahías, canales y estrechos que generan una forma irregular que puede ser captada mediante estructuras de grafos con suficiente resolución.

Bajo esta premisa, surgió el modelo Graph Memory Neural Network (GMNN), donde se combinan GNNs (para modelar relaciones espaciales) con redes neuronales de memoria a corto y largo plazo (LSTM - *Long Short Term Memory*) para predecir la evolución de la temperatura superficial del océano (SST - *Sea Surface Temperature*) [Liang et al., 2023]. Este modelo emplea múltiples capas de GNN que, de manera iterativa, extraen relaciones espaciales considerando tanto información de los nodos como de las conexiones. Otra de sus características innovadoras frente a anteriores modelos es que puede trabajar con áreas incompletas, donde hay datos faltantes por la irregularidad del terreno, que se rellenan usando información del grafo. Las estimaciones devueltas por el decodificador, basado en capas totalmente conectadas, mostraron una mayor exactitud y estabilidad que modelos anteriores al aplicarse sobre la región del Pacífico noroeste. Las pruebas se llevaron a cabo con distintas escalas temporales y horizontes de predicción.

Las GNNs también se han aplicado en tareas de predicción compleja, como la Tomografía Acústica Costera (CAT - *Coastal Acoustic Tomography*). Esta técnica se basa en la velocidad de la propagación del sonido para estimar, de forma inversa, atributos dinámicos del océano. Para su funcionamiento se requiere una red de emisores y receptores que pueden ser representados como nodos, mientras que las transmisiones entre ellos se modelan como aristas. GraphSAGE [Xu et al., 2024] ha demostrado el rendimiento de las GNNs en la predicción indirecta de datos oceanográficos. Su arquitectura consiste en una GNN convolucional espacial, en la cual cada nodo actualiza sus características a partir de operaciones convolucionales con información de su vecindad. Esta información se obtiene mediante un muestreo donde se selecciona aleatoriamente un subconjunto de vecinos, cuya información se recopila en un único vector de características agregado. GraphSAGE ha alcanzado errores medios bajos tanto en la predicción de la velocidad del sonido como de la temperatura, confirmando su capacidad de generalización y uso en aplicaciones de tiempo real.

2.5. Aprendizaje por conjuntos

Los modelos climáticos basados en aprendizaje profundo suelen ser criticados por su limitada capacidad para representar la incertidumbre en las predicciones. En climatología, es habitual obtener un rango de posibles resultados considerando distintas fuentes de incertidumbre, mediante el uso de conjuntos de predicciones o modelos (también llamados *ensembles*) [de Burgh-Day and Leeuwenburg, 2023].

El aprendizaje por conjuntos (*ensemble learning*) es una técnica que agrupa un conjunto de métodos basados en la combinación de múltiples modelos "débiles" para generar una predicción. Los modelos base que componen el conjunto se entrenan de forma individual con un conjunto de datos, para ser utilizados posteriormente en fase de inferencia. Esta estrategia se relaciona con el concepto de sabiduría de la multitud, donde se espera que los

errores individuales de los modelos base se compensen entre sí, obteniendo una predicción generalmente mejor que la de un único modelo. Las salidas se combinan mediante el uso de una función de agregación que unifique los resultados [Sagi and Rokach, 2018].

Para que un conjunto de modelos sea capaz de superar al rendimiento de un modelo individual, es necesario cumplir con ciertos requisitos centrados en el concepto de diversidad. La condición principal es que los modelos empleados deben tener mejor rendimiento que un modelo aleatorio y sus errores sean, en la medida de lo posible, independientes. Es decir, que dos modelos no cometan el mismo error ante una misma entrada. Si se cumple esta condición, el conjunto de modelos debería ser más fiable que cualquiera de los modelos individuales que lo compone [Dietterich, 2000].

El aprendizaje por conjuntos funciona, entre otras razones, por fundamentos estadísticos. Cuando se dispone de un conjunto de datos limitado, el espacio de modelos posibles es más amplio. Al combinar varios modelos, se reduce el riesgo de seleccionar un modelo incorrecto y se mitiga el sobreajuste [Dietterich, 2000, Sagi and Rokach, 2018].

La Figura 2.3 muestra cómo la irrupción del aprendizaje profundo ha impulsado la investigación y el uso de las técnicas de aprendizaje por conjuntos.

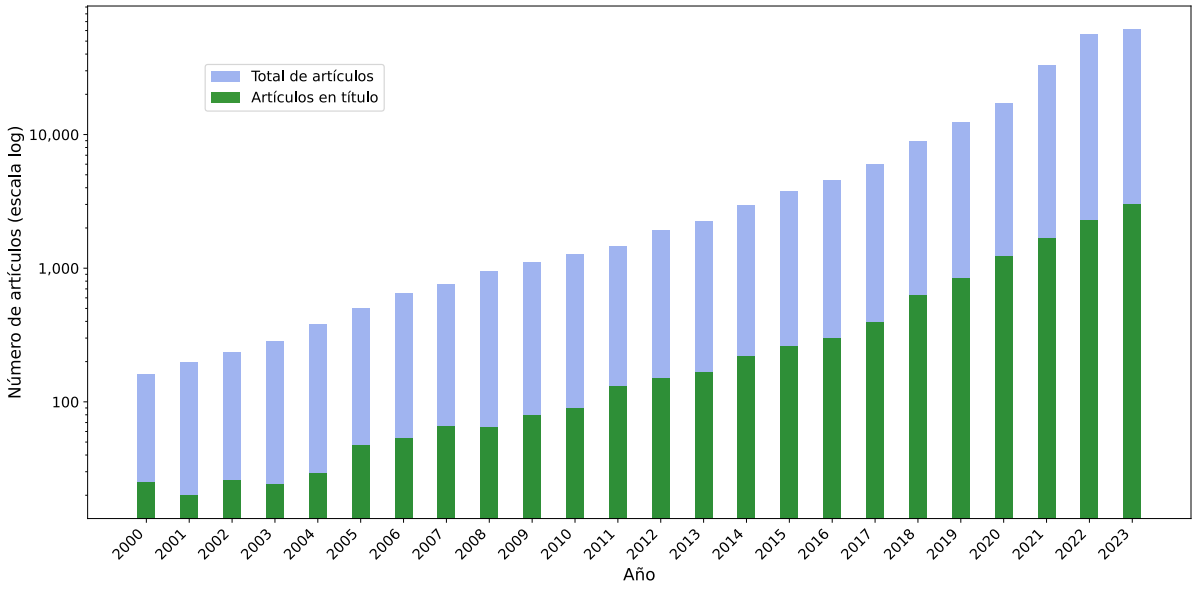


Figura 2.3: Evolución del número de artículos que mencionan “Ensemble Learning” o incluyen el término en el título por año, según Google Scholar.

Uno de los principales focos de investigación consiste en adaptar modelos modernos, como las redes neuronales profundas, para maximizar su rendimiento con técnicas de aprendizaje por conjuntos. Cabe destacar que estas redes suelen contener millones de parámetros, lo que hace que su entrenamiento y evaluación mediante técnicas de conjuntos sean computacionalmente costosos [Yang et al., 2023]. Por ello, es fundamental prestar especial atención al diseño y selección de las técnicas utilizadas para conformar el conjunto.

Los modelos base se denominan “aprendices débiles” (*weak learners*) porque no es necesario que sean resultado de una búsqueda de hiperparámetros exhaustiva ni que tengan arquitecturas muy sofisticadas, lo cual sería muy costoso. Como se mencionó previamente, basta con que estos modelos tengan un rendimiento mejor que un predictor aleatorio para que su combinación pueda lograr un rendimiento aceptable. Sin embargo, aunque

los algoritmos empleados pueden ser relativamente simples, es necesario que exista diversidad entre ellos. La diversidad del conjunto se puede alcanzar mediante dos estrategias principales.

La primera es la estrategia heterogénea, en la cual los modelos base no parten de una misma arquitectura, siendo la técnica más común el Stacking. Por otro lado, se encuentra la estrategia homogénea, donde los modelos base parten de una misma arquitectura y solo podrían presentar cambios en los parámetros. Es la estrategia más popular debido a su menor coste computacional, ya que los modelos comparten la misma arquitectura, por lo que basta con entrenar un solo modelo. Para garantizar la diversidad en este segundo caso, se recurre a la modificación de la entrada de los distintos modelos base.

En problemas de regresión el método más común de agregación de conjuntos es el promedio de las salidas de los modelos, con lo que se obtiene la predicción final del conjunto. Otra solución es combinar las salidas de los modelos con una suma ponderada, ajustando los pesos en base al error cometido por cada modelo. Estos pesos se refinan minimizando una función objetivo. En el caso de modelos diseñados para clasificación, la decisión de los modelos se toma mediante un mecanismo de votación. El más simple es elegir la clase que la mayoría haya predicho, aunque también existen variantes ponderadas de forma similar a los problemas de regresión [Yang et al., 2023].

A continuación, se explica brevemente cada una de las estrategias más comunes de aprendizaje por conjuntos junto con sus ventajas.

2.5.1. Bagging

El Bagging (*bootstrap aggregating*) es una estrategia homogénea que consiste en crear varios subconjuntos desde un conjunto de datos original aplicando muestreo aleatorio con sustitución, lo que permite que algunas muestras se repitan mientras otras quedan fuera. Esto permite entrenar cada modelo de forma individual y paralela con entradas distintas [Yang et al., 2023]. Una vez entrenados, se combinan sus predicciones, lo que reduce la varianza del modelo y evita el sobreajuste. El Bagging ha demostrado ser especialmente eficaz en modelos con predicciones inestables. Un ejemplo de su aplicación son los árboles de decisión con el algoritmo Random Forest, donde se introduce aleatoriedad tanto en la selección del subconjunto de datos como en las variables para cada árbol, con el objetivo de garantizar la diversidad y mejorar la generalización [Rout, 2024].

También existe la posibilidad de realizar manipulaciones de datos de formas no estandarizadas, dependiendo del contexto específico. Por ejemplo, en la predicción de variables oceanográficas, es frecuente que ciertos sensores introduzcan ruido, por lo que se asume cierta incertidumbre en las observaciones. Esta incertidumbre puede ser representada con un cierto intervalo donde se espera que varíen los valores observados. Estudios sobre la incertidumbre en el Golfo de Vizcaya [Vervatis et al., 2025] muestran técnicas de parametrización estocástica que potencialmente podrían ser empleadas para generar conjuntos de entrada para distintos modelos base. Pese a la relación con el aprendizaje por conjuntos, el propósito del trabajo mencionado es descubrir la configuración óptima para describir las incertidumbres presentes en observaciones reales.

2.5.2. Boosting

El Boosting es una estrategia homogénea de aprendizaje por conjuntos, donde los modelos base se crean de forma secuencial. En cada iteración, el nuevo modelo corrige los

errores cometidos por el anterior, aumentando los pesos de las observaciones que fueron predichas incorrectamente [González et al., 2020]. A diferencia del Bagging, el objetivo del Boosting es reducir el sesgo, asegurando que los modelos se ajusten a los datos para mejorar la precisión. Entre los algoritmos de Boosting más conocidos se encuentran AdaBoost, Gradient Boosting Machines, XGBoost y LightGBM.

Aunque tanto Bagging como Boosting buscan obtener un conjunto de modelos que, combinados, generen predicciones más consistentes y precisas, difieren en ciertos aspectos. El Boosting suele ser aplicado en casos donde se trabaja con datos complejos, escasos o difíciles de modelar, por lo que es necesario que el modelo se adapte a los datos de la mejor manera posible para reducir el sesgo. Por otro lado, el Bagging se enfoca en la estabilidad de las predicciones, de forma que no varíen excesivamente para reducir la varianza. Por último, el proceso de entrenamiento en Boosting es secuencial, mientras que en Bagging los modelos pueden ser entrenados en paralelo [Kalirane, 2025].

2.5.3. Stacking

El Stacking es un método que introduce el concepto de metamodelo y suele trabajar con modelos heterogéneos. Un metamodelo es un bloque que se coloca tras los modelos base, y que toma todas sus salidas para generar un único resultado. Mientras los modelos base aprenden de los datos de entrada, el metamodelo aprende de las salidas de los modelos base. Los metamodelos pueden ser tan sencillos como la regresión lineal o una MLP, y pueden mejorar la habilidad de abstracción con una gran cantidad de datos, a diferencia de los métodos de unificación clásicos que no son capaces de aprovechar toda la información [Yang et al., 2023].

A diferencia de las dos estrategias anteriores, en Stacking la diversidad se introduce con arquitecturas de modelos distintas. Al juntar modelos con diferentes puntos fuertes, se compensan mutuamente sus puntos débiles y se obtiene una mayor flexibilidad en la elección de modelos. En cambio, el entrenamiento de los modelos puede ser más costoso computacionalmente, dado que no se parte de un modelo común para generar el conjunto, como sí ocurre en el caso Bagging. Por el mismo motivo, resulta complicado realizar una búsqueda de hiperparámetros sobre los modelos [Rout, 2024]. Combinar Stacking con técnicas de ponderación puede mejorar la capacidad de abstracción del modelo, aunque con un mayor coste computacional en función de la cantidad de modelos base [Omari and Figueiras-Vidal, 2015].

Una de las variantes más empleadas de Stacking se conoce como mezcla de expertos. Consiste en entrenar a cada modelo para tratar un subproblema distinto, de forma que cada uno se especialice en cierta información. Tras ello, un metamodelo aprende de las salidas de los modelos comparándolos con las observaciones reales para reducir el error [Yang et al., 2023]. Las arquitecturas pueden incluir una red neuronal separada (llamada red de puerta) que, dados unos datos de entrada, es capaz de elegir los submodelos especializados y asignarles pesos en función de cómo deben contribuir al resultado final. Este planteamiento es capaz de reducir el coste computacional, ya que, en lugar de usar todos los modelos para determinar el resultado, solo se usa un subconjunto de ellos [Dutta, 2025].

Finalmente, se considera que la combinación del aprendizaje por conjuntos con el aprendizaje profundo representa una gran innovación en la predicción climática. En comparación con los conjuntos de modelos numéricos tradicionales, se conseguiría un gran número de modelos base con un coste computacional menor, tiempos de generación más

rápidos y una mejor calibración [CW3E, 2025].

2.6. Aprendizaje por conjuntos aplicado a redes neuronales de grafos

Las GNNs se caracterizan por su rendimiento en tareas con datos estructurados en forma de grafo, debido a la complejidad de los modelos. Como consecuencia, se suelen ver afectadas por el sobreajuste, lo que genera un bajo rendimiento con datos no vistos. Usar distintas inicializaciones y arquitecturas de GNNs permite mejorar la generalización de los modelos mediante el aprendizaje por conjuntos [Wong et al., 2023]. Debido a la complejidad de los modelos, es especialmente relevante el principio "muchos pueden ser mejor que todos" [Zhou et al., 2002]. Realizar una selección efectiva de modelos, en lugar de utilizarlos todos, suele ser una decisión mejor.

En este principio se basa Ensemble Learning for Graph Neural Networks (ELGNN - Aprendizaje por conjuntos para redes neuronales de grafos) [Wong et al., 2023], un enfoque en el que se generan múltiples modelos teniendo en cuenta su rendimiento sobre el conjunto de datos de `ogbl-ddi`, utilizado para la evaluación de GNNs, y sus diferencias arquitectónicas. Se aprovechan tanto las características estructurales del grafo como la información de los nodos, con el objetivo de identificar pares de interacciones farmacológicas reales frente a falsas. El aprendizaje por conjuntos se aplica en la capa de predicción de cada modelo, lo que permite conservar las particularidades de cada uno y mejorar la capacidad predictiva. Uno de los aspectos más destacables de este proyecto es la eficiencia en el entrenamiento gracias a la paralelización, ya que cada modelo se entrena y se emplea en fase de inferencia de forma independiente. Las diversas pruebas sobre `ogbl-ddi` han mostrado que supera a cada uno de los modelos individuales y se ha consolidado como una nueva referencia en tareas de predicción de enlaces.

Otro caso interesante de combinación de redes neuronales de grafos con aprendizaje por conjuntos es GNN-Ensemble [Wei et al., 2023]. La solución aplicada consiste en dividir el problema en subgrafos que se usan como entrada para entrenar los modelos, una estrategia alineada con el enfoque de Bagging y mezcla de expertos. Este enfoque se centra en generar modelos base que aprendan sobre distintas partes del grafo, utilizando como entradas subgrafos aleatoriamente muestreados, así como subconjuntos de las características de los nodos. Posteriormente, el conocimiento se combina mediante votación entre los distintos algoritmos base durante la predicción. Los resultados confirmaron que el modelo es fácilmente paralelizable y fue capaz de superar a modelos de referencia en tareas de clasificación de nodos gracias a su efectividad para reducir el sobreajuste. Además, mostró resistencia frente a ataques de adversario, que supone una amenaza para muchas GNNs.

2.7. Introducción a SeaCast

La estructura sobre la cual se construye este trabajo es el proyecto SeaCast. SeaCast surge como respuesta a la falta de métodos específicamente dirigidos a la predicción en el ámbito marino. Se trata de un modelo autorregresivo basado en GNNs, diseñado para el pronóstico a medio plazo del estado de cuerpos acuáticos regionales. El modelo tiene como características principales la utilización de grafos para representar la geometría irregular del océano, junto con forzantes atmosféricos y límites oceánicos para mejorar las predicciones. Fue entrenado con 37 años de datos del Mar Mediterráneo (35 años

de reanálisis y 2 de análisis) y sus resultados han sido validados frente a los modelos numéricos de CMEMS [Holmberg et al., 2024]. En los siguientes capítulos se analiza en mayor profundidad la arquitectura de SeaCast, basada principalmente en Neural-LAM, explicando su funcionamiento y su relación con otros modelos como GraphCast y el modelo de grafos de Keisler [Keisler, 2022].

2.8. Limitaciones actuales

A pesar del potencial de las GNNs, cuentan con una serie de inconvenientes. Uno de ellos es que es posible que dos nodos con vecindades distintas utilicen el mismo procedimiento para generar su representación vectorial, lo que los vuelve indistinguibles. Este problema afecta tanto a tareas de clasificación de nodos (incapacidad de distinguir nodos en grafos regulares en los que cada nodo tiene el mismo número de vecinos), de predicción de conexiones (no pueden diferenciar nodos con vecindarios idénticos pero distintas distancias al nodo objetivo) y de clasificación de grafos (no pueden diferenciar grafos regulares cuyos nodos tienen el mismo número de vecinos) [You et al., 2021].

El mecanismo de paso de mensajes, núcleo de las GNNs, también presenta ciertas limitaciones. La principal es que una gran cantidad de capas tiende a generar representaciones similares para cada nodo, lo que afecta a la capacidad del modelo para diferenciarlos. Además, el incremento en el número de capas incrementa el coste computacional de la inferencia, lo que dificulta su uso en aplicaciones de tiempo real y compromete la escalabilidad. Por otro lado, el mecanismo actual suele fallar en capturar relaciones de carácter global en el grafo [Zhang et al., 2025].

El aprendizaje por conjuntos de las GNNs también ha sido objeto de estudio debido a las particularidades que presenta. Una de las peculiaridades es la estructura jerárquica de estas redes, la cual se ve ignorada por los algoritmos tradicionales de aprendizaje por conjuntos, ya que las salidas de la GNN solo se combinan tras la última capa. Para abordar esta limitación, se puede realizar el proceso en dos pasos. En primer lugar, se manipula la estructura de la red, conservando las conexiones dentro de cada nivel, pero eliminando aquellas que conectan distintos niveles. A continuación, se incorporan modelos base entre cada capa de la red, combinando sus salidas antes del paso de mensajes [Duan et al., 2024].

Capítulo 3

Metodología y planificación del trabajo

En los últimos años, la evolución de los proyectos de ciencia de datos ha dado lugar a la creación de marcos de trabajo comunes, aplicables a distintos contextos. Estos proyectos suelen enfrentarse a una falta de objetivos claros, dificultad de establecer estimaciones temporales, altos costes y problemas de comunicación entre equipos. El objetivo de las metodologías es aportar un marco integral que guíe los procesos orientados por los datos, facilite su reproducibilidad, se adapte a largos ciclos de trabajo y permita gestionar la escalabilidad. Además, contribuyen a mejorar los controles de calidad sobre los datos, la interpretación de resultados y la retención del conocimiento [Martinez et al., 2021].

A lo largo de este capítulo se describen las fases de trabajo realizadas durante el proyecto. En primer lugar, se introduce Cross-Industry Standard Process for Data Mining (CRISP-DM), una de las metodologías más empleadas en ciencia de datos, explicando cómo se ha adoptado en este proyecto. Tras ello, se revisa la planificación inicial, incluyendo las tareas ejecutadas y su duración estimada, así como las modificaciones llevadas a cabo a lo largo del desarrollo del proyecto. Por último, se enumeran los recursos empleados, tanto hardware como software, así como las librerías especializadas utilizadas.

3.1. Metodología

CRISP-DM surgió en 1996 por iniciativa de empresas pioneras en el campo de la minería de datos [Chapman et al., 2000], con el objetivo de desarrollar un modelo documentado libre que permitiera a las organizaciones iniciar sus propios proyectos de minería de datos dentro de un marco de buenas prácticas [Shearer, 2000]. La Figura 3.1 muestra las seis fases de CRISP-DM.

La fase de Comprensión del Negocio comenzó con la revisión del artículo del modelo SeaCast, desarrollado para la predicción de variables oceanográficas y que, en este proyecto, se adapta a la zona del Atlántico Norte. Para comprender su funcionamiento, se realizaron pruebas con los datos originales del mar Mediterráneo siguiendo la documentación. Como objetivo inicial, se planteó reducir los errores de las predicciones individuales haciendo uso de un conjunto de varios modelos. Se espera que el efecto sea más notable a medida que aumenta el horizonte de la predicción.

En la fase de Comprensión de Datos se exploraron los productos disponibles en CMEMS para la región del Atlántico Norte. Tras la selección del conjunto de datos, se eligió la variable SST como objeto del estudio. A lo largo de esta etapa, también se incorporaron

otros productos complementarios, como la batimetría y forzantes externos obtenidos del Servicio de Cambio Climático de Copernicus (*Copernicus Climate Change Services*). Por último, se analizó el formato y estructura de los datos, junto con una primera exploración para verificar su integridad.

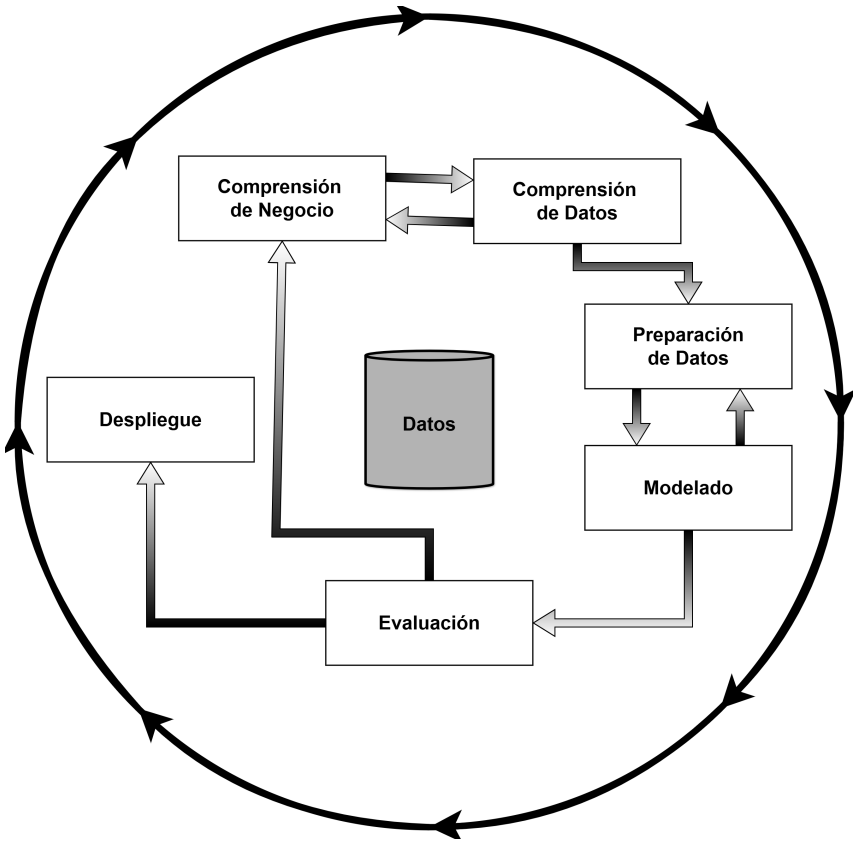


Figura 3.1: Fases de CRISP-DM adaptadas. Fuente: [Shearer, 2000].

La fase de Preparación de Datos se centró en adaptar los datos obtenidos a las necesidades del proyecto. En primer lugar, se delimitaron los datos recortándolos según las coordenadas que definen la zona de estudio. El código de procesamiento de datos de SeaCast fue modificado para ajustarlo a las particularidades de este trabajo. Una de las diferencias más destacables es que el código original está diseñado para distintos niveles de profundidad, mientras los datos utilizados solo abarcan un nivel, lo que afecta a las dimensiones. Esta diferencia provocó que fuera necesario crear una máscara binaria tierra-océano, al igual que modificar la descarga de datos externos como la batimetría, las condiciones externas y los pesos basados en el coseno de la latitud. Finalmente, se prescindió de ciertos datos utilizados en el modelo original.

Como parte de la fase de Modelado se estudiaron técnicas de aprendizaje por conjuntos, con especial atención en aquellas utilizadas en predicción climática, así como en GNNs. Durante esta etapa, se concluyó que la opción más viable sería emplear la estrategia de Bagging sobre los datos oceanográficos, dado que entrenar modelos independientes en varias ocasiones resultaría bastante costoso computacionalmente y requeriría un tiempo considerable. En su lugar, la diversidad se introduce con ruido en los estados iniciales durante la fase de inferencia, generando predicciones a partir de datos de entrada distintos, las cuales se agregaron posteriormente para obtener el resultado final. Se recopilaron diversos tipos de ruido y estrategias relacionadas con conjuntos durante la evaluación.

Finalmente, tuvo lugar una selección de técnicas sobre las que se realizaron pruebas para evaluar su rendimiento. La fase de Evaluación comenzó con pruebas preliminares realizando predicciones a partir de un solo día. El aprendizaje por conjuntos se caracteriza por el uso de métricas probabilísticas para su evaluación, entre las que se incluyen el CRPS, el *spread-skill ratio* (relación dispersión-habilidad) y el RMSE sin sesgo. Estas métricas se adaptan desde la herramienta de evaluación de WeatherBench-X. Asimismo, se evaluaron los resultados combinados mediante métricas deterministas tradicionales, como el RMSE y el sesgo. Una vez analizado el rendimiento de los modelos, se seleccionó un subconjunto para realizar pruebas con el conjunto completo de datos de evaluación, obteniendo resultados y conclusiones más sólidas.

Como último paso, tendría lugar la fase de Despliegue, en la que se integraría el sistema en un entorno real. Aunque el modelo es funcional, al tratarse de una adaptación de un modelo que ya ha sido validado, sería necesario considerar sus limitaciones, especialmente en cuanto a datos y ajustes de entrenamiento, y muy particularmente al número de predicciones del conjunto. Los conjuntos de modelos, tanto en modelos numéricos como de aprendizaje automático, suelen componerse de un gran número de miembros, lo cual resulta poco práctico en este proyecto dados los recursos disponibles. Por tanto, este enfoque debería considerarse como una base sobre la cual se podría elaborar un modelo más complejo. Dicho modelo podría ser integrado en soluciones para apoyar la toma decisiones en sectores relacionados con la economía azul.

3.2. Planificación

La definición de una planificación inicial fue de gran importancia para poder trabajar de forma estructurada y guiada en el proyecto. El trabajo se distribuyó en cuatro etapas, que agrupan en su mayoría las fases contempladas en CRISP-DM, tal y muestra la Tabla 3.1, la cual incluye sus respectivas tareas y duración estimada.

Fases (Duración estimada en horas)	**Tareas**
Comprensión de Negocio / Comprensión de Datos (90)	Tarea 1.1: Definir el problema y objetivos
	Tarea 1.2: Revisión de la literatura
	Tarea 1.3: Recolección de datos
	Tarea 1.4: Análisis exploratorio de los datos
Preparación de Datos / Modelado (300)	Tarea 2.1: Extracción de variables y limpieza
	Tarea 2.2: Transformación de datos
	Tarea 2.3: Diseño de los métodos
	Tarea 2.4: Diseño de prueba
	Tarea 2.5: Entrenamiento de los modelos
Evaluación / Despliegue (130)	Tarea 3.1: Validación del rendimiento de modelos
	Tarea 3.2: Comparación de rendimiento
Documentación / Presentación (80)	Tarea 4.1: Elaboración de memoria
	Tarea 4.2: Preparación de la exposición del proyecto

Tabla 3.1: Fases y tareas planificadas del proyecto.

En cuanto al primer bloque de Comprensión de Negocio y Datos, las tareas giraron en torno a familiarizarse con el modelo y los datos sobre los cuales se trabajaría. La previsión inicial fue de unas 90 horas, dado que, pese a tener que investigar en profundidad SeaCast y los métodos de aprendizaje por conjuntos, el resto de la fase consistió en descargar los datos necesarios y realizar un análisis exploratorio de ellos, que resultó más breve.

La siguiente fase de Preparación de Datos y Modelado tenía una duración prevista de 300 horas, siendo la más extensa. Las tareas incluidas abarcan la extracción de variables y la transformación de datos, que en cierta medida se solapan con las dos últimas de la fase anterior. Por un lado, durante la descarga de datos ya se definían las características del producto deseado, mientras que la transformación de datos se apoyaba en los resultados del análisis exploratorio, adaptándose a las dimensiones esperadas por el modelo. La mayor parte del tiempo de esta fase se dedicó al diseño de métodos de aprendizaje por conjuntos, al diseño de pruebas y al entrenamiento de modelos. Durante este tramo estaba previsto entrenar el modelo de forma paralela, sin embargo, los prolongados tiempos de ejecución impidieron llegar a este escenario. De igual manera, tampoco se pudo realizar una búsqueda de hiperparámetros. No obstante, si se logró generar predicciones a la vez que se entrenaban modelos, ya que la fase de inferencia resulta menos costosa computacionalmente. Finalmente, se descartó usar la alternativa de Stacking, ya que implicaba utilizar distintas arquitecturas de modelos, lo que nuevamente requeriría el entrenamiento individual de modelos.

La etapa siguiente, tanto en el orden de planificación como en la previsión de tiempo (130 horas), es la de Evaluación y Despliegue, donde se validó el rendimiento de los modelos y se compararon las diferentes configuraciones definidas. En este caso, no se pudo aplicar validación cruzada debido a que requería entrenar el modelo en distintas iteraciones. Además, la naturaleza temporal de los datos complica la división del conjunto de datos para esta técnica, dado que podría ocasionar pérdida de información. Esta fase está estrechamente relacionada con la anterior, ya que una vez analizado el rendimiento de los modelos con un subconjunto de datos, se diseñaron nuevas pruebas con el conjunto de datos completo.

La última fase fue la Documentación y Presentación, que se estimó en 80 horas de redacción de memoria y preparación de la exposición del proyecto. Aunque esta etapa se sitúa al final del proyecto, la documentación fue elaborándose a lo largo de este. No obstante, fue en esta fase donde se centraron los esfuerzos para incluir las comparaciones y resultados. La Tabla 3.2 muestra las tareas realizadas tras los cambios descritos anteriormente.

Fases (Duración estimada)	**Tareas**
Comprensión de Negocio / Comprensión de Datos (90)	Tarea 1.1: Definir el problema y objetivos
	Tarea 1.2: Revisión de la literatura
	Tarea 1.3: Recolección de datos
	Tarea 1.4: Análisis exploratorio de los datos
Preparación de Datos / Modelado (300)	Tarea 2.1: Extracción de variables y limpieza
	Tarea 2.2: Transformación de datos
	Tarea 2.3: Diseño de los métodos (usando Bagging)
	Tarea 2.4: Diseño de prueba
	Tarea 2.5: Entrenamiento de los modelos (sin búsqueda de hiperparámetros ni paralelización)
Evaluación / Despliegue (130)	Tarea 3.1: Comparación de rendimiento basado en datos de prueba
Documentación / Presentación (80)	Tarea 4.1: Elaboración de memoria
	Tarea 4.2: Preparación de la exposición del proyecto

Tabla 3.2: Fases y tareas desarrolladas en el proyecto.

3.3. Recursos empleados

En esta sección se detallan los recursos hardware, software y las librerías específicas utilizadas para el desarrollo del trabajo. De la misma forma, se indican otras herramientas que facilitaron el flujo de trabajo.

3.3.1. Hardware

Uno de las partes más importantes del proyecto era el entrenamiento del modelo, para lo cual se necesitó un equipo capaz de ejecutar el entrenamiento con un gran volumen de datos. Debido a esta necesidad, se usó durante la fase de entrenamiento un equipo del Laboratorio de Investigación número 2 del Edificio de Informática y Matemáticas, perteneciente a la Universidad de Las Palmas de Gran Canaria. El equipo dispone de un procesador Intel Core i9-9900K con una frecuencia de 3.60 GHz, 8 núcleos físicos y 16 hilos lógicos. En cuanto a la tarjeta gráfica, posee una NVIDIA GeForce RTX 3060 con 12 GB de VRAM, que fue el componente más clave para el entrenamiento gracias a su capacidad de procesamiento acelerado. Por último, el equipo cuenta con 32 GB de RAM, necesarios para cargar los grandes conjuntos de datos en memoria.

Para la inferencia se usó un portátil personal, dado que el uso del modelo en fase de prueba consume menos recursos que durante el entrenamiento. Su procesador es un Intel Core i7-8750H a 2.20 GHz, con 6 núcleos físicos y 12 hilos lógicos. En cuanto a la GPU, dispone de una NVIDIA GeForce GTX 1050 de 4 GB de VRAM. Por último, el equipo cuenta con 16 GB de RAM.

3.3.2. Software

En cuanto al software, ambos equipos tienen instalado el sistema operativo Windows 10 (el de sobremesa en su versión Enterprise y el portátil en su versión Pro). El lenguaje

de programación utilizado fue Python en su versión 3.10.16.

3.3.3. Librerías

A lo largo del proyecto se emplearon librerías de Python orientadas a distintos aspectos del código, de las cuales se destacan las más relevantes. Para el entrenamiento se usó `pytorch-lightning`, una variante de `pytorch` que permite mayor flexibilidad, automatización y control sobre el entrenamiento. La librería `torch-geometric` resultó clave, ya que contiene funciones especializadas para trabajar con GNNs. Los grafos de estas redes fueron creados mediante el paquete `networkx`. Para la descarga de datos mediante una interfaz de programación de aplicaciones (API) se emplearon `copernicusmarine` (datos oceanográficos) y `ecmwf-opendata` (forzantes externos atmosféricos). Para el manejo de los datos multidimensionales descargados, se usó `xarray`, así como `numpy` para operar con los datos una vez procesados. El código de evaluación fue desarrollado utilizando métricas probabilísticas y deterministas con el framework `weatherbenchx`. Por último, para la visualización de datos y gráficas se optó por `Matplotlib`.

3.3.4. Otras herramientas utilizadas

Para la gestión del proyecto y control de versiones se utilizó Git (local) y Github (remoto), permitiendo garantizar la trazabilidad del código y la integridad del proyecto en todo momento. Para aprovechar la capacidad de cómputo de las tarjetas gráficas, se instaló CUDA Toolkit, lo que aceleró los entrenamientos. La ejecución de código de evaluación y comparación de métricas se realizó usando Jupyter Notebooks, facilitando la visualización de gráficas. Por último, para monitorizar los entrenamientos y su rendimiento en tiempo real, se conectaron las ejecuciones con Weights & Biases. Finalmente, la memoria fue escrita en LaTex mediante el editor web Overleaf.

Capítulo 4

Conjuntos de datos

En este capítulo se explican las fuentes de datos empleadas, así como las características de los datos descargados. Para el entrenamiento del modelo, se utilizaron tres conjuntos de datos:

- Datos del Atlántico de CMEMS: se obtiene un producto con la temperatura superficial del océano (SST) reprocesada, que abarca desde la zona del Mar del Norte hasta la costa senegalesa sin incluir Cabo Verde. Los valores de este conjunto de datos son aquellos sobre los que se realizarán predicciones [CMEMS, 2023].
- Datos de forzantes: una de las aproximaciones del proyecto original es usar una serie de variables atmosféricas para mejorar el rendimiento del modelo. ERA5 permite obtener los componentes vectoriales u y v de la velocidad del viento. Estos datos se obtienen del Copernicus Climate Data Store [Hersbach et al., 2025].
- Datos de profundidad: la batimetría se utiliza como una constante para cada punto de nuestra región, representando una medida de la profundidad del océano. Los datos empleados para obtener la batimetría provienen del modelo de relieve global ETOPO [NOAA, 2022a].

Durante este capítulo también se justifican las decisiones de diseño tomadas respecto al uso de determinadas variables, señalando las diferencias respecto a SeaCast. Asimismo, se comenta la inclusión de ciertos productos que sirven como apoyo para el procesamiento, aunque no se descargan directamente desde fuentes externas. Por último, se detalla la estrategia de división de datos empleada para facilitar un entrenamiento eficaz del modelo.

4.1. Fuentes de datos

En esta sección se describen las principales fuentes de datos utilizadas en este proyecto. Estas corresponden al Copernicus Marine Service, el Copernicus Climate Data Service y la Administración Nacional Oceánica y Atmosférica, tres instituciones internacionales con gran responsabilidad en la provisión de información climática.

4.1.1. Copernicus Marine Service

El primer paso, tras definir los objetivos y el problema, es obtener las variables base del estudio. En este caso, la variable objetivo a predecir es la SST. Los datos se obtienen desde Copernicus Marine Service (CMEMS).

CMEMS forma parte del programa Copernicus, una iniciativa de la Unión Europea dedicada a la observación de la Tierra. Su misión es proporcionar información de nuestro planeta y su entorno para el beneficio de todos los ciudadanos. Esta información proviene de satélites, datos in situ obtenidos mediante sensores (no espaciales) y modelos numéricos.

Dentro de este marco, CMEMS se encarga de proveer datos sobre el océano azul (físico), blanco (hielo del océano) y verde (biogeoquímico). El catálogo de datos incluye productos a escala global y regional, siendo estos una información oficial, gratuita y regulada [CMEMS, 2025a].

En cuanto al producto empleado, su nombre oficial es "*European North West Shelf/ Iberia Biscay Irish Seas - High Resolution L4 Sea Surface Temperature Reprocessed*" [CMEMS, 2023]. Este producto ofrece datos de la zona de la plataforma noroccidental europea, así como de la península Ibérica, Vizcaya y el Mar de Irlanda. El producto utilizado corresponde a un nivel de procesamiento L4.

Copernicus clasifica los datos según la fase de procesamiento en la que se encuentren. Estos niveles van desde L0 (los datos en bruto captados por el sensor) hasta L4. Los únicos disponibles para los usuarios finales son L3 y L4. Uno de los problemas discutidos en la selección de datos es el número reducido de variables disponibles en el nivel L4, como se verá con más detalle. Un conjunto de datos L3 puede contener una mayor cantidad de variables que uno de nivel L4. Sin embargo, al ser su nivel de procesamiento menor, presenta datos faltantes en ciertas casillas tras combinar la información de varios satélites. El nivel L4 corrige este aspecto aplicando un procesamiento y validación de datos adicionales, a costa de ofrecer datos menos actuales y con menos variables [CMEMS, 2024c].

Esta fuente provee una API mediante la librería de Python `copernicusmarine`, que permite acceder al catálogo de datos.

4.1.2. Copernicus Climate Data Store

Forma parte de la familia del programa Copernicus, al igual que CMEMS. Sin embargo, este producto se encuentra implementado por el ECMWF y se enmarca dentro del proyecto Servicio de Cambio Climático de Copernicus (C3S).

C3S es uno de los seis servicios de información temáticos del programa Copernicus y dentro de este tiene la función de ofrecer un portal de datos climáticos oficial, consistente y gratuito para todo tipo de usuarios. Su misión principal es dar soporte a las políticas de adaptación y mitigación frente al cambio climático de la Unión Europea [C3S, 2025a].

C3S proporciona sus datos a través de la herramienta Climate Data Store. En su sección de conjunto de datos climáticos ofrecen observaciones (datos obtenidos desde sensores), reanálisis climáticos, predicciones estacionales y proyecciones climáticas [C3S, 2025b].

Los datos utilizados como condiciones externas del modelo provienen de reanálisis climáticos. Estas técnicas combinan datos pasados con modelos numéricos, con el objetivo de producir series temporales coherentes para un conjunto de variables climáticas mayor aplicando las leyes de la física [C3S, 2025c]. Este proceso, conocido como asimilación de datos, se utiliza en los centros de predicción numérica climática.

En estos centros a intervalos regulares se integra una predicción previa con nuevas observaciones, creando un conjunto de análisis donde se mejoran las predicciones originales. El reanálisis aplica este mismo principio, pero a una resolución reducida para ofrecer un histórico de datos mayor. El producto específico utilizado es "*ERA5 hourly data on single levels from 1940 to present*", que proporciona estimaciones cada hora sobre diversas variables atmosféricas, oceánicas y terrestres. Se actualiza diariamente con una latencia

de cinco días. Los datos pueden ser corregidos en caso de inconsistencias tras dos o tres meses, cuando se lanzan las versiones finales [Hersbach et al., 2025].

4.1.3. National Oceanic and Atmospheric Administration

La Administración Nacional Oceánica y Atmosférica (NOAA) [1] es una agencia del Departamento de Comercio de los Estados Unidos dedicada al estudio de los cambios en el clima, incluyendo la oceanografía, alertas ante fenómenos meteorológicos adversos y la predicción climática. Al igual que Copernicus, proporciona a la sociedad datos y servicios que pueden ser empleados tanto en la investigación como en la protección del medio ambiente.

Esta fuente de datos se emplea para descargar la batimetría que forma parte del Modelo de Relieve Global ETOPO [NOAA, 2022a]. Este producto incluye información también acerca de topografía, líneas de costa, superficies de hielo y roca base. La versión más reciente corresponde al año 2022.

Para descargar los datos se emplea la API ArcGIS, donde están disponibles gran parte de los productos desarrollados por NOAA. En este caso, los datos se presentan en forma de imagen ráster, a la cual se accede mediante una solicitud al endpoint (API) del servidor que las almacena [2].

4.2. Productos descargados y selección de variables

En esta sección se abordarán en mayor profundidad las características de cada producto de datos descargado, así como su proceso de descarga, los elementos generados para su procesamiento y otros aspectos que contribuyen al modelo.

4.2.1. Zona de estudio

La zona de estudio seleccionada se corresponde a la costa noroeste de África e incluye casi toda la costa atlántica de Marruecos, el norte de Mauritania, así como las Islas Canarias y el archipiélago de Madeira, tal como la Figura 4.1 muestra.

Esta región ha protagonizado un gran número de estudios debido a su importancia en el fenómeno de afloramiento costero (llamado también *upwelling*). El afloramiento costero consiste en el desplazamiento de masas acuáticas debido al empuje del viento sobre la superficie, lo que provoca que aguas más profundas suban a la superficie. Las aguas ascendentes se caracterizan por su temperatura inferior y su riqueza en nutrientes, teniendo una alta productividad biológica [NOAA, 2025b].

El ecosistema de afloramiento de la Corriente de Canarias es uno de los cuatro principales sistemas de afloramiento en los márgenes orientales de costas a nivel mundial, que en las últimas décadas se ha visto afectado por el incremento de temperaturas asociado al cambio climático. El sistema se puede dividir en cinco subregiones, donde la zona de estudio abarca las subregiones de Marruecos, las Islas Canarias y la de Senegal-Mauritania. La subregión marroquí, cubre desde el cabo Sim (al norte de Canarias) hasta el cabo Blanco (norte de Mauritania). Esta subregión presenta afloramiento costero durante todo el año, aunque existen variaciones locales provocadas por los cabos, islas y la anchura de

[1] NOAA. What is upwelling? 2025

[2] NOAA. Event-Driven Map Services REST API Documentation. 2021.

la plataforma continental en forma de remolinos y corrientes menores. Por otro lado, la subregión de Mauritania, desde el cabo Blanco, no es cubierta en detalle en este trabajo, pero se caracteriza por la influencia de aguas del Atlántico Sur, así como perturbaciones ocasionadas por el archipiélago de Cabo Verde [Arístegui et al., 2009].

Las Islas Canarias juegan un papel fundamental dentro de este sistema. Actúan como una zona de transición entre las aguas costeras afectadas por el afloramiento y el océano abierto, caracterizado por aguas más cálidas. Introducen una segunda variación al perturbar el flujo mediante corrientes y los vientos alisios. Todo ello genera variabilidad regional, donde se destaca un filamento de afloramiento provocado por un remolino ciclónico desde la costa africana que es crucial para el transporte de peces y el intercambio de nutrientes [Barton et al., 1998]. La zona de mayor afloramiento se corresponde con la costa africana más próxima a las islas, entre el cabo Ghir y el cabo Juby, donde se produce este filamento y la actividad de afloramiento aumenta en verano [Arístegui et al., 2009].

La Tabla 4.1 muestra las coordenadas de latitud y longitud que delimitan el cuadrado de la zona de estudio.

Parámetro	**Valor**
Latitud máxima	34.525º
Latitud mínima	19.55º
Longitud máxima	-5.975º
Longitud mínima	-20.97º

Tabla 4.1: Coordenadas que delimitan el área de estudio.

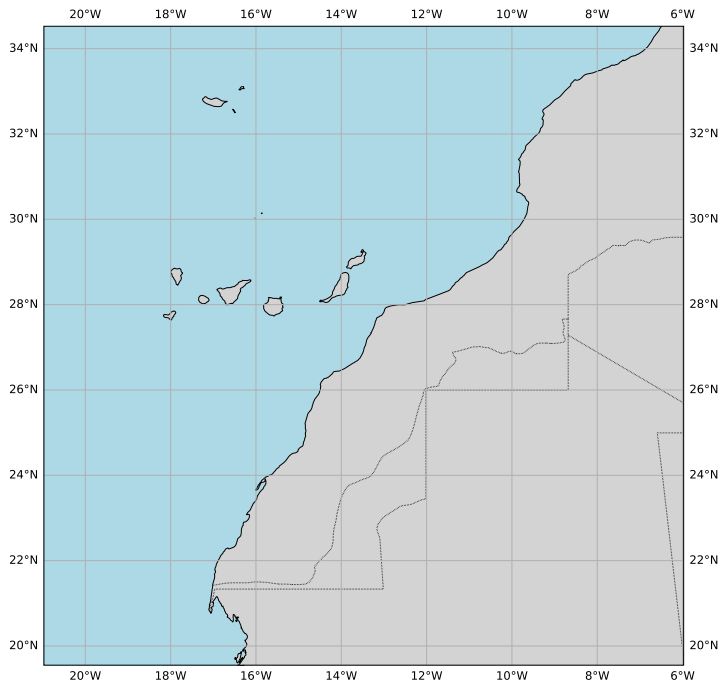


Figura 4.1: Zona geográfica cubierta por el estudio.

4.2.2. Decisiones de diseño

Como se ha mencionado anteriormente, este trabajo parte del modelo SeaCast, donde se trabaja con datos del mar Mediterráneo. A diferencia de la región de estudio de este proyecto, Copernicus ha desarrollado para el Mediterráneo productos con un mayor nivel de detalle. En el estudio original se emplean tanto datos de reanálisis como de análisis para entrenar el modelo.

Los datos de análisis representan una imagen del estado del océano en un instante concreto, calculada con un modelo en desarrollo. Estos datos se ven como un punto de inicio de la predicción. Mientras tanto, los datos de reanálisis son modelos más consolidados, diseñados para predicciones a largo plazo (varios años o décadas), y se entrenan con las observaciones más precisas disponibles, tanto de instrumentos físicos como de satélites [CMEMS, 2025b]. En la zona de estudio no se dispone de datos de análisis, por lo que se adaptó el código para no usarlos. Además, su uso no es estrictamente necesario, dado que en el trabajo original los datos de reanálisis cubren de 1987 a 2021, se emplean junto con los datos de análisis, que abarcan hasta 2024. Por lo tanto, el período de reanálisis es lo suficientemente amplio para entrenar el modelo.

Por otro lado, existen ciertas constantes que tampoco se encuentran en los datos y reconstruirlas desde otros productos quedaba fuera del alcance del trabajo. El producto con identificador `cmems_mod_med_phy_my_4.2km_static`, incluye tanto datos de batimetría (bajo la variable `sea_floor_depth_below_geoid`) como de la topografía dinámica media (bajo la variable `sea_surface_height_above_geoid`) [CMEMS, 2024b]. Ambas constantes se calculan respecto al geoide [3], que es un modelo que corresponde con el nivel medio global del mar, utilizado para medir tanto las profundidades oceánicas como las elevaciones de la superficie terrestre.

Se realizaron pruebas para generar la topografía dinámica media utilizando datos a nivel global (Global Ocean Physics Reanalysis [CMEMS, 2022]) y europeo (European Seas Mean Dynamic Topography [CMEMS, 2024a]). Sin embargo, la resolución de estos productos es inferior a la del original, obteniendo datos con menor nivel de detalle. La necesidad de aplicar una operación de submuestreo (*down-sampling*) sobre los datos oceanográficos o de sobremuestreo (*up-sampling*) sobre los datos de Topografía Dinámica Media llevó a cuestionar el uso de esta constante. Además, se detectaron discrepancias en los valores globales y europeos, como muestra la Figura 4.2. Los datos representados han sido sobremuestreados para igualar la resolución del conjunto de datos empleado de CMEMS. Finalmente, se descartó el uso de esta constante.

Otra de las características del modelo base es el uso de forzamiento en las fronteras del mar Mediterráneo. En particular, se aplica una máscara binaria para identificar el estrecho de Gibraltar, permitiendo representar y asegurar la compatibilidad del modelo con el flujo de entrada y salida con el océano. Los autores identifican esta zona como una parte crucial del modelo. Por ello, en lugar de alimentar al modelo con sus propias predicciones, la zona fronteriza utiliza en todo momento observaciones reales (durante el entrenamiento) o predicciones de modelos numéricos de Copernicus (durante la inferencia). Como estas últimas predicciones están disponibles en un horizonte de diez días, y el período empleado en el estudio es de quince, se extienden las predicciones repitiendo hasta en cinco ocasiones el último estado [Holmberg et al., 2024].

En la fase inicial de Modelado, se realizaron prototipos de máscaras binarias para aplicar este tipo de forzamiento en el proyecto. Sin embargo, se desestimó su inclusión

[3] NOAA. What is the geoid? 2021.

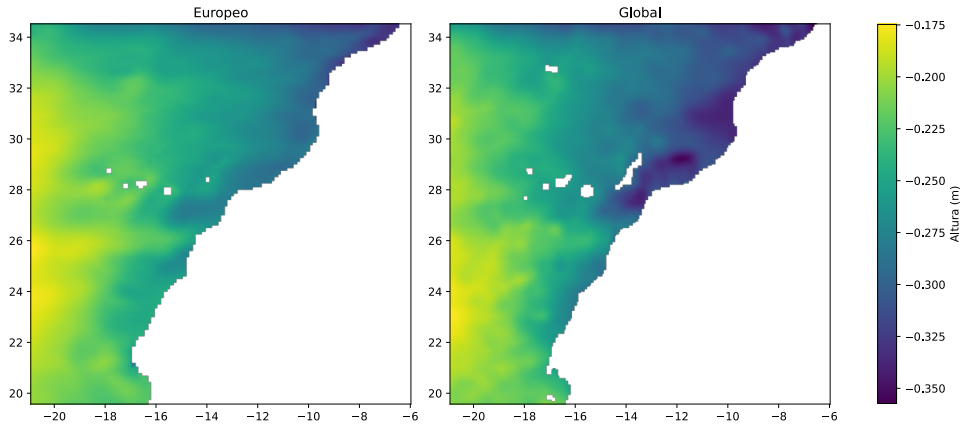


Figura 4.2: Comparación de Topografía Dinámica Media obtenida entre datos europeos y globales.

por la complejidad adicional que implicaba. Esta decisión fue apoyada por la diferencia en el área de estudio respecto a SeaCast, ya que en el océano abierto las diferencias no serían tan significativas, además de que evaluar distintas alternativas de máscaras requeriría un tiempo considerable. Todo ello motivó considerarlo como un futuro tema a explorar en otras investigaciones.

Por otro lado, en el trabajo original se incluyen variables no disponibles en el conjunto de datos oceanográfico empleado en este proyecto, algunas de ellas representadas en múltiples niveles. Además, estos datos tienen una dimensión menos respecto a los originales, ya que trabajamos con un único nivel y la resolución temporal también es diferente. En cuanto a los forzantes atmosféricos, el modelo original utiliza cuatro variables distintas, mientras en este estudio se emplean dos. Pese a ello, el conjunto de datos utilizado para las condiciones externas durante el entrenamiento es el mismo.

SeaCast emplea además forzantes provenientes de otras dos fuentes durante la fase de evaluación. Se usan predicciones del modelo de control de los modelos numéricos de ECMWF basados en conjuntos, así como predicciones de AIFS, que es el modelo de inteligencia artificial desarrollado por ECMWF para predicción [Holmberg et al., 2024]. El modelo de control del Sistema de Predicción de Conjuntos de ECMWF, es una simulación que comienza con las mejores condiciones iniciales sin perturbaciones, mientras que el resto de los cincuenta miembros tiene condiciones iniciales ligeramente modificadas [ECMWF, 2012]. El propósito de este enfoque es simular durante la fase de inferencia la indisponibilidad de observaciones precisas, usando predicciones de modelos que no van a ser tan exactas. Debido al enfoque de este proyecto, se optó por no recurrir a estas fuentes de datos, ya que complicaba el uso durante la fase de inferencia.

4.2.3. Datos oceanográficos

El conjunto de datos oceanográficos mencionado previamente, corresponde al producto denominado “Plataforma Continental Noroeste Europea / Mar de Irlanda, Golfo de Vizcaya e Iberia - Temperatura Superficial del Mar en Alta Resolución L4 Reprocesada”, que se obtiene mediante el servicio de datos de CMEMS. Se trata de un producto de reanálisis de nivel 4, lo que significa que ha sido procesado para corregir valores faltantes de observaciones satelitales, por ello, su actualización es anual.

Los datos tienen una resolución temporal diaria y una resolución espacial de 0.05º

x 0.05º (que representa la distancia angular mínima entre dos puntos contiguos de la rejilla). La serie de datos cubre desde el 1 de enero de 1982 hasta el 31 de diciembre de 2023. Estos datos fueron recopilados y procesados por parte del Instituto Francés de Investigación para la Explotación del Mar (IFREMER), que unificó dos fuentes de datos satelitales de productos de nivel 3.

En cuanto al área geográfica cubierta, abarca desde 8.9º hasta 62º en latitud norte, y entre -21º y 13º de longitud este, incluyendo la parte occidental del mar Mediterráneo, el mar de Irlanda, el mar Cantábrico y el océano Atlántico nororiental desde el sur de Islandia hasta la costa de Mauritania, excluyendo las Islas Azores. Los datos, presentados en forma de rejilla, representan la temperatura superficial del mar (variable `analysed_sst`) en grados Kelvin (K), medida a 20 centímetros de profundidad. Dentro del conjunto de datos existe otra variable, que representa la desviación estándar estimada del error de la temperatura superficial del mar analizada (variable `analysis_error`), también expresada en Kelvin. Esta variable se descartó para el entrenamiento, puesto que significaría usar datos correlacionados con la temperatura [CMEMS, 2023].

La descarga se realiza a través de la API que CMEMS ofrece para Python, mediante el paquete `copernicusmarine`. Para acceder, es necesario estar registrado en el servicio. En el código base ya existía un archivo llamado `download_data.py` que gestiona la lógica de descarga de datos, por lo que se reutilizó esta estructura. Para ello fue necesario modificar el argumento de dataset de modo que, en caso de introducir "atlantic", se use el identificador del conjunto de datos específico, junto con su versión y variable correspondiente, y manteniendo también la posibilidad de descargar datos del Mediterráneo. El resultado de la consulta a la API es un conjunto de datos en formato *Network Common Data Form* (NetCDF-4), un formato de archivo empleado para el almacenamiento de datos multidimensionales. Este formato es ampliamente empleado en el campo de los Sistemas de Información Geográfica (SIG) y se ha adoptado como estándar para ciertos tipos de datos científicos [Esri, 2025b].

La función encargada de descargar los datos recibe como argumento el rango de fechas deseado. Se itera sobre este rango mes a mes para obtener el conjunto de datos correspondiente de CMEMS. Como se ha mencionado, estos datos tienen un formato multidimensional, por lo que se emplea `xarray` para su procesamiento. El objetivo es recorrer el conjunto de datos específico de un mes y almacenar los datos de forma diaria en un archivo `numpy`. La Figura 4.3 muestra los registros de SST para el día 1 de enero de 2018. Como se puede observar, los datos se representan en una cuadrícula de 300 x 300, donde el color blanco indica la superficie terrestre (valores nulos). Sin embargo, el archivo `.npy` almacenado contiene los datos en forma de una matriz con dimensiones (49061, 1), donde cada fila corresponde a una coordenada oceánica y la columna representa la variable almacenada (SST). Esto se debe a que en él solo se guardan los registros correspondientes a puntos en el océano, evitando así almacenar datos nulos correspondientes a zonas terrestres.

Para ello se utiliza una máscara binaria tierra-océano, compuesta por valores booleanos (verdadero para océano, falso para tierra), que juega un papel fundamental en este proceso, ya que permite tanto el filtrado previo al almacenamiento como la reconstrucción espacial de los datos para su representación gráfica.

En este caso, no existe un problema de incompatibilidad dimensional. Aunque los datos originales de SeaCast en el Mediterráneo tienen variables con múltiples niveles verticales (con un total de 75 variables, considerando cada nivel como una variable individual), no existe una dimensión adicional para representar el nivel. En su lugar, los niveles de cada

variable están representados como columnas de la matriz, por lo que su segunda dimensión sería de tamaño 75. Esto permite que la forma de los datos se conserve, sin necesidad de realizar cambios adicionales sobre la fase de descarga.

Sin embargo, para adaptar los datos a las entradas que espera SeaCast, es necesario modificar algunas constantes que tienen el tamaño de las dimensiones explícitamente definidas.

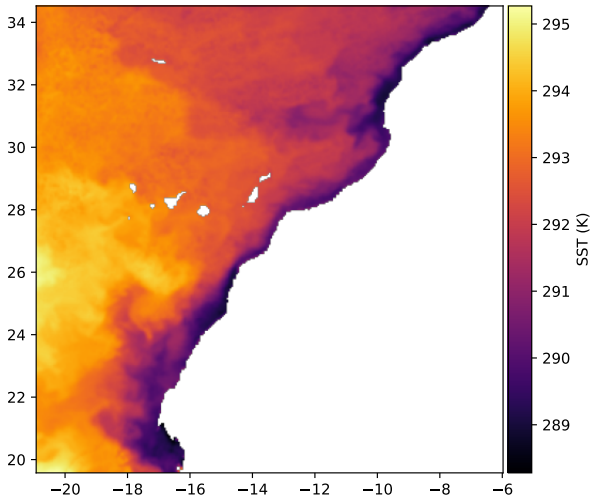


Figura 4.3: Datos de SST en grados Kelvin para el día 1 de enero de 2018.

4.2.4. Datos de forzantes atmosféricos

El producto seleccionado se denomina "datos horarios de ERA5 en niveles simples desde 1940 hasta la actualidad" (*ERA5 hourly data on single levels from 1940 to present*) [Hersbach et al., 2025]. Este es el mismo que se utilizó en el proyecto original, puesto que también cubre la zona de estudio (los datos son de enfoque global). De forma similar a los datos oceanográficos, el código de SeaCast incluye una función para descargar los datos de ERA5 de reanálisis mediante la API de Climate Data Store, con el paquete `cdsapi`. ERA5 es un conjunto de datos bastante extenso que incorpora una gran cantidad de variables atmosféricas, terrestres y de oleaje a escala horaria. No solo están disponibles datos de reanálisis, sino que también existen versiones de conjuntos para representar la incertidumbre, ofreciendo datos de los miembros y métricas calculadas como la media o la dispersión.

Los datos presentan una resolución espacial de 0.25^{o} x 0.25^{o} y abarcan desde el año 1940 hasta la actualidad, con cinco días de latencia. El formato predeterminado de los datos es *General Regularly distributed Information in Binary form* (GRIB), que fue diseñado específicamente para la distribución de datos climáticos [ECMWF, 2025b]. Sin embargo, Copernicus proporciona la opción de descargar los datos en formato NetCDF-4. También es posible obtenerlos en formato comprimido en lugar de en un archivo con extensión `.nc`, aunque esta opción genera problemas con el código original al no crear un único archivo para su procesamiento, por lo que se optó por usar el formato NetCDF estándar.

Las variables seleccionadas representan los componentes del viento a 10 metros sobre la superficie, uno en sentido este-oeste (componente u, variable `u10`) y otro en la dirección norte-sur (componente v, variable `v10`), ambos medidos en metros por segundo [Hersbach et al., 2025].

Respecto al código de descarga, la función recibe un intervalo de fechas como argumento, así como las coordenadas de la zona de estudio extraídas de una máscara binaria en formato `.nc`. En este caso, se guardan los datos inicialmente en formato NetCDF-4 (extensión .nc) antes de ser procesados. La descarga se realiza agrupando bloques de seis meses, recuperando los datos para todos los días de ese período en cuatro instantes horarios diarios (00:00, 06:00, 12:00, 18:00). Tras guardar los datos, el archivo NetCDF-4 se abre para calcular una media diaria, además de para interpolar los datos de manera que compartan resolución espacial con los datos de CMEMS mediante una operación de sobremuestreo. Finalmente, se guarda cada registro del archivo multidimensional procesado en un archivo `numpy` individual para cada día. Cada uno de estos archivos contiene dos columnas y un total de 49601 filas, que representan únicamente los valores sobre el océano, con una forma de (49601, 2). La Figura 4.4 muestra un ejemplo de los datos comparando los componentes u y v para el día 1 de enero de 2018.

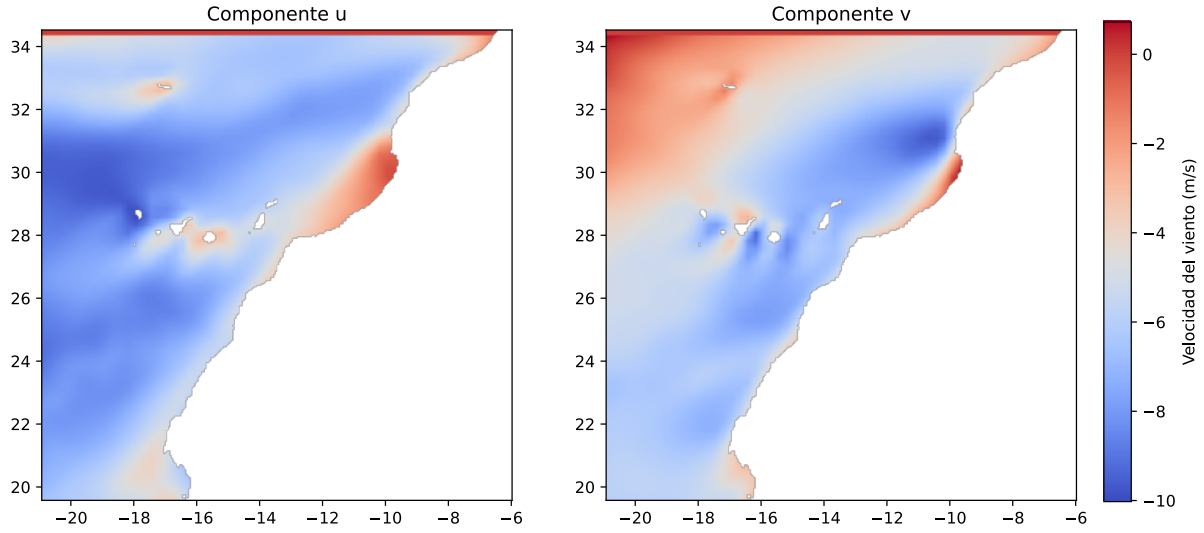


Figura 4.4: Componentes u y v del viento a 10 metros de la superficie para el día 1 de enero de 2018.

4.2.5. Datos sobre la batimetría

En la sección 4.2.2 se identificó como una de las dificultades de la adaptación del proyecto al Atlántico Norte la falta de datos batimétricos en el conjunto de datos oceanográficos. Para evitar esta limitación, se recurrió al Modelo de Relieve Global ETOPO [NOAA, 2022a], del cual se extrajo la profundidad oceánica en cada punto.

Aunque en esta etapa se volvió a usar una API para la descarga de datos, no existe una biblioteca específica de Python para este caso. En su lugar, se realiza una consulta directa a través de una *Uniform Resource Locator* (URL) particular, en la cual se especifican los parámetros y el servicio requerido. La API utilizada es la de ArcGIS, una plataforma con recursos software enfocados en los Sistemas de Información Geográfica desarrollada por Esri [Esri, 2025a]. Los servicios y recursos disponibles se organizan en una jerarquía que puede ser explorada mediante la interfaz web de ArcGIS [4].

[4]NOAA. Event-Driven Web Services REST API Documentation. 2021.

Se utiliza el servicio de imágenes para obtener los datos en forma de ráster (datos geográficos en forma de una cuadrícula de píxeles) que son procesados dinámicamente. Estos conjuntos de imágenes se conocen como conjuntos de datos mosaico y es posible acceder a ellos delimitando la zona de interés mediante teselas. Sin embargo, los datos descargados corresponden a imágenes completas [5]. Al realizar la solicitud, se especifica que el recurso a generar es un Modelo de Elevación Digital (DEM), que representa la superficie terrestre y el fondo marino de la Tierra [NOAA, 2023].

Tras indicar en la URL el servidor de imágenes y el recurso que se desea generar, es necesario señalar la operación a realizar. En este caso, sería la de exportar imagen, la cual permite incluir argumentos para ajustar los datos que incluyen la interpolación, el formato del archivo, el tipo de datos del píxel o la fuente. Es importante señalar que, al tratarse de una solicitud para obtener una imagen, el servidor responde transmitiendo directamente los bytes de la imagen generada, por lo que no se cuenta con metadatos acerca de ella [6].

Entre los argumentos más relevantes se encuentra la regla del mosaico, mediante la cual se indica el modelo específico de ETOPO que actúa como fuente. En este caso, se usa la versión ETOPO 2022 de 0.0083º de resolución, en su variante de roca firme (*bedrock*). El conjunto de datos base sobre el que se construye ETOPO para obtener la batimetría oceánica global es GEBCO [NOAA, 2022b].

Una vez obtenidos los datos, se ajustaron las dimensiones mediante un proceso de submuestreo y se invirtieron las alturas para dar más importancia a los valores oceanográficos frente a los terrestres. Los valores correspondientes a la superficie terrestre se establecen a 0 para evitar que afecten al modelo, en caso de que no sean correctamente filtrados por la máscara tierra-agua. La Figura 4.5 muestra el resultado final.

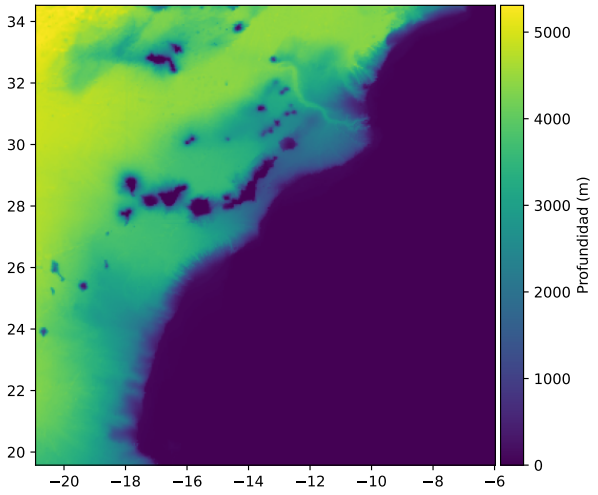


Figura 4.5: Batimetría procesada descargada desde ETOPO.

4.2.6. Datos auxiliares

A continuación, se presentan datos derivados de los productos descargados o generados desde cero, que se utilizan para el preprocesamiento de los datos descargados y para complementar el modelo con características adicionales.

[5] NOAA. ArcGIS REST API Documentation – Image Service. 2021.
[6] NOAA. ArcGIS REST API Documentation – Export Image. 2021.

Máscara tierra-océano

Existen ciertos datos que no se descargan desde fuentes externas, sino que se construyen o se calculan implícitamente durante el entrenamiento. Uno de los elementos mencionados anteriormente es la máscara tierra-océano, que se genera a partir de un archivo multidimensional NetCDF-4. Este archivo (`bathy_mask.nc`) contiene las coordenadas de latitud y longitud, junto con una variable binaria `mask`, que toma el valor 1 para el océano y 0 para la tierra.

Este archivo multidimensional actúa como base para la creación de otros archivos derivados. De él se extrae la máscara mencionada (`sea_mask.npy`), con forma (1, 300, 300) y valores booleanos según tierra y océano. Esta máscara es la usada durante el preprocesamiento de los datos descargados para asegurar que no se almacenen valores nulos. También se extrae un archivo de coordenadas (`coordinates.npy`), de forma (2, 300, 300) donde cada matriz componente contiene los valores de latitud y longitud correspondientes a cada punto.

Cabe destacar que la máscara tierra-océano asigna verdadero a los valores no nulos de temperatura y falsos a los nulos. La Figura 4.6 muestra la visualización de esta máscara.

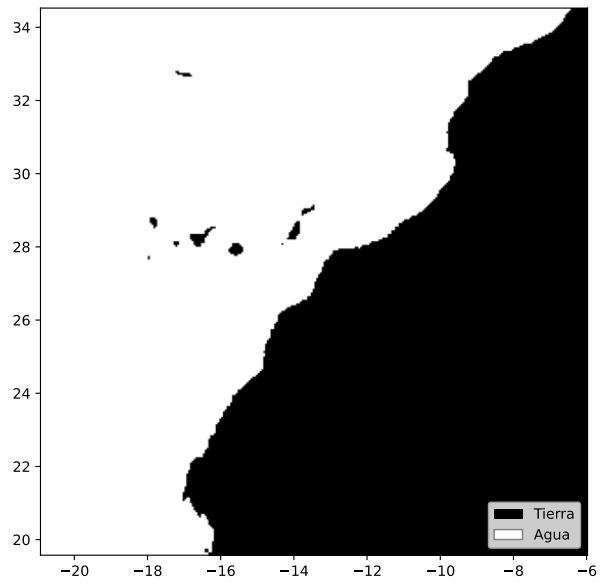


Figura 4.6: Máscara binaria para diferenciar entre océano y tierra.

Por último, a partir de la máscara binaria tierra-océano se genera un archivo (`nwp_xy.npy`) que contiene los índices espaciales correspondientes a las coordenadas, con una estructura similar a la del archivo de coordenadas.

Pesos latitud-longitud

Debido a la forma esférica del planeta, la rejilla regular de latitud-longitud no representa áreas iguales en superficie. Las celdas situadas cerca de los polos son más pequeñas que las próximas al ecuador. Si se asignara el mismo peso a todas las celdas, se produciría una pérdida de información a favor de las celdas más pequeñas [Rasp et al., 2024].

Para solventarlo, se emplea la ecuación (4.1), usada también en proyectos como WeatherBench para ponderar cada punto en función de su latitud.

$$w_i = \frac{\cos(\theta_i)}{\frac{1}{N}\sum_{j=1}^{N}\cos(\theta_j)}, \tag{4.1}$$

donde θ_i es la latitud del punto i, expresada en radianes, y N es el número total de puntos.

Esta fórmula está normalizada de manera que la suma de todos los pesos sea igual al número de celdas, dividiendo por la media de todos los cosenos. Así, se garantiza que el promedio de los pesos w_i sea uno. Se utiliza el coseno de la latitud porque este se aproxima a uno en el ecuador (latitud 0^{o}), mientras que en los polos tenderá a cero [Rasp et al., 2020], reflejando la importancia de cada celda. En la Figura 4.7 se muestra el mapa con los pesos correspondientes, que se usan durante el cálculo de la función de pérdida en el entrenamiento.

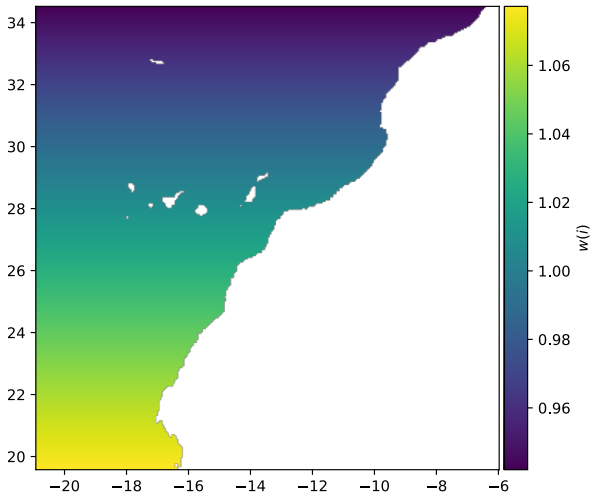


Figura 4.7: Mapa de pesos según la latitud.

Forzantes temporales

El seno y el coseno del día del año son utilizados como condicionantes externos, con el objetivo de capturar variaciones estacionales. Los valores se encuentran normalizados entre cero y uno, y la lógica para su cálculo ya estaba implementada incorporada en el modelo base de SeaCast[Holmberg et al., 2024].

4.2.7. Datos de entrada del modelo

La Tabla 4.2 presenta un resumen de los datos utilizados durante el entrenamiento por parte del modelo. En el próximo capítulo se describe en detalle cómo se procesan estos datos en cada etapa del entrenamiento. Podría considerarse que el modelo usa tres tipos de datos: variables, estáticos y forzantes.

Los datos relacionados con el océano incluyen la variable SST y los campos estáticos de latitud, longitud y la batimetría. Por otro lado, los forzantes están relacionados con condiciones externas que pueden ayudar a mejorar la precisión. En este caso, son los componentes vectoriales del viento a 10 metros sobre la superficie, así como el seno y el coseno del día del año. La variable a predecir es la SST, estimada a lo largo de un horizonte temporal determinado.

Categoría	Nombre	Descripción
Variables	sst	Temperatura Superficial del Mar (K)
Campos estáticos	sea_depth	Batimetría (profundidad en metros)
	lat	Latitud (º)
	lon	Longitud (º)
Forzantes	u10	Componente este-oeste de la velocidad del viento a 10 metros sobre la superficie (m/s)
	v10	Componente norte-sur de la velocidad del viento a 10 metros sobre la superficie (m/s)
	seno del día del año	Transformación cíclica del día del año (seno) para representar estacionalidad
	coseno del día del año	Transformación cíclica del día del año (coseno) para representar estacionalidad

Tabla 4.2: Resumen de todas las variables, campos estáticos y características de forzamiento empleados.

4.3. División de los datos

En el capítulo de Planificación 3, uno de los problemas identificados fue la dificultad para aplicar validación cruzada. Pese a ello, una de las estrategias para evitar el sobreajuste en el modelo consiste en dividir los datos en distintos conjuntos, de modo que unos se empleen en la fase de entrenamiento, otros en la de validación y otros en la de evaluación. De esta forma, se puede simular el comportamiento del modelo enfrentándose a datos que nunca ha visto anteriormente.

Aunque inicialmente se había planteado trabajar con cinco años de datos, finalmente se optó por trabajar con un total de 21 años. Esta decisión se tomó debido a la necesidad de contar con un modelo sólido sobre el cual realizar los experimentos en fase de inferencia. Como se comentó anteriormente, entrenar múltiples modelos individuales sería muy costoso computacionalmente, por lo que se procuró que el modelo final fuera lo más robusto y preciso posible, ya que serviría como base para crear conjuntos de predicciones.

En cuanto a la división de los datos, los datos de entrenamiento cubren desde 2003 hasta 2019 (17 años), los de validación corresponden a 2020 y 2021 (2 años), y los de prueba abarcan 2022 y 2023 (2 años). Aproximadamente el 81 % de los datos se emplean para entrenar el modelo, mientras que un 9.5 % se reserva para la validación y otro 9.5 % para la evaluación. La Figura 4.8 muestra de forma gráfica esta división de datos.

Esta estrategia es coherente con la usada en proyectos de predicción climatológica u oceanográfica, donde se utilizan una gran cantidad de datos históricos para crear un modelo que sea lo más preciso posible.

El código base de SeaCast contiene una lógica que permite la organización de los conjuntos de datos anteriores por rangos de fechas, sin embargo, emplea un parámetro llamado estados. No obstante, esto se explicará con más detalle en el próximo capítulo, dedicado a la predicción oceanográfica con redes neuronales de grafo.

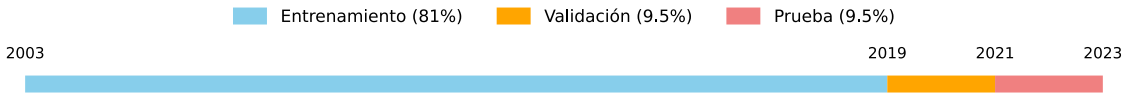


Figura 4.8: Distribución temporal del conjunto de datos entre las fases de entrenamiento, validación y prueba.

Capítulo 5

Predicción de la dinámica oceanográfica con redes de grafos

En este capítulo se explica en mayor detalle SeaCast, una arquitectura basada en red neuronales diseñada para realizar predicciones de alta resolución a medio plazo. Su mecanismo se basa en el uso de GNNs, lo que le permite adaptarse a la compleja geometría irregular de los océanos, en combinación con un enfoque autorregresivo. El modelo original se centra en la predicción de variables oceanográficas en el mar Mediterráneo, utilizando tanto forzantes externos como datos de reanálisis y análisis. En este trabajo, dicha configuración se ha adaptado al océano Atlántico.

En primer lugar, se detalla el método empleado por SeaCast para realizar las predicciones, incluyendo su estrategia autorregresiva y función de pérdida. A continuación, se profundiza en el diseño del grafo utilizado por la arquitectura GNN, revisando los componentes principales de esta arquitectura: codificador, procesador y decodificador. Tras ello, se describe la ejecución de un conjunto de predicciones de SeaCast partiendo de perturbaciones iniciales, desde la descarga de datos hasta la unificación de la salida de un conjunto. Finalmente, se destacan los cambios realizados para poder adaptar el modelo a nivel de código, así como para la integración de las métricas de WeatherBench-X.

5.1. Metodología de SeaCast

El primer paso es definir de manera general el problema de predicción para entender cómo opera el modelo. SeaCast trabaja en ventanas temporales utilizando el concepto de estado, donde cada estado representa las variables oceanográficas para un día concreto. La ventana histórica de datos puede expresarse como $X^{-h:0} = (X^{-h}, \ldots, X^0)$, donde X^{-h} corresponde al estado en el día h del pasado, mientras que X^0 corresponde al estado actual. Por otro lado, las predicciones se expresan como $X^{1:T} = (X^1, \ldots, X^T)$, donde T representa la longitud, en días, del horizonte de la predicción. De forma similar, se pueden identificar los forzantes como $F^{1:T} = (F^1, \ldots, F^T)$ [Holmberg et al., 2024].

La predicción se puede formular como:

$$\hat{X}^t = f\left(X^{t-2:t-1}, F^t\right). \tag{5.1}$$

La primera entrada para la predicción en la ecuación son los estados iniciales, compuestos por las dos observaciones más recientes al instante t a predecir. Por ejemplo, para estimar el estado futuro $\hat{X}^1$, los estados iniciales serían la ventana $X^{-1:0}$.

La otra entrada a la red será F^t, que representa las condiciones externas. Con esta configuración se obtiene una primera predicción. No obstante, gracias a la naturaleza autorregresiva del modelo, es posible extender la predicción hasta una longitud T cualquiera. En ese caso, empezando desde el estado actual X^0 se generarían los estados $\hat{X}^{1:T}$. En la Figura 5.1, se realiza un esquema general de la autorregresión de SeaCast [Holmberg et al., 2024].

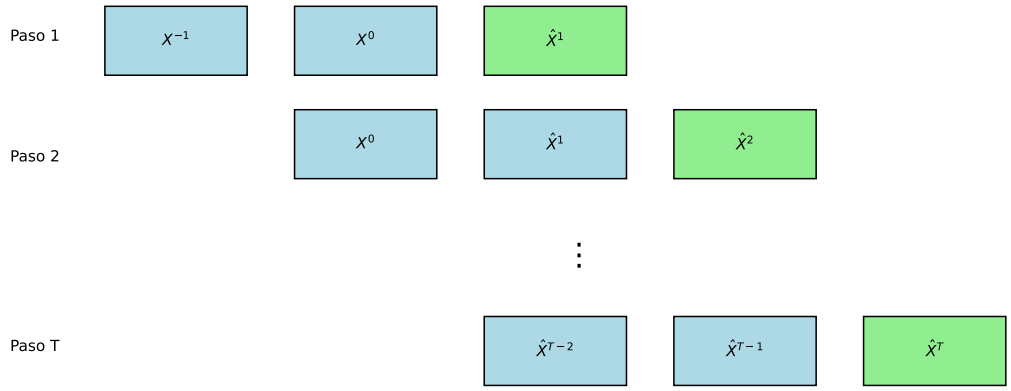


Figura 5.1: Esquema del funcionamiento autorregresivo de SeaCast.

En la subsección 4.3 se mencionó que, al dividir los datos en los conjuntos para entrenamiento, validación y prueba, se especificaba el número de estados como argumento. Aunque se valoró la posibilidad de modificar este parámetro, finalmente se mantuvo el tamaño de estados para cada conjunto usado originalmente en SeaCast con un pequeño matiz. El estudio de SeaCast determina seis estados para entrenamiento, otros seis para la evaluación y 17 para la fase de inferencia. Teniendo en cuenta que cada estado tiene la forma $X_t \in \mathbb{R}^{N \times d_x}$, donde N son los puntos del océano y d_x son la cantidad de campos por estado, obtenemos las formas de cada uno de los componentes de los conjuntos de datos.

La Tabla 5.1 muestra que los conjuntos de entrenamiento y evaluación comparten las mismas dimensiones. Como se explicó, la predicción necesita cuatro componentes: los estados iniciales, los estados objetivos, los forzantes y los datos estáticos. Los estados iniciales corresponden a dos días siempre, incluyendo el estado actual y el anterior. Los estados objetivos representan las variables oceanográficas de los T días de predicción, siendo en este caso un único día. De manera similar, los forzantes también tienen los datos para un único día, puesto que el modelo se entrena realizando predicciones para un paso temporal. En cuanto a los datos estáticos, al ser constantes, se utilizan los mismos valores en cada instante y siempre comprenden tres campos: latitud, longitud y batimetría.

Todos los elementos tienen datos en los 49,061 puntos oceánicos. La última dimensión del tensor representa las variables (o campos). Como se detalló en la sección 4.2.3, los datos oceanográficos solo incluyen SST, por lo que los estados iniciales y objetivos cuentan con una sola variable. Por otro lado, los datos forzantes tienen una dimensión de 12. Teniendo en cuenta que se emplean cuatro forzantes (componentes u y v del viento, así como el seno y coseno del día del año), estos se repiten tres veces. Esta repetición refleja el uso de una ventana temporal de tres días para cada predicción: los forzantes del día anterior,

los del actual y los del posterior.

Por otro lado, la Tabla 5.2 muestra que las dimensiones de los tensores se mantienen en la fase de evaluación, salvo por el número de días. En el conjunto de prueba se hacen predicciones en un horizonte temporal de 15 días. Los estados iniciales siguen consistiendo en dos días, pero los forzantes y los estados objetivos abarcan 15 días. Esto permite, por un lado, calcular el error de las predicciones, y en el caso de los forzantes, alimentar el modelo durante la fase autorregresiva, donde se combinan con las predicciones generadas para inferir los siguientes estados.

Tipo de datos	**Nº de Días**	**Nº de Observaciones**	**Nº de Campos**
Estados Iniciales	2	49,061	1
Estados Objetivos	1	49,061	1
Forzantes	1	49,061	12

Tabla 5.1: Descripción de los tipos de datos utilizados en el conjunto de entrenamiento y evaluación.

Tipo de datos	**Nº de Días**	**Nº de Observaciones**	**Nº de Campos**
Estados Iniciales	2	49,061	1
Estados Objetivos	15	49,061	1
Forzantes	15	49,061	12

Tabla 5.2: Descripción de los tipos de datos utilizados en el conjunto de prueba.

5.2. Pasos autorregresivos

SeaCast, en su configuración original, trabaja con secuencias de seis días durante la fase de entrenamiento y validación, donde dos días se usan como estados iniciales y cuatro como objetivos (por lo tanto, los forzantes abarcan cuatro días también). Esto se debe a que el modelo puede ser entrenado para mejorar sus predicciones a largo plazo mediante pasos autorregresivos.

En este proyecto, se usa un único paso autorregresivo durante el entrenamiento, puesto que solo se realiza una predicción. En el trabajo original de SeaCast se emplean hasta cuatro pasos autorregresivos, cuya evolución a lo largo del entrenamiento se determina mediante dos parámetros del modelo. Aunque en esta sección no se abordan en detalle los ajustes específicos utilizados, se presenta un ejemplo para entender cómo evolucionarían los pasos autorregresivos durante el entrenamiento cuando se adopta esta estrategia. La siguiente fórmula define en qué épocas del entrenamiento se incrementan los pasos autorregresivos:

$$e_i = \lfloor \alpha \cdot E \rfloor + i \cdot \left\lfloor \frac{E - \lfloor \alpha \cdot E \rfloor}{S - 1} \right\rfloor \quad \text{para } i = 0, 1, \ldots, S - 2, \tag{5.2}$$

donde E es el número total de épocas de entrenamiento, S es el número de pasos autorregresivos máximos y α es el porcentaje del entrenamiento donde se comenzarán a aumentar

los pasos autorregresivos. El resultado e_i, representa la época donde se cambia al número de pasos autorregresivos dado por $i+2$. El índice i está en varía hasta el valor $S-2$.

Como caso de ejemplo, se supone que se quiere realizar un entrenamiento de 30 épocas (E), con un máximo de tres pasos autorregresivos (S) y el comienzo del aumento de los pasos autorregresivos en el 60 % del entrenamiento (α). El primer término de la fórmula calcula la época donde se empezarían a incrementar los pasos autorregresivos.

$$e_{\text{comienzo}} = [\alpha \cdot E] = [0{,}6 \cdot 30] = 18. \tag{5.3}$$

Como se observa, la época donde se incrementarían por primera vez los pasos autorregresivos sería la 18. Por lo tanto, para el caso $i = 0$, correspondiente al primer intervalo de entrenamiento en el que se incrementa el número de pasos autorregresivos (de uno a dos), la expresión quedaría de la siguiente manera:

$$e_0 = \lfloor \alpha \cdot E \rfloor + 0 \cdot \left\lfloor \frac{E - \lfloor \alpha \cdot E \rfloor}{S-1} \right\rfloor, \tag{5.4}$$

donde sustituyendo los datos, la operación resulta en:

$$e_0 = \lfloor 0{,}6 \cdot 30 \rfloor + 0 \cdot \left\lfloor \frac{30 - \lfloor 0{,}6 \cdot 30 \rfloor}{3-1} \right\rfloor = 18 + 0 \cdot \left\lfloor \frac{12}{2} \right\rfloor = 18. \tag{5.5}$$

Es decir, se usaría un único paso autorregresivo hasta la época 18, realizando una única predicción a partir de los estados iniciales. Después, quedaría un único valor siguiente de i para determinar el próximo punto de cambio en los pasos autorregresivos, que sería:

$$e_1 = 18 + 1 \cdot \left\lfloor \frac{30 - 18}{2} \right\rfloor = 18 + 6 = 24. \tag{5.6}$$

Por tanto, se aplicarían dos pasos autorregresivos (es decir, se realizan dos predicciones consecutivas) desde la época 18 hasta la 24, y tres pasos autorregresivos (tres predicciones consecutivas) desde la época 24 hasta la 30, que marca el final del entrenamiento. Este último valor se calcula de forma implícita por el número máximo de épocas del entrenamiento.

5.3. Función de pérdida

El modelo se entrena con el objetivo de minimizar una función basada en el Error Cuadrático Medio (MSE). Esta función de pérdida (L) tiene en cuenta la secuencia de estados generados mediante pasos autorregresivos, también conocida como *rollout*. La función se representa con la siguiente ecuación:

$$L = \frac{1}{T_{\text{rollout}}} \sum_{t=1}^{T_{\text{rollout}}} \sum_{i=1}^{C} \sum_{l=1}^{L_i} \frac{1}{|G_l|} \sum_{v \in G_{l(i)}} a_v \lambda_i \left(\hat{X}_{v,i}^t - X_{v,i}^t \right)^2, \tag{5.7}$$

donde T_{rollout} es el número de estados desplegados o el número de pasos autorregresivos, C es es el total de variables, L_i es el número de niveles de profundidad de la característica i y G_l representa el número de puntos en el océano en el nivel l. El término a_v representa la ponderación según el tamaño de la celda, calculada usando el coseno de la latitud (4.1)

para cada celda v y nivel de profundidad l. Por último, λ_i es la inversa de la varianza de las estimaciones para la variable i [Holmberg et al., 2024].

La inclusión de estos dos últimos términos de ponderación en la función de pérdida fue propuesta originalmente por Keisler [Keisler, 2022], en uno de los trabajos en los que se basa SeaCast. Su objetivo es reescalar cada variable de forma que tenga varianza unitaria, antes de calcular el MSE. En dicho trabajo, esta normalización se realiza en cada variable antes de calcular el error, dividiéndolas por la desviación estándar de sus diferencias temporales a tres horas, promediada a lo largo del espacio y el tiempo utilizando 100 ventanas temporales.

En la función de SeaCast, esto se incorpora como una medida posterior para normalizar el cálculo del MSE. Tanto $\hat{X}^t_{v,i}$ como $X^t_{v,i}$ se encuentran normalizados y con varianza unitaria, ya que las entradas de la red se normalizan siguiendo la ecuación:

$$\tilde{X}^t_{v,i} = m_v \cdot \frac{X^t_{v,i} - \mu_i}{\sigma_i}, \tag{5.8}$$

donde μ_i representa la media de la muestra de los estados, mientras que σ_i representa la desviación estándar. m_v representa el valor en máscara tierra-océano para el punto v, lo que evita asignar valores a puntos que no pertenecen al océano. Para normalizar la pérdida, SeaCast multiplica el MSE por la varianza inversa de las estimaciones λ_i. Esta ecuación también se aplica a los forzantes antes de entrar a la red.

Aunque ambos enfoques buscan normalizar la magnitud de las distintas variables para equilibrar su influencia en el cálculo del error, puede apreciarse que SeaCast aplica una versión modificada, ligeramente más sencilla. A diferencia del método de Keisler, se evita el cálculo de estadísticos a gran escala y, en su lugar, se realiza la normalización agrupando los estados para cada entrada de datos. Además, SeaCast introduce una normalización adicional dividiendo por la varianza antes de ponderar según la latitud. Por último, calcula el promedio del error para cada estado desplegado, nivel y variable, dividiendo por el número de nodos oceánicos en cada nivel ($\frac{1}{G_l}$). Es importante destacar que, en la función de error, solo se consideran aquellos nodos que estén en el océano (representados en el sumatorio interno como $v \in G_{l(i)}$).

Pese a que la función de pérdida pueda parecer compleja debido a la presencia de varios sumatorios, conviene recordar que, en el contexto de este estudio, solo existe un único nivel y una única variable. Esto permite simplificar notablemente la función, pudiendo resumirla como:

$$L = \frac{1}{T_{\text{rollout}}} \sum_{t=1}^{T_{\text{rollout}}} \frac{1}{|G|} \sum_{v \in G} a_v \lambda \left(\hat{X}^t_v - X^t_v \right)^2. \tag{5.9}$$

5.4. Arquitectura de redes neuronales de grafos

Hasta ahora se ha explicado cómo se calculan las predicciones y el formato de las entradas de SeaCast. No obstante, aún no se ha abordado el eje central que permite realizar dichas predicciones: la GNN utilizada. La función de predicción se implementa como una serie de GNNs estructuradas bajo el esquema de codificador-procesador-decodificador, haciendo uso del concepto de red jerárquica y aprovechando el mecanismo de paso de mensajes. Esta sección se centra en describir la arquitectura de las redes de cada módulo, la función de cada operación y cómo se construyen los grafos de entrada a estas redes.

GraphCast

Como se ha mencionado anteriormente, SeaCast es el resultado de la consolidación de otros trabajos adaptados al ámbito oceanográfico. Una de las herencias más directas proviene de Oskarsson y sus colaboradores en el trabajo "Graph-based Neural Weather Prediction for Limited Area Modeling" [Oskarsson et al., 2023], cuyo repositorio de modelos es Neural-LAM. Este estudio constituye la base principal sobre la cual se construye SeaCast. Al igual que SeaCast y este trabajo, los modelos de Neural-LAM se centran en realizar predicciones a nivel regional frente a las predicciones globales climáticas tradicionales, aunque su enfoque está dirigido a la predicción de variables atmosféricas.

Uno de los aportes de este modelo es que presenta dos variantes de su Modelo de Área Limitada (LAM). La primera es GC-LAM, que deriva de GraphCast, desarrollado por Google DeepMind, adaptado a la región nórdica. GraphCast es un modelo de predicción climática a medio plazo a escala global cuya arquitectura principal se basa en GNNs, siguiendo la estructura codificador-procesador-decodificador. La función del codificador es transformar los datos de la rejilla latitud-longitud en nodos de un grafo multi-malla.

Un grafo multi-malla es una representación homogénea, que, a diferencia de las proyecciones geográficas de mapas, no presenta sesgos en los polos. Este grafo se crea a partir de un icosaedro regular, una figura geométrica con 20 caras triangulares, 12 vértices y 30 aristas. Esta representación ya había sido previamente introducida en otro trabajo de predicción climática por Keisler [Keisler, 2022] con el objetivo de representar una malla de puntos para su uso en GNNs.

GraphCast introduce la innovación de trabajar con un total de siete resoluciones de malla, siendo la malla base (M^0) el icosaedro regular descrito anteriormente. Las resoluciones posteriores se generan mediante iteraciones sobre esta malla base, dividiendo cada triángulo en otros cuatro más pequeños. Esto provoca que la séptima malla (M^6) tenga un total 49,692 nodos, siendo la resolución más fina [Lam et al., 2023]. La Figura 5.2 muestra las resoluciones de los distintos niveles de malla.

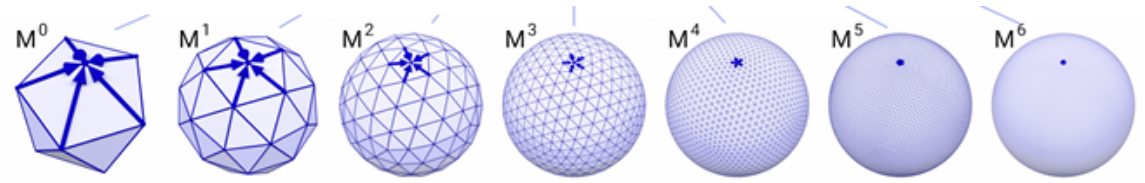


Figura 5.2: Niveles de resolución de las mallas empleadas por el modelo GraphCast. Fuente: GraphCast [Lam et al., 2023].

Una propiedad interesante de este grafo es que es posible superponer todas las aristas de las mallas sobre la malla de mayor resolución. Esto se debe a que las mallas se construyen de forma iterativa, lo que implica que los nodos de las resoluciones más bajas son subconjuntos de los nodos de las resoluciones superiores, de forma que la mayor resolución (M^6) contiene todos los nodos del resto de niveles.

Al combinar las aristas de todas las mallas en un único grafo se consigue integrar la transmisión de información a largas distancias, proporcionada por las resoluciones más gruesas, con la captura de patrones regionales o locales habilitada por las resoluciones más finas. De esta forma, la arquitectura multi-malla tiene una comunicación implícita que permite la transmisión de información entre nodos a distintas escalas usando la propagación de mensajes [Lam et al., 2023].

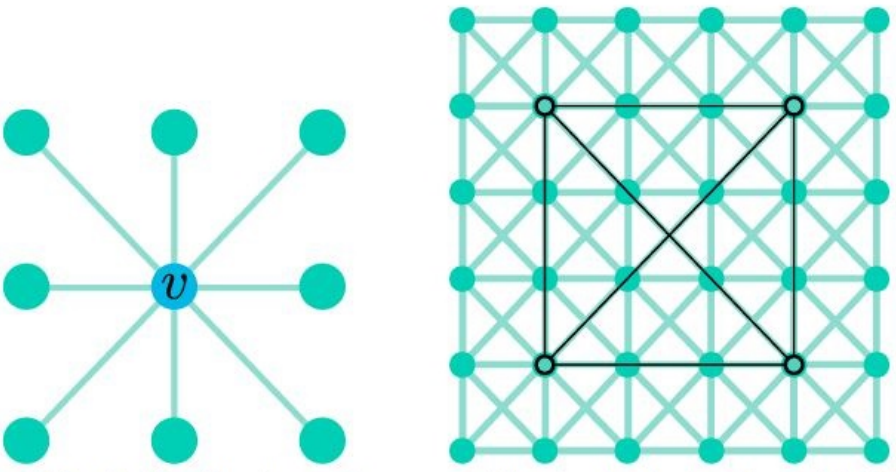
Neural-LAM y la red jerárquica

GraphCast logró resultados competitivos en comparativas frente a modelos numéricos, estableciéndose como una alternativa sólida basada en aprendizaje automático. Además, mostró un buen rendimiento en la predicción de fenómenos adversos (como ciclones o temperaturas extremas), pese a no haber sido entrenado específicamente para esta tarea [Lam et al., 2023].

No obstante, el equipo de Neural-LAM [Oskarsson et al., 2023] detectó un patrón circular en las predicciones realizadas por GraphCast en torno a nodos pertenecientes a distintos niveles de resolución, que son aquellos con más aristas entrantes. La hipótesis planteada sugiere que estos nodos asumen demasiadas responsabilidades, participando tanto en la transmisión de información a larga distancia como en la captura de patrones locales. Además, al tener un mayor número de vecinos, sus representaciones pueden verse alteradas debido a la cantidad de información.

La solución propuesta por Neural-LAM, que es la adoptada tanto en este proyecto como en SeaCat, introduce el modelo Hi-LAM, basado en un grafo de malla jerárquica. Al igual que en el grafo multi-malla de GraphCast, Hi-LAM opera con distintos niveles de resolución.

No obstante, este enfoque presenta ciertas diferencias, siendo la primera el proceso de construcción de las mallas. En lugar de trabajar con una figura geométrica como base, los nodos se disponen en una cuadrícula con filas y columnas sobre la zona de estudio. Cada nodo tiene ocho vecinos, excepto aquellos situados en los bordes, dado que se establecen conexiones bidireccionales con los nodos adyacentes en las direcciones vertical, horizontal y diagonal. Se construyen cuatro mallas distintas, donde cada malla triplica la distancia entre nodos del nivel de resolución anterior. Esta es otra de las diferencias conceptuales respecto a GraphCast, donde las mallas más altas en la jerarquía tienen un mayor nivel de resolución. De forma similar a GraphCast, los nodos de un nivel superior (l) son subconjuntos de los nodos del nivel inmediatamente inferior ($l-1$) y coinciden espacialmente con el nodo central de subcuadrículas de 3 x 3 nodos. La Figura 5.3 muestra un esquema gráfico de esta propiedad.



(a) Cada nodo de la malla está conectado con sus vecinos horizontalmente, verticalmente y diagonalmente.

(b) Alineamiento de nodos de malla ○ en el nivel de resolución l con nodos de malla ● en el de resolución $l-1$.

Figura 5.3: Ilustración del proceso de construcción de la malla de Hi-LAM. Fuente: Neural-LAM [Oskarsson et al., 2023].

Otra característica distintiva de Hi-LAM es que los grafos resultantes de distintas resoluciones no se combinan entre sí, aunque esto sería posible. En su lugar, se usa una estructura jerárquica que da lugar a una GNN jerárquica.

Para establecer comunicación entre niveles consecutivos, se introducen dos conjuntos de aristas adicionales, denominados $\mathcal{E}^{[l\nearrow l+1]}$ y $\mathcal{E}^{[l+1\searrow l]}$. Estas aristas se basan en la propiedad reflejada en la Figura 5.3, donde ciertos nodos actúan como centro de vecindades de 3 x 3 nodos en el nivel de resolución inferior. Esto evita la sobrecarga de conexiones que se presentaba en GraphCast con ciertos nodos, limitando cada nodo a nueve conexiones, ya que cada nodo del nivel inferior se conecta únicamente con el nodo más cercano del nivel superior. El conjunto $\mathcal{E}^{[l\nearrow l+1]}$ representa las conexiones ascendentes, mientras que $\mathcal{E}^{[l+1\searrow l]}$ es el mismo conjunto de aristas, transmitiendo la información desde los niveles más gruesos hasta los más finos.

En SeaCast [Holmberg et al., 2024], este conjunto de aristas se depura, excluyendo aquellas que crucen tierra durante al menos ocho celdas consecutivas (por ejemplo, aquellas que atraviesan las Islas Canarias). Este filtrado también se aplica sobre las aristas internas de cada nivel. Además, los nodos ubicados sobre la superficie terrestre también son eliminados de la malla base.

Tras analizar la construcción de la malla y la estructura jerárquica del grafo, se describe a continuación el proceso de codificador-procesador-decodificador, iniciando por el codificador. Es importante destacar que cada etapa del proceso cuenta con su propia GNN independiente.

5.5. Codificador

El primer paso en la predicción con GNNs consiste en transformar las entradas definidas sobre la rejilla de latitud-longitud a la estructura de malla descrita anteriormente. Para ello se crea un grafo bipartito de rejilla a malla, denotado como $\mathcal{G}_{G2M}$. Un grafo bipartito tiene la propiedad de que los nodos pueden dividirse en dos conjuntos disjuntos, de modo que todas las aristas conectan nodos únicamente de un conjunto con los del contrario [Holmberg et al., 2024].

Antes de comenzar el procesamiento principal es necesario crear representaciones vectoriales para las características de las celdas de la rejilla, los nodos de la malla y las aristas que los conectan. Para los nodos de la rejilla (denotados como v^G), se emplean como datos de entrada los dos estados oceánicos previos (en este caso, con la variable SST), los forzantes externos (en este caso, los componentes vectoriales del viento y el día del año codificado mediante funciones de seno y coseno) y las características estáticas (en este caso, la batimetría y las coordenadas). A partir de esta información se obtiene un vector v_s^G para cada celda. Los nodos de la malla (denotados como v^M) y las aristas solo reciben características estáticas inicialmente. En el caso de los nodos de la malla, corresponden a sus coordenadas proyectadas, a partir de las cuales se obtiene un vector $v_r^{M[l]}$ para cada nodo. Los subíndices en estas notaciones indican el rol de cada nodo en la relación, donde s representa el nodo origen (*source*) y r el nodo destino (*receiver*). Sin embargo, dependiendo de la etapa del proceso, este papel puede cambiar.

En cuanto a las aristas, se distinguen cinco tipos de aristas según el origen y destino de los nodos que conectan en la arquitectura. No obstante, todas emplean como características su longitud y la diferencia de las coordenadas de los nodos que conectan. Los cinco tipos de aristas existentes son conexiones desde la rejilla hacia la malla inferior (denotadas como $e_{s\to r}^{\mathrm{G2M}}$), conexiones bidireccionales dentro del nivel de cada malla ($e_{s\to r}^{[l\leftrightarrow l]}$),

conexiones en sentido ascendente entre niveles de la malla ($e_{s\to r}^{[l\nearrow l+1]}$), conexiones en sentido descendente entre distintos niveles de la malla ($e_{s\to r}^{[l\searrow l-1]}$) y desde la malla inferior hacia la rejilla (denotadas como $e_{r\to s}^{\mathrm{M2G}}$). Para cada componente se emplea una MLP distinta que incorpora una capa oculta con Swish como función de activación no lineal, seguido de una normalización por capas, lo que ayuda a mejorar la estabilidad y eficiencia del entrenamiento [Oskarsson et al., 2023]. Cabe destacar, que tanto los nodos de la malla como las aristas, internas o entre niveles, tienen su propia MLP en cada nivel, así como MLPs independientes en cada fase del procesamiento.

La fórmula de Swish es:

$$f(x) = x \cdot \sigma(\beta x) = \frac{x}{1 + e^{-\beta x}}, \tag{5.10}$$

donde x es la salida lineal de la capa oculta que se calcula como el producto de los pesos W y la entrada h, sumando el sesgo b:

$$x = W \cdot h + b, \tag{5.11}$$

y β es un parámetro que puede ser fijo o entrenable. Swish se caracteriza por ser una función no monotónica, a diferencia de la sigmoide ($\sigma(x)$) o ReLU ($0, \max(x)$). La sigmoide es monotónica, pero puede sufrir saturación de valores, lo que afecta a la propagación del gradiente. ReLU es una función monótona no decreciente y no suave, ya que no tiene derivada para $x = 0$ y su derivada es cero para $x < 0$. Swish corrige ambos aspectos al ser una función suave con derivada continua, donde β controla el grado de interpolación entre un comportamiento lineal y uno similar a ReLU. Esto genera una pequeña curva antes del cambio de pendiente en $x = 0$, permitiendo que los valores negativos de x contribuyan al entrenamiento [Ramachandran et al., 2017].

La misma configuración de perceptrones multicapa se mantiene para la arquitectura de cada una de las GNNs utilizadas. En este caso, las redes empleadas son conocidas como redes de interacción (*interaction networks*), diseñadas originalmente para predecir dinámicas físicas a partir de relaciones de objetos e influencias externas. Estas redes surgieron como propuesta a la falta de métodos capaces de incorporar la falta de métodos capaces de incorporar información relacional. Cada capa de GNN usa dos MLPs, en línea con la estructura de las redes de interacción. Una función modela la relación entre un par de nodos (emisor y receptor) y los efectos de su interacción, y otra función a nivel nodo recopila todas las interacciones recibidas que se han calculado con la función anterior para determinar el siguiente estado del nodo receptor [Battaglia et al., 2016].

De esta forma, el paso de mensajes de Hi-LAM [Oskarsson et al., 2023] en la fase de codificación, tal como se utiliza en SeaCast, comienza con la ejecución del paso de mensajes en el grafo bipartito. Para ello utiliza el conjunto de aristas que conecta la rejilla con la malla inferior, así como los nodos de la rejilla y los de dicha malla, correspondiente al nivel 1. Todas las actualizaciones se realizan conservando parte del valor anterior de cada representación mediante conexiones residuales. En primer lugar, se actualiza la representación de las aristas del grafo bipartito ($e_{s\to r}^{\mathrm{G2M}}$) con la siguiente expresión:

$$\tilde{e}_{s\to r}^{\mathrm{G2M}} = \mathrm{MLP}_E^{G2M}([e_{s\to r}^{\mathrm{G2M}}, v_s^G, v_r^{M[1]}]), \tag{5.12}$$

$$e_{s\to r}^{\mathrm{G2M}} = e_{s\to r}^{\mathrm{G2M}} + \tilde{e}_{s\to r}^{\mathrm{G2M}}, \tag{5.13}$$

donde para cada arista que conecta un nodo de la rejilla con un nodo de la malla inferior es actualizada mediante una MLP que toma como entrada la concatenación ($[\cdot]$) de su propia

representación inicial, junto con las representaciones de los nodos que enlaza (siendo $v_r^{M[1]}$ la representación de un nodo de la malla de resolución más detallada que es a donde apunta la arista). El valor de salida de esta MLP se suma a la representación inicial de la arista, completando su actualización. Esta primera etapa se corresponde a la función del modelado de las redes de interacción. Tras ello se actualizan las representaciones de cada uno de los nodos de la rejilla de la siguiente forma:

$$v_s^G = \text{MLP}_G^{G2M}(v_s^G), \tag{5.14}$$

donde solo se utiliza únicamente la información de la propia celda para actualizar su representación. Aunque pueda parecer que las celdas de la rejilla no desempeñan un papel relevante en esta etapa, tendrán una función importante en la fase de decodificación. Para concluir esta primera fase de paso de mensajes, se realiza la actualización de los nodos de la malla de nivel inferior, mediante una operación que puede interpretarse como la fase de actualización a nivel de nodo de la red de interacción, descrita por:

$$v_r^{M[1]} = v_r^{M[1]} + \text{MLP}_{\text{Enc.}}^{M[1]}\left(\left[v_r^{M[1]}, \sum_{s:(s,r)\in\mathcal{E}^{G2M}} \tilde{e}_{s\to r}^{\text{G2M}}\right]\right), \tag{5.15}$$

donde la nueva representación de cada nodo se calcula sumando su vector actual con el resultado de una MLP, que toma como entrada la concatenación de su propia representación y la suma de las actualizaciones provenientes de todas las aristas entrantes desde la rejilla hasta él. Todas las actualizaciones descritas individualmente para cada componente se realizan de forma simultánea.

Una de las diferencias destacadas entre Hi-LAM y GraphCast es que la fase de codificación de GraphCast finaliza en este punto. En cambio, Hi-LAM extiende esta fase inicial para garantizar que los niveles superiores de la malla tengan información, ya que, de lo contrario, conservarían únicamente sus representaciones iniciales. Aunque dicha información resulta relevante, no es la fuente principal de las predicciones. Por ello, se realizan operaciones similares en los distintos niveles superiores para poder aprovechar de manera más eficiente los recursos computacionales. Es importante tener en cuenta que las siguientes etapas comparten una estructura operativa similar. Lo único que varían son los conjuntos de nodos y aristas actualizados, ya que la forma de entrada de cada MLP se mantiene constante para las aristas y nodos de la malla. Una vez dicho esto, la fase de codificación finaliza con:

$$\tilde{e}_{s\to r}^{[l\nearrow l+1]} = \text{MLP}_{Enc.}^{[l\nearrow l+1]}([e_{s\to r}^{[l\nearrow l+1]}, v_s^{M[l]}, v_r^{M[l+1]}]), \tag{5.16}$$

$$e_{s\to r}^{[l\nearrow l+1]} = e_{s\to r}^{[l\nearrow l+1]} + \tilde{e}_{s\to r}^{[l\nearrow l+1]}, \tag{5.17}$$

$$v_r^{M[l+1]} = v_r^{M[l+1]} + \text{MLP}_{\text{Enc.}}^{M[l+1]}\left(\left[v_r^{M[l+1]}, \sum_{s:(s,r)\in\mathcal{E}^{[l\nearrow l+1]}} \tilde{e}_{s\to r}^{[l\nearrow l+1]}\right]\right), \tag{5.18}$$

donde, para cada nivel de resolución, se actualizan las aristas que conectan desde el nivel inferior hacia el actual, así como los nodos del propio nivel $l+1$. De esta forma cada nivel, de resolución se beneficia de la información ya actualizada del nivel anterior. Se observa que el funcionamiento es idéntico a la primera fase, aunque se aplica únicamente a los

distintos niveles de la malla, por lo que ya no se opera con el grafo bipartito ni con los nodos de la rejilla.

Esta forma de actualizar las representaciones también se aplica en la fase del procesador, donde las aristas se actualizan utilizando tanto su propia información como la de los nodos que conectan. De forma similar, cada nodo combina su información inicial con la suma de las salidas de la MLP correspondiente a todas las aristas entrantes. Esto último también sucede en la fase de decodificación, aunque estas otras dos fases se describen con más detalle a continuación.

Cada actualización de nivel de resolución se procesa de forma secuencial hasta alcanzar el nivel superior de la malla, correspondiente al de menor detalle. Por ello, la fase de codificación puede interpretarse como un recorrido desde las celdas de la rejilla hasta el nivel superior de la malla.

5.6. Procesador

En la etapa de procesamiento, el objetivo es aplicar operaciones sobre las representaciones latentes de los distintos niveles de forma secuencial [Holmberg et al., 2024].

Esta fase puede considerarse la más relevante para la predicción, dado que es donde se actualizan las representaciones latentes de los datos y se capturan los patrones en ellos de forma más profunda. Durante esta etapa, se definen las capas de procesamiento, encargadas de recorrer la estructura de la malla para actualizar las representaciones de forma secuencial a lo largo de la jerarquía. Cabe recordar que, al finalizar la fase de codificación, el procesamiento se detuvo en el nivel superior de la malla. La arquitectura de codificador-procesador-decodificador debe interpretarse como un proceso continuo, en el que las transiciones entre fases se realizan de forma ordenada y secuencial.

Por lo tanto, en Hi-LAM [Oskarsson et al., 2023], esta fase comienza por el nivel superior de la malla, donde se partirá del caso donde hay una única capa de procesamiento. Esta capa está compuesta por una serie de capas de GNN independientes para cada nivel. Cada capa de procesamiento (denominada como k) comienza con la actualización de representaciones dentro del propio nivel de resolución, descrita como la siguiente manera:

$$\tilde{e}_{s\to r}^{[l\leftrightarrow l]} = \mathrm{MLP}_{k\downarrow}^{[l\leftrightarrow l]}([e_{s\to r}^{[l\leftrightarrow l]}, v_s^{M[l]}, v_r^{M[l]}]), \tag{5.19}$$

$$e_{s\to r}^{[l\leftrightarrow l]} = e_{s\to r}^{[l\leftrightarrow l]} + \tilde{e}_{s\to r}^{[l\leftrightarrow l]}, \tag{5.20}$$

$$v_r^{M[l]} = v_r^{M[l]} + \mathrm{MLP}_{k\downarrow}^{[l]}\left(\left[v_r^{M[l]}, \sum_{s:(s,r)\in\mathcal{E}^{[l\leftrightarrow l]}} \tilde{e}_{s\to r}^{[l\leftrightarrow l]}\right]\right), \tag{5.21}$$

donde en primer lugar se realiza la actualización de las representaciones de las aristas internas del nivel, para posteriormente actualizar cada nodo usando la información del propio nivel. Antes de pasar al siguiente nivel, la capa de procesamiento realiza una segunda tarea, donde se actualizan las representaciones de las aristas entre niveles en sentido descendente, así como los nodos del nivel inferior, de forma que:

$$\tilde{e}_{s\to r}^{[l\searrow l-1]} = \mathrm{MLP}_{k\downarrow}^{[l\searrow l-1]}([e_{s\to r}^{[l\searrow l-1]}, v_s^{M[l]}, v_r^{M[l-1]}]), \tag{5.22}$$

$$e_{s\to r}^{[l\searrow l-1]} = e_{s\to r}^{[l\searrow l-1]} + \tilde{e}_{s\to r}^{[l\searrow l-1]}, \tag{5.23}$$

$$v_r^{M[l-1]} = v_r^{M[l-1]} + \text{MLP}_{k\downarrow}^{M[l-1]} \left(\left[v_r^{M[l-1]}, \sum_{s:(s,r)\in\mathcal{E}^{[l\searrow l-1]}} \tilde{e}_{s\to r}^{[l\searrow l-1]} \right] \right). \tag{5.24}$$

Una vez completada esta actualización descendente entre niveles, se considera que la capa de procesamiento ha finalizado su ejecución en el nivel l. Este procedimiento se repite en cada nivel hasta llegar al nivel de menor resolución de la malla, donde no es posible llevar a cabo la actualización de representaciones inter-nivel descendente, por lo que solo se realiza la primera parte. Al llegar a este punto, la capa de procesamiento ha completado la primera mitad de su ejecución. La siguiente fase consiste en recorrer la jerarquía de niveles en sentido ascendente para actualizar nuevamente las representaciones propias de cada nivel, así como las inter-nivel ascendentes. Dado que la actualización intra-nivel es idéntica a las ecuaciones (5.19–5.20) cambiando las MLPs, a continuación, se detalla la actualización de representaciones de aristas ascendentes y de nodos del nivel superior:

$$\tilde{e}_{s\to r}^{[l\nearrow l+1]} = \text{MLP}_{k\uparrow}^{[l\nearrow l+1]}([e_{s\to r}^{[l\nearrow l+1]}, v_s^{M[l]}, v_r^{M[l+1]}]), \tag{5.25}$$

$$e_{s\to r}^{[l\nearrow l+1]} = e_{s\to r}^{[l\nearrow l+1]} + \tilde{e}_{s\to r}^{[l\nearrow l+1]}, \tag{5.26}$$

$$v_r^{M[l+1]} = v_r^{M[l+1]} + \text{MLP}_{k\uparrow}^{M[l+1]} \left(\left[v_r^{M[l+1]}, \sum_{s:(s,r)\in\mathcal{E}^{[l\nearrow l+1]}} \tilde{e}_{s\to r}^{[l\nearrow l+1]} \right] \right). \tag{5.27}$$

Una vez finalizada la ejecución de la capa de procesamiento, podría considerarse que la fase de procesamiento ha finalizado. No obstante, es posible emplear un número mayor de capas de procesamiento para profundizar en la actualización de las representaciones, configurando así un bucle de varias iteraciones. Como se mencionó anteriormente, las capas de GNN que conforman cada capa de procesamiento son completamente independientes entre sí y respecto a otras capas. Esto permite una mayor flexibilidad, así como evitar compartir pesos entre distintos niveles de escala.

5.7. Decodificador

El proceso de decodificación funciona de forma inversa y análoga al del codificador. Su propósito es transformar los datos desde las representaciones en la malla a la rejilla latitud-longitud, haciendo uso de un grafo bipartito de malla a rejilla (denotado como $\mathcal{G}_{M2G}$), a partir del cual se obtiene el resultado de la predicción [Holmberg et al., 2024].

Teniendo en cuenta que el procesamiento de Hi-LAM [Oskarsson et al., 2023] finaliza en el nivel superior de menor resolución de la malla, el decodificador realiza un barrido en sentido descendente hasta el nivel inferior, el cual contiene toda la información de las distintas escalas de la malla, resumida en una representación de mayor resolución. Las últimas actualizaciones realizadas son las siguientes para cada nivel:

$$\tilde{e}_{s\to r}^{[l\searrow l-1]} = \text{MLP}_{\text{Dec.}}^{[l\searrow l-1]}([e_{s\to r}^{[l\searrow l-1]}, v_s^{M[l]}, v_r^{M[l-1]}]), \tag{5.28}$$

$$v_r^{M[l-1]} = v_r^{M[l-1]} + \text{MLP}_{\text{Dec.}}^{M[l-1]} \left(\left[v_r^{M[l-1]}, \sum_{s:(s,r)\in\mathcal{E}^{[l\searrow l-1]}} \tilde{e}_{s\to r}^{[l\searrow l-1]} \right] \right), \tag{5.29}$$

donde se observa que en esta última fase, antes de transformar los datos de la malla a la rejilla latitud-longitud, se actualizan las representaciones vectoriales de los nodos con el objetivo de incorporar toda la información obtenida en el nivel de resolución final. A diferencia de las etapas anteriores, en este paso no se actualizan las representaciones de las conexiones. Basta con calcular su actualización de forma implícita, ya que es suficiente para actualizar los nodos. Al tratarse del paso final del modelo, las aristas de la jerarquía y de los niveles no se volverán a usar.

Por último, se lleva a cabo un proceso inverso a las ecuaciones (5.12, 5.15), con el fin de obtener una representación actualizada de las celdas de la rejilla. Este paso utiliza las aristas del grafo bipartito $\mathcal{G}_{M2G}$ actualizadas, el estado de los nodos del nivel de mayor resolución de la jerarquía de la malla, y las representaciones de las celdas tras la transformación previa de rejilla a malla. El proceso sería:

$$\tilde{e}^{\mathrm{M2G}}_{s\to r} = \mathrm{MLP}^{M2G}_{E}([e^{\mathrm{M2G}}_{s\to r}, v^{M[1]}_{s}, v^{G}_{r}]), \tag{5.30}$$

$$v^{G}_{r} = v^{G}_{r} + \mathrm{MLP}^{M2G}_{\mathrm{E}}\left(\left[v^{G}_{r}, \sum_{s:(s,r)\in\mathcal{E}^{M2G}} \tilde{e}^{\mathrm{M2G}}_{s\to r}\right]\right). \tag{5.31}$$

Una característica fundamental de la predicción es que el modelo realmente no aprende a generar el estado predicho desde cero. En realidad, lo que aprende es a estimar la diferencia entre estados, por lo que la predicción se podría ver como:

$$\hat{X}^{t}_{r} = f\left(X^{t-2:t-1}, F^{t}\right)_{r} = X^{t-1}_{r} + \mathrm{MLP}^{\mathrm{pred}}(v^{G}_{r}), \tag{5.32}$$

donde v^{G}_{r} representa el estado de la rejilla latitud-longitud para la celda r, obtenido desde el grafo $\mathcal{G}_{M2G}$. Para calcular el estado predicho se emplea una conexión residual que parte del estado anterior. La MLP utilizada en la predicción es la única que no incorpora normalización de capa tras la función de activación y, además, no estima el estado completo, sino la diferencia respecto al instante anterior. La Figura 5.4 muestra un esquema que resume las tres etapas del modelo.

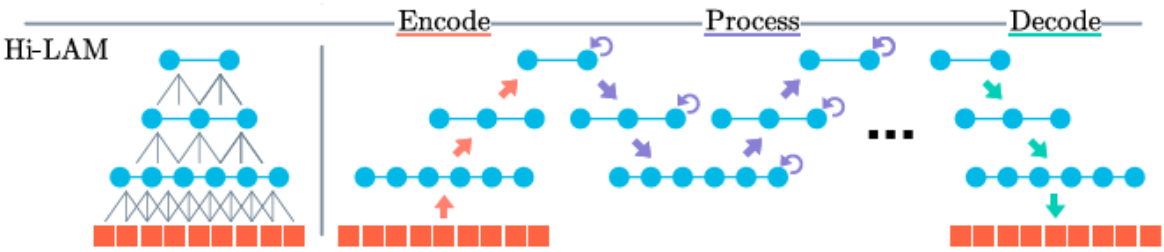


Figura 5.4: Esquema de funcionamiento de la red jerárquica Hi-LAM. Fuente: Neural-LAM [Oskarsson et al., 2023].

5.8. Flujo de trabajo operativo de SeaCast

Tras haber expuesto los fundamentos teóricos del modelo y su arquitectura, en esta sección se describen las distintas etapas operativas que se llevan a cabo para generar

una predicción. Es decir, se detallan los pasos para la ejecución práctica del código, que abarca desde la descarga de datos hasta la unificación y evaluación de los conjuntos de predicciones. La Figura 5.5 muestra un esquema general que resume el flujo operativo del sistema.

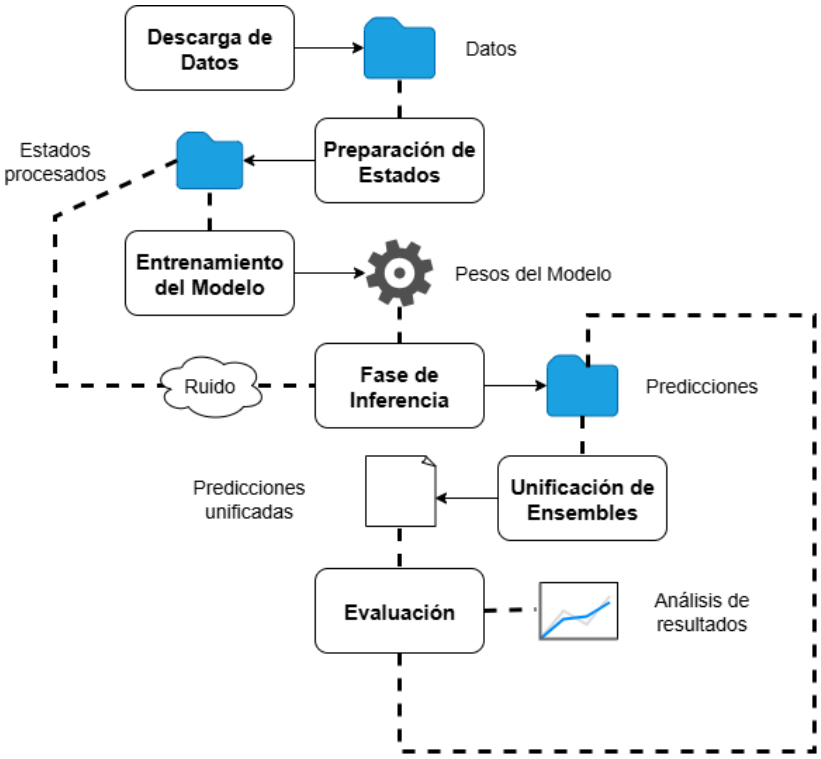


Figura 5.5: Esquema general de flujo operativo de SeaCast.

El primer paso para generar una predicción es la descarga de los datos. Como se explicó en la sección 4, se descargan tres tipos de datos: oceanográficos, atmosféricos y de batimetría. Los datos del proyecto se organizan en una jerarquía de carpetas. Por ejemplo, al trabajar con datos del océano Atlántico la carpeta principal dentro de la jerarquía tiene el nombre de `atlantic`. En esta etapa, los datos aún no han sido procesados para adaptarlos a la estructura de la red, por lo que se almacenan en una subcarpeta llamada `raw`.

A partir del tercer nivel de la jerarquía, los datos se categorizan según su fuente. Los datos atmosféricos se encuentran en la carpeta `era5` y los oceanográficos en la carpeta `reanalysis`. Cada archivo dentro de estas dos carpetas lleva como nombre la fecha de las observaciones que contiene. Por otro lado, los datos estáticos como la batimetría y los datos auxiliares creados manualmente, se almacenan en el segundo nivel de la jerarquía bajo la carpeta `static`, también dentro de `atlantic`. El orden de descarga comienza con los datos estáticos, ya que en esta etapa también se generan objetos que se utilizan para procesar la descarga de los datos de oceanografía y forzantes atmosféricos. La función `download_data` gestiona el proceso de descarga de los distintos tipos de datos.

Una vez descargados, los datos deben ser procesados. Durante esta fase se dividen los datos en los conjuntos de entrenamiento, validación y prueba según rangos de fechas. Esta división se realiza usando un módulo llamado `prepare_states`, que organiza los datos en una carpeta ubicada en el segundo nivel de la jerarquía, llamada `samples`. Esta carpeta se divide en otras tres, una para cada conjunto de datos de las fases del entrenamiento.

Los archivos contenidos en estas carpetas cuentan con la forma explicada en la sección 5, la cual varía en función de la etapa.

Se distinguen dos tipos de archivo dentro de estas carpetas: los correspondientes a la oceanografía y los de forzantes. Los archivos oceanográficos se identifican porque su nombre comienza por `rea_data`, seguido de la fecha del primer estado objetivo que almacenan. Además, estos archivos contienen los dos estados iniciales necesarios para iniciar la predicción. Los archivos de forzantes poseen la misma estructura en cuanto a su contenido, y siguen la misma lógica de nomenclatura que los de oceanografía, pero comienzan por `forcing_`.

Una vez procesados y divididos los datos en conjuntos, es necesario preparar ciertos elementos del modelo antes de iniciar el entrenamiento. En primer lugar, se obtienen las características estáticas de las celdas de la rejilla (batimetría y coordenadas), las cuales se almacenan en un tensor mediante el módulo `create_grid_features`. En segundo lugar, se calculan estadísticas (media y desviación estándar) que el modelo utiliza para normalizar las entradas oceánicas y atmosféricas del modelo, de modo que estos valores ya estén precalculados. Además, se calcula la media y desviación estándar de las diferencias entre los pasos temporales para la variable SST, que servirán para reescalar la salida de la red. Estas estadísticas se obtienen con el módulo `create_parameter_weights`. Finalmente, antes de comenzar el entrenamiento, se crea la estructura de la malla jerárquica empleada por la red con el módulo `create_mesh`. En esta etapa, se generan archivos para las características estáticas de las conexiones de la malla, así como de las conexiones de los grafos bipartitos que conectan con la rejilla latitud-longitud. También se guardan las conexiones entre nodos para los distintos tipos de niveles, almacenando dos listas paralelas en las que los nodos ubicados en la misma posición están conectados entre sí.

Una vez cumplidos los requisitos previos para usar el modelo, los datos de entrenamiento se utilizan para ejecutar SeaCast. Durante esta fase, los pesos de la red obtenidos del entrenamiento se almacenan en una carpeta gestionada por Weights & Biases. Esta herramienta permite guardar información sobre la ejecución, incluyendo los pesos y las predicciones realizadas. En este caso, al tratarse de la fase de entrenamiento del modelo, es importante guardar los pesos para poder utilizarlos posteriormente en la fase de inferencia. La carga de datos durante el entrenamiento es gestionada por un módulo llamado `weather_dataset`, que suministra los datos día a día a medida que el modelo avanza.

Con los pesos del modelo, se pueden realizar predicciones sobre el conjunto de prueba. El modelo se utiliza en fase de evaluación, ejecutando cinco predicciones distintas con los mismos datos de prueba. La diversidad se introduce durante la ejecución de la predicción, en el momento en el que `weather_dataset` carga el fichero de datos correspondiente a un día de datos oceanográficos. En este proceso, se toman los dos estados iniciales (correspondientes a los dos primeros elementos del archivo) y se les añade ruido aleatorio antes de convertirlos en un tensor. Los datos contaminados alimentan al modelo, que realiza predicciones con un horizonte de 15 días para cada entrada diaria. Las ejecuciones del modelo se realizan de forma manual, lo que facilita agrupar correctamente las carpetas de predicciones según su pertenencia a un conjunto, en función del ruido aplicado. Automatizar esta tarea habría sido costoso debido al tamaño de las predicciones. Los archivos de predicción contienen únicamente los 15 estados predichos, y su nombre corresponde a la fecha del primer día predicho. Es importante destacar que el ruido solo se aplica en estados iniciales oceanográficos, no a los datos de forzantes externos.

Las carpetas que contienen las predicciones de cada conjunto se procesan con el módulo `ensemble_unifier`, donde se combinan las predicciones de todos los modelos para cada

día en único archivo, siguiendo el criterio establecido. Por el momento, únicamente se ha implementado la opción de unificación mediante el cálculo de la media diaria entre las predicciones generadas por los distintos miembros del conjunto. Cabe señalar que, aunque la Figura 5.5 muestra un símbolo de un fichero, en realidad el resultado de la unificación se guarda en una carpeta. Cada archivo dentro de esta carpeta corresponde a un día y contiene la información resumida de los modelos. Por lo tanto, dicho símbolo solo pretende representar la salida final unificada.

Por último, los resultados de la combinación del conjunto se evalúan mediante métricas deterministas aplicadas a lo largo del horizonte temporal de la predicción. También se emplean métricas probabilísticas para evaluar el conjunto de modelos, considerando la incertidumbre y la distribución de probabilidad generada. Ambos tipos de métricas se obtienen de la librería WeatherBench-X y se emplean en notebooks de Jupyter, lo que facilita la comparación visual de resultados. Para calcular las métricas, es necesario primero computar las estadísticas que componen a la métrica, indicando tanto el conjunto de datos de observaciones reales (*target*), como el de predicciones *predictions*. Asimismo, es necesario especificar las dimensiones a reducir, es decir, aquellas sobre las cuales se realiza el promedio de la métrica. Finalmente, hay que ignorar los valores nulos para evitar que el resultado de las métricas se vea sesgado, ya que estos corresponden a zonas terrestres sin datos.

5.9. Cambios realizados sobre SeaCast y métricas

SeaCast es un proyecto realizado a una escala mayor que este trabajo, por lo que fue necesario realizar una serie de modificaciones para poder ejecutar el modelo. En la sección de 4 se describen las diferencias principales en los conjuntos de datos. Sin embargo, en esta sección se explican los cambios realizados en el código, así como las funciones añadidas para su uso en aprendizaje por conjuntos.

Las primeras modificaciones del código se realizaron con el objetivo de adaptarlo a las nuevas fuentes de datos, dado que los productos y variables utilizadas estaban definidas explícitamente mediante constantes, en lugar de como argumentos configurables. Por ello, se modificaron las funciones de descarga de datos, aunque tratando de conservar las funciones originales de SeaCast. El cambio más significativo fue en la función de descarga de datos estáticos, puesto que en este caso no se disponía de ellos y se obtienen desde una fuente distinta a CMEMS, la utilizada en el proyecto original. Por ello, se creó una nueva función para obtener la batimetría. Sin embargo, para el resto de los casos fue suficiente con añadir un argumento que permita seleccionar si la región de descarga corresponde al Atlántico o al Mediterráneo. En el caso de los datos oceanográficos, hubo que copiar la función original y modificar el identificador, versión y variables a extraer del conjunto de datos, mientras que, en el caso de los forzantes atmosféricos, bastó con cambiar las variables utilizadas.

En esta etapa también se realizaron cambios en ciertas constantes que definían el número de variables oceanográficas, el tamaño de los forzantes atmosféricos, así como las coordenadas y dimensiones de la rejilla latitud-longitud. En la máscara de forzantes fronterizos se rellenaron los valores con ceros para indicarle al modelo que, debe utilizar sus predicciones durante la fase de autorregresión, en lugar de alimentarse de los valores observados en determinadas zonas. Además, se eliminó la máscara de topografía dinámica media.

Afortunadamente, la presencia de variables multinivel en SeaCast no representó un

problema a lo largo del proyecto, ya que cada nivel se organiza como una variable independiente. Sin embargo, para mantener la compatibilidad con el proyecto original y permitir su uso en casos con variables multinivel, la máscara de tierra-océano contiene una dimensión adicional.

Adicionalmente, fue necesario adaptar la función de entrenamiento para poder ejecutarla en una única GPU, ya que SeaCast fue diseñado para entrenarse en paralelo utilizando un total de 32 tarjetas gráficas [Holmberg et al., 2024]. En lugar de usar una estrategia *Fully Shared Data Parallel* (FSDP), se comprueba el número de tarjetas gráficas disponibles para el entrenamiento y se utiliza una estrategia automática que permite que `pytorch` seleccione la mejor opción.

Hasta este punto se han comentado los cambios realizados para adaptar el proyecto a los datos, pero no se ha explicado cómo se agrega la diversidad al modelo. Para ello se creó un archivo que contiene tres tipos de ruido, que se describirán posteriormente: ruido Gaussiano, ruido de Perlin y ruido de Perlin en su versión fractal.

Este archivo (`noise.py`) define varias clases que heredan de una clase base de ruido. Cada clase específica invoca funciones de generación de ruido de distintas librerías, y se encarga tanto de la integración del ruido en los datos originales como de asegurar que las partes correspondientes a la tierra no se vean contaminadas durante este proceso.

Estas clases también gestionan las dimensiones de los datos, ya que inicialmente los datos se encuentran en dos dimensiones (día, puntos oceánicos), en lugar de en tres dimensiones estructuradas según la rejilla espacial (día, latitud, longitud), necesarias para aplicar el ruido. Además, estas dimensiones junto con la máscara tierra-océano, facilitan el filtrado de los puntos oceánicos. Una vez aplicado el ruido, los datos se adaptan nuevamente a las dimensiones esperadas por la red. Otra clase, encargada de la generación de ruido (ubicada en el archivo `noise_generator.py`), actúa como una interfaz para las diferentes funciones definidas en las clases anteriores. Su función es recibir el tipo de ruido a generar como una cadena de texto (`string`) y crear la clase de ruido específica que lo aplica.

La clase de entrenamiento fue modificada para aplicar el ruido durante la ejecución al llamar al cargador de datos. Aunque este método presenta la desventaja de la falta de reproducibilidad por la eliminación de cualquier semilla para garantizar la aleatoriedad, permite evitar la gestión de múltiples conjuntos de datos con ruido aplicado. Esto reduce la carga de almacenamiento en disco, lo cual es relevante dado que las predicciones y los datos llegan a ocupar más de 100 GB. La otra opción habría sido disponer de distintas carpetas con muestras ruidosas, lo que también hubiera implicado tener que invertir un tiempo significativo en la alteración de datos fuera de la fase de inferencia.

La librería WeatherBench-X, que proporciona acceso a las métricas de evaluación, también fue modificada. Sus funciones esperan como entrada conjuntos de datos de `Xarray`, que suelen estar almacenados en formato NetCDF4. Sin embargo, como se ha mencionado anteriormente, los datos del proyecto están organizados en carpetas que contienen archivos `numpy`, por lo que fue necesario sobrescribir la función de carga de datos.

Para ello se creó una nueva clase base (`NpLoaders`) que hereda de la clase base `DataLoader` proporcionada por WeatherBench-X. Esta clase base está diseñada para que cualquier usuario pueda crear su propio cargador de datos según sus necesidades. La clase creada recibe como argumentos el directorio donde se encuentran los archivos `numpy`, que serán concatenados en un conjunto de datos de `Xarray`, además del archivo con las coordenadas de latitud y longitud, así como la máscara tierra-océano. El flujo de esta operación comienza con la validación de que existen el directorio y los archivos indicados

como argumento. A continuación, se recorre dicho directorio con los archivos a cargar para obtener el nombre y fecha de cada archivo. Utilizando la máscara de tierra-océano, se cargan los datos de forma que se preservan como valores nulos en las zonas de tierra, y asegurando que únicamente contienen estados objetivos (*targets*) o predicciones (*predictions*). Finalmente, se utiliza la información de las fechas extraídas, el contenido de los archivos y las coordenadas para generar el conjunto de datos de `Xarray`. WeatherBench-X trabaja con dos tipos de cargadores de datos en función de su naturaleza: los estados objetivos y las predicciones. Las coordenadas empleadas en cada caso se muestran en la Tabla 5.9.

Nombre	**Tipo de datos**	**Forma**	**Descripción**
latitude	float64	(latitud)	Valores de latitud geográfica en grados.
longitude	float64	(longitud)	Valores de longitud geográfica en grados.
init_time	datetime64[ns]	(init_time)	Tiempo de inicio de la predicción.
lead_time	timedelta64[ns]	(lead_time)	Tiempo desde el inicio de la predicción, horizonte temporal.
valid_time	datetime64[ns]	(init_time, lead_time)	Tiempo válido de la predicción, calculado como init_time + lead_time

Tabla 5.3: Descripción de las coordenadas del conjunto de datos xarray.

La Tabla 5.9 muestra tres coordenadas que no están presentes originalmente en nuestros datos: `init_time`, `lead_time` y `valid_time`. La coordenada `init_time` indica la fecha en la que comenzó la predicción. Por su parte, `lead_time` representa el intervalo de tiempo transcurrido desde el inicio de la predicción para cada estado previsto. Finalmente, `valid_time` indica la fecha para la cual la predicción es válida, y se calcula sumando `init_time` y `lead_time`.

Las fechas de los archivos se extrajeron para calcular estas tres coordenadas restantes. Recordando la nomenclatura de los archivos procesados, aquellos que contenían los estados iniciales y objetivos incluyen en su nombre la fecha del primer estado objetivo. Dado que la ejecución de la predicción comienza el día anterior (correspondiente al segundo estado inicial), se debe restar un intervalo de 24 horas a la fecha de cada archivo para obtener el tiempo de inicio.

Una vez obtenida esta coordenada, calcular los intervalos desde el tiempo de inicio hasta los estados objetivos resulta relativamente sencillo. Consiste en generar un vector de 15 elementos para cada tiempo de inicio, donde el primer elemento corresponde a 24 horas y los siguientes incrementan este intervalo en múltiplos de 24 horas. Aunque la escala temporal de las predicciones es diaria, las unidades utilizadas son nanosegundos.

Por último, la coordenada de tiempo válido indica cuando está disponible la predicción y se obtiene sumando cada tiempo de inicio con los intervalos de predicción. Esta coordenada es exclusiva del conjunto de datos de las observaciones reales (cargadas con `TargetsFromNumpy`), ya que en estos datos la información temporal se encuentra disponible de antemano como referencia. En cambio, para las predicciones (cargadas con `PredictionsFromNumpy`) se trata de una información derivada de la ejecución del modelo, por lo que no se incluye.

Aunque la explicación de la creación de estas coordenadas se ha centrado en la carga de datos reales, el mecanismo es el mismo para los archivos de predicción, ya que siguen una nomenclatura similar. La única diferencia es que no incluyen estados iniciales como parte del archivo, por lo que no se eliminan los dos primeros estados al cargar los datos. Esta lógica se gestiona mediante un filtro que comprueba el tamaño de la dimensión correspondiente al número de días. Este filtro es gestionado por la clase base de `NpLoaders`, que contiene todas las funciones de carga y procesamiento de datos para su integración en un conjunto de datos de `Xarray`. Las clases específicas de cargadores para datos reales y de predicción se limitan a llamar a la función de construcción del conjunto de datos, indicando si debe calcularse la coordenada de `valid_time` o no.

Para poder emplear las métricas probabilísticas, se creó una nueva clase llamada `EnsembleFromNpLoaders`, que recibe como argumento únicamente una lista de cargadores de `PredictionsFromNumpy`. Dentro de la función de construcción del conjunto de predicciones, se carga cada uno de los conjuntos de datos y luego se concatenan, añadiendo una dimensión adicional llamada `number`.

Capítulo 6

Configuración de experimentos

En este capítulo se describen los métodos y herramientas utilizadas para diseñar y evaluar las pruebas realizadas en este trabajo. En primer lugar, se detallan las diferentes estrategias de aprendizaje por conjuntos aplicadas, así como las consideradas a lo largo del proyecto. A continuación, se indica el mecanismo de unificación empleado y se presentan las métricas, tanto deterministas como probabilísticas, utilizadas para evaluar los experimentos. Tras ello, se explican los ajustes aplicados durante el entrenamiento del modelo y el proceso para obtener el modelo final. Por último, se definen las configuraciones utilizadas para la realización de pruebas de aprendizaje por conjuntos.

6.1. Estrategias de aprendizaje por conjuntos

En el capítulo de Planificación 3 se señaló que la opción más plausible para este proyecto era añadir la diversidad durante la fase de inferencia. Esto implica que no se entrenan modelos individuales con distintas arquitecturas o conjuntos de datos, sino que se modifica la entrada de datos oceanográficos en esta etapa, repitiendo el proceso varias veces para llegar a una predicción final.

Este enfoque se enmarca dentro de la estrategia de Bagging. A continuación, se presentan algunos de los métodos considerados para su implementación, así como se explica cuáles han sido los finalmente aplicados.

6.1.1. Ruido Gaussiano

La forma más sencilla de implementar variación en los datos es mediante ruido aleatorio que sigue una distribución normal. Este tipo de ruido sigue la forma de la campana de Gauss y su uso es bastante frecuente por su aparición en la naturaleza (por ejemplo, en la distribución del peso o coeficiente intelectual de la población). La explicación más aceptada a este fenómeno es el Teorema del Límite Central, que determina que, bajo ciertas condiciones, la suma de variables aleatorias independientes e idénticamente distribuidas se aproxima a una distribución normal [Lyon, 2014].

La función de densidad de la distribución normal queda representada como:

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\, e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \tag{6.1}$$

donde μ es la media y σ^2 es la varianza.

La función en la ecuación 6.1 alcanza su valor máximo en la media, mientras que las frecuencias decrecen de forma simétrica desde este punto, con una amplitud determinada por la varianza. La función empleada (`random.normal`) [NumPy Developers, 2025] pertenece a la librería `numpy`, y requiere como argumentos la media y la desviación estándar de la distribución para generar el ruido.

Este tipo de ruido es utilizado en modelos climáticos que ya han sido probados y que se incluyen en comparativas de rendimiento realizadas por Google [Google Research, 2024]. Entre ellos se encuentran ArchesWeatherGen, NeuralGCM, FuXi [Chen et al., 2024] y FourCastNet [Pathak et al., 2022].

La Figura 6.1 muestra cómo la desviación estándar de la distribución de la que se extrae el ruido afecta a su intensidad, considerando que todas las distribuciones tienen media cero. Se observa que una desviación estándar de 1 es capaz de generar valores de ruido cercanos a 4 en las colas de la distribución, mientras que una desviación estándar de 0.5 produce ruidos de menor intensidad, reflejando una menor dispersión. En el caso de desviación estándar 0.1, las diferencias son poco apreciables debido a la escala compartida con las demás configuraciones. A pesar de estas variaciones en la desviación estándar, todos los mapas comparten un patrón espacial completamente aleatorio, con una apariencia dispersa o salpicada.

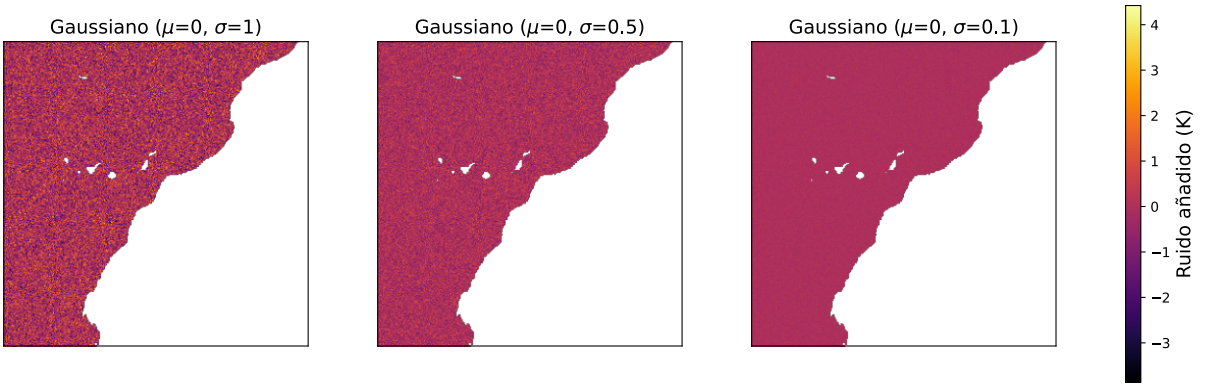


Figura 6.1: Ejemplos de ruido Gaussiano añadido a la SST, generado con diferentes desviaciones estándar en la distribución de muestreo.

6.1.2. Ruido de Perlin

El ruido de Perlin fue introducido en el artículo *An Image Synthesizer* por Ken Perlin. El objetivo era explicar *Pixel Stream Editor*, descrito como "un filtro que convierte imágenes de entrada en imágenes de salida mediante la ejecución de un mismo programa en cada píxel" [Perlin, 1985]. Su creación fue motivada por el duro trabajo que implicaba probar distintas funciones para crear texturas de apariencia natural.

Con esta innovación se logró crear texturas sólidas como nubes, fuego o agua mediante algoritmos computacionalmente ligeros que operan a nivel de píxel. Una de las funciones empleadas para generar estas texturas recibe el nombre de $Noise()$. Aunque la implementación se centra en imágenes de objetos tridimensionales, también ha sido adaptada para una dimensión y dos dimensiones. Esta función corresponde a lo que se conoce como Ruido Perlin, que destaca por su invarianza estadística bajo traslación y rotación y generar

características visibles contenidas dentro de un intervalo reducido de escalas (es decir, sin detalles muy grandes ni pequeños).

Para generar el ruido, primero se considera una rejilla tridimensional cuyos puntos tienen coordenadas enteras. A estos puntos, que actúan como vértices del cubo de la rejilla, se les asigna valores pseudoaleatorios y vectores gradiente mediante una función de resumen (*hash*) H, la cual produce cuatro valores reales independientes entre sí. Para un punto con coordenadas $[x, y, z]$, la siguiente ecuación representa esta asignación de valores:

$$[a, b, c, d] = H([x, y, z]), \tag{6.2}$$

donde $[a, b, c]$ representa el gradiente y d el valor de la función en el punto $[x, y, z]$.

La función de ruido se define como Noise($[x, y, z]$). Si el punto $[x, y, z]$ evaluado forma parte de la rejilla, entonces el valor de ruido coincide con d. Sin embargo, si el punto se encuentra dentro del volumen delimitado por los vértices de un cubo, se calcula el ruido mediante una interpolación suave (por ejemplo, un polinomio cúbico) utilizando los valores y gradientes de los vértices cercanos. Esta interpolación se realiza primero a lo largo del eje x (entre aristas), después a lo largo del eje y (sobre las caras del cubo) y finalmente a lo largo del eje z (entre planos). Esto permite generar ruido con un componente espacial [Perlin, 1985].

El ruido de Perlin destaca por su aplicación en Pangu-Weather para generar perturbaciones aleatorias [Bi et al., 2023a]. En este proyecto, se ha utilizado la misma librería de Python para generar ruido de Perlin que Pangu-Weather, llamada `perlin-numpy`. Este repositorio incluye un blog con explicaciones sobre las implementaciones de los algoritmos, además de implementar dos versiones distintas del ruido de Perlin.

Para la función de ruido en tres dimensiones, `generate_perlin_noise_3d`, se requieren como argumentos la forma del ruido a generar (`shape`), el número de rejillas de ruido a generar a lo largo de cada eje (`res`) y una opción booleana para definir si el ruido debería ser coherente en alguna de las dimensiones (`tileable`). Estos tres argumentos son tuplas, y en particular, la resolución determina el nivel de detalle del ruido [Vigier, 2018].

La Figura 6.2 muestra dos configuraciones distintas de ruido de Perlin que difieren en la resolución espacial aplicada. En el caso de mayor resolución espacial en los ejes de latitud y longitud ((2, 12, 12)), se observa una mayor cantidad de patrones de ruido, mientras que en la configuración de baja resolución ((2, 3, 3)), los patrones son más grandes y se aprecia con mayor claridad como la interpolación suaviza las transiciones dentro de cada celda.

Otra característica importante es que el ruido de mayor resolución presenta una intensidad ligeramente superior. A diferencia del caso del ruido Gaussiano, donde la desviación estándar controla la amplitud del ruido, en el ruido de Perlin esta intensidad no se debe a un parámetro directo. En su lugar, se debe a que una mayor resolución implica más celdas, y, por tanto, una mayor densidad de vectores gradientes en el espacio. Esto provoca transiciones más rápidas entre valores, aumentando la variabilidad local del ruido. No obstante, debido a la implementación usada del algoritmo, los valores de ruido de Perlin utilizados se acotan entre -1 y 1, independientemente de la resolución utilizada.

La principal diferencia entre el ruido Gaussiano y el ruido de Perlin es la presencia de una estructura espacial. Aunque ambos son aleatorios, el ruido de Perlin no asigna valores directamente a cada punto del espacio, sino únicamente a los vértices de determinadas regiones. A partir de estos, se calculan los valores intermedios mediante interpolación

suave, lo que permite transiciones continuas tanto dentro de cada celda como entre celdas vecinas.

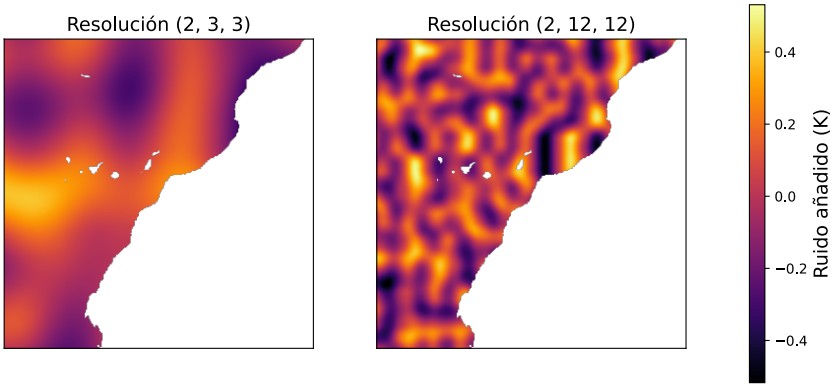


Figura 6.2: Ejemplos de ruido de Perlin añadido a la SST, generado con diferentes resoluciones espaciales.

6.1.3. Ruido de Perlin fractal

Existe otra función alternativa para generar ruido de Perlin, denominada como ruido de Perlin fractal (`generate_fractal_noise_3d`), que ofrece la librería `perlin-numpy`. Esta función se distingue por el empleo de tres parámetros adicionales. El primero son las octavas (`octaves`), que representan el número de iteraciones del ruido. Luego está la persistencia (`persistence`), que actúa como el factor de escalado de la amplitud del ruido entre octavas consecutivas. Por último, la lacunaridad (`lacunarity`) es el factor de incremento en la frecuencia entre dos octavas. Antes de profundizar en el funcionamiento y los parámetros, se define una ecuación que ayuda a entender el ruido:

$$F(x, y, z) = \sum_{i=0}^{O-1} a_i \cdot H(x, y, z; f_i), \tag{6.3}$$

donde O representa el número de octavas, a_i es la amplitud del ruido, y fi la frecuencia que modifica la resolución. La lacunaridad, la persistencia y la resolución evolucionan de la siguiente forma:

$$\text{resolución}_i = f_i \cdot \text{resolución}_0; \quad f_i = \text{lacunaridad} \cdot f_{i-1}; \quad a_i = \text{persistencia} \cdot a_{i-1}. \tag{6.4}$$

Es importante mencionar que tanto la amplitud como la frecuencia comienzan con un valor inicial de uno. En la ecuación (6.3), la amplitud controla la influencia de cada octava en el ruido total. Como se muestra en la ecuación (6.4), esta amplitud varía a lo largo de las iteraciones de ruido en función de la persistencia. En la biblioteca `perlin-numpy`, el valor por defecto de la persistencia es 0.5, lo que implica que las primeras octavas recibirán más peso en el total de ruido. Si la persistencia es mayor que uno, las octavas finales tendrán una mayor influencia en el resultado final.

Por otro lado, la frecuencia afecta a la resolución de ruido aplicada en cada octava, como se observa en la ecuación (6.4). La frecuencia es controlada por la lacunaridad, cuyo valor por defecto es 2. Una lacunaridad alta se traduce en un incremento de la frecuencia,

generando un mayor nivel de resolución y detalle en las últimas octavas. En cambio, una lacunaridad menor a uno provoca que la resolución disminuya a lo largo de las iteraciones, donde el ruido tendrá patrones más amplios y menos definidos.

Con los valores por defecto de la librería en cuanto a persistencia (0.5) y lacunaridad (2), las primeras octavas del ruido son las que más contribuyen al ruido final, ya que la amplitud disminuye progresivamente con el paso de las iteraciones debido a una persistencia menor que uno. Estas primeras octavas se caracterizan por tener una resolución de ruido más baja, generando patrones generales y de mayor escala. A medida que avanzan las octavas, la lacunaridad incrementa la frecuencia del ruido, aumentando consecuentemente la resolución. Las últimas octavas presentan detalles más finos de ruido, pero con un menor impacto en el resultado global. Por tanto, el propósito del ruido fractal es crear una base sólida de ruido en las primeras octavas, sobre la que se añaden progresivamente patrones de mayor resolución, aportando complejidad al ruido con un peso relativo menor [Vigier, 2018].

Una condición importante que no se ha tratado hasta ahora corresponde a las restricciones sobre la forma de la entrada de la función, las cuales vienen determinadas por la siguiente ecuación:

$$\text{shape} \bmod \left(\text{res} \cdot f_0 \cdot \text{lacunaridad}^{O-1}\right) = 0. \tag{6.5}$$

La ecuación anterior garantiza que la resolución del ruido aplicada en cada octava sea siempre un divisor exacto de las dimensiones de la forma del ruido deseada. En particular, basta con comprobar que la resolución final, obtenida tras multiplicar la resolución base por la frecuencia correspondiente a la última octava, sea múltiplo de las dimensiones del ruido. Dado que la frecuencia inicial (f_0) es igual a uno, la frecuencia final se convierte en la lacunaridad elevada al número de octavas menos uno. Para ilustrar esta restricción, se expone un problema que surgió durante el desarrollo de este proyecto.

El ruido de Perlin utilizado en Pangu-Weather es, precisamente, la variante fractal. Durante las primeras fases del Modelado, se optó por replicar la configuración empleada por los autores de Pangu-Weather, documentada tanto en su artículo como en su repositorio de GitHub [Bi et al., 2023b].

Los parámetros iniciales eran: tres octavas ($O = 3$), una resolución de (12, 12), persistencia de 0.5 y una lacunaridad igual a dos. La forma del ruido a generar en Pangu-Weather es de 721 x 1440. Al probar esta configuración, el código lanzó un error relacionado con la incompatibilidad de dimensiones, dado que la forma de los datos de este proyecto es de 300 x 300, obteniendo:

$$300 \bmod \left(12 \cdot 2^{3-1}\right) = 300 \bmod (48) \neq 0. \tag{6.6}$$

Por lo tanto, 300 no era múltiplo de la resolución aplicada en la última octava, a pesar de serlo para la resolución inicial (12). Esto obligó a explorar otras alternativas que sí cumplieran con esta condición, como por ejemplo la resolución 15, donde:

$$300 \bmod \left(15 \cdot 2^{3-1}\right) = 300 \bmod (60) = 0. \tag{6.7}$$

Esta condición también permite comprender cómo funciona la resolución del ruido. Por ejemplo, al usar una resolución de 15, la cuadrícula de 300 x 300 se divide en un total de 400 celdas más pequeñas de tamaño 15 x 15 (es decir, 20 celdas por eje). Cada una de estas subcuadrículas genera su propio patrón de ruido.

Por lo tanto, se observa que cuanto mayor sea la resolución, se generan más cuadrículas, lo que implica una mayor cantidad de patrones de ruido distintos. Por otro lado, una menor resolución divide en menos subcuadrículas la forma original.

A lo largo de este ejemplo se ha trabajado con datos en dos dimensiones. Sin embargo, es importante mencionar que las funciones de Perlin tridimensionales, presentadas hasta ahora, son simplemente una extensión directa y funcionan de forma idéntica. Por último, cabe mencionar que Pangu-Weather modifica levemente la aplicación del ruido de Perlin añadiendo un argumento adicional, de forma que:

$$F(x, y, z) = \alpha \cdot F(x, y, z), \tag{6.8}$$

donde α representa la escala del ruido, un parámetro que controla la magnitud del ruido en el resultado final de Perlin fractal. Su valor por defecto es 0.2.

La Figura A.5 muestra dos configuraciones distintas de ruido de Perlin fractal, que únicamente difieren por la resolución espacial aplicada. Para ambas se usan tres octavas en la iteración del ruido, con una persistencia de 0.5, lo que implica que la intensidad este disminuye progresivamente en cada iteración. La lacunaridad es de dos, lo que significa que, conforme avanzan las octavas, se emplean patrones cada vez más pequeños, resultando en ruido más detallado y complejo. Además, se aplica un coeficiente de escalabilidad de ruido (α) de 0.2 sobre el ruido resultante tras la última octava.

Se observa que el uso de un coeficiente de escalabilidad permite igualar las intensidades del ruido final entre ambas resoluciones, a pesar de la diferencia en la cantidad de patrones generados. En cuanto a los patrones, se aprecia que las resoluciones iniciales son las que aportan la mayor contribución al ruido final. La configuración con resolución (5, 5) presenta patrones más amplios en comparación con la resolución (15, 15). Sin embargo, a diferencia de las configuraciones de Perlin base en la Figura 6.2, los patrones de ruido fractal son algo más complejos y las transiciones entre ellos son más difuminadas. Esta diferencia se debe a que el refinamiento del ruido a lo largo de las octavas introduce detalles de menor intensidad de resoluciones superiores, agregando pequeñas variaciones que aumentan la complejidad del ruido fractal. Los ruidos de Perlin base no tienen esta característica, puesto que solo aplican una resolución inicial, de forma que los patrones son más nítidos.

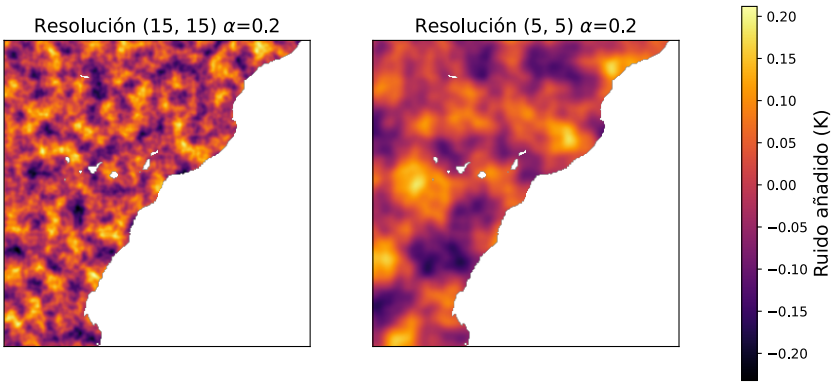


Figura 6.3: Ejemplos de ruido de Perlin Fractal con tres octavas añadido a la SST, generado con diferentes resoluciones espaciales y la misma escalabilidad de ruido. El resto de los parámetros corresponde a los valores por defecto.

Finalmente, se puede comprobar el efecto de los parámetros de lacunaridad y persis-

tencia, ya que efectivamente los ruidos que prevalecen son los añadidos con la resolución inicial. Mientras, los detalles finales agregados en las siguientes octavas tienen un menor peso en la composición final del ruido.

6.1.4. Otras estrategias consideradas

Aunque los métodos descritos anteriormente fueron los seleccionados para llevar a cabo las pruebas, durante la fase de estudio se analizaron otras estrategias que se describen a continuación.

Ruido simplex

Simplex [Gustavson, 2005] es un tipo de ruido desarrollado como una evolución del ruido de Perlin, diseñado también por Ken Perlin. Se caracteriza por solucionar algunas de las limitaciones detectadas en el ruido de Perlin, incluyendo la aparición de patrones repetitivos y el alto coste computacional. Entre sus principales ventajas está la capacidad de generar ruido en un mayor número de dimensiones con un coste computacional reducido, además de resolver el problema de trabajar con un espacio de coordenadas no entero. En este método, cada punto tiene un gradiente continuo en el espacio, evitando la necesidad de derivar ruido mediante operaciones de interpolación. A pesar de las mejoras presentadas, la mayoría de los proyectos, como Pangu-Weather, siguen prefiriendo el uso de ruido de Perlin, una decisión que se ha mantenido durante el desarrollo de este trabajo.

Predicciones rezagadas

El uso de predicciones rezagadas [Brenowitz et al., 2025] es una técnica inspirada en una metodología utilizada principalmente en la comparación de métodos de predicción por conjuntos. Surgió como respuesta a la dificultad de encontrar una forma justa de evaluar las capacidades de los modelos de aprendizaje por conjuntos basados en aprendizaje automático frente a los modelos deterministas. En bastantes casos, resulta complicado realizar una comparación debido a las configuraciones específicas utilizadas para crear el conjunto. Esta alternativa permite comparar los modelos sin recurrir a parámetros ni a operaciones de procesamiento costosas, ya que consiste en realizar predicciones sobre los modelos, pero con distintas fechas de inicialización. La fórmula es:

$$x_m(t, t') = x(t, t' - mh), \quad -M \leq m \leq M, \tag{6.9}$$

donde $x(t, t')$ representa el estado predicho válido para el tiempo t e inicializado en el tiempo t', cuyo horizonte de predicción es $t - t'$. Además, h representa el incremento fijo del intervalo de predicción y M es el número de miembros.

Con esta fórmula es posible generar un total de $2M + 1$ predicciones, donde el tiempo inicial de la predicción sería distinto, pero el tiempo de validez sería el mismo para todos. De esta forma, se introduce la diversidad mediante alteraciones temporales en el conjunto.

Aunque en ningún modelo es viable usar información de estados futuros para realizar predicciones, esta estrategia puede replicarse usando únicamente estados pasados. En ese caso, las predicciones rezagadas se detendrían antes que la original, igualando así la longitud temporal de cada miembro.

Sin embargo, por la forma de trabajar con los archivos de datos en este trabajo, resulta complicado crear distintos conjuntos con las fechas desplazadas, lo cual habría requerido una mayor modificación del código.

Técnica de difusión de GenCast

GenCast [Price et al., 2023] es un modelo de predicción climática probabilístico basado en aprendizaje automático, desarrollado por Google DeepMind, que ha superado en precisión a modelos numéricos de aprendizaje por conjuntos como ECMWF ENS. Aunque GenCast comparte gran parte de su arquitectura con GraphCast, presenta ciertas diferencias. Una de las diferencias se encuentra en la malla, donde en lugar de utilizar el enfoque multimalla de GraphCast, GenCast emplea una malla icosaédrica refinada seis veces, equivalente a la malla de mayor resolución de GraphCast.

GenCast se implementa como un modelo de difusión, una clase de modelos de aprendizaje profundo que generan muestras a partir de una distribución de datos aprendida. A diferencia de GraphCast, en GenCast no se usa una arquitectura de GNN basada en paso de mensajes, sino que en su lugar se trabaja en el módulo procesador basado en transformer. Este módulo procesa una esfera de ruido para generar el nuevo estado predicho. La esfera está compuesta de ruido Gaussiano generado en el espacio de las armónicas esféricas, que permite que el ruido sea más equilibrado dada la naturaleza esférica del planeta.

El módulo de eliminación de ruido (*denoiser*) se entrena aplicando ruido Gaussiano, controlado por σ (que corresponde a la desviación estándar), a un estado y aprende a reducir dicho ruido para reconstruir el estado limpio. Además, durante el proceso de eliminación de ruido, también recibe como contexto los dos estados iniciales. La capacidad de generar predicciones a partir de ruido puro permite repetir el proceso con distintas muestras de ruido a refinar. De esta forma, se puede generar un conjunto de resultados de modelos a nivel de inferencia, tal y como se realiza en este trabajo.

Además, durante la fase de evaluación, GenCast introduce incertidumbre adicional en los datos de entrada. Esto se logra añadiendo ruido Gaussiano a ciertas variables y utilizando el conjunto de datos ERA5 EDA (ERA5 Ensemble of Data Assimilations), que se compone de 10 miembros con variables atmosféricas que siguen trayectorias diferentes basadas en la incertidumbre de las condiciones iniciales. GenCast fue entrenado para realizar predicciones con un único paso autorregresivo. Sin embargo, también es capaz de realizar predicciones en un horizonte temporal más amplio alimentándose de sus propias salidas.

A pesar del potencial y la eficacia de este enfoque, no se consideró una opción viable implementarlo, puesto que sería necesario entrenar otro módulo para poder generar diversidad en los datos a partir de ruido puro. En este trabajo, al realizar las predicciones con otro modelo, no es posible reutilizar datos de entrenamiento. Además, resultaría contraproducente tratar de crear un conjunto de modelos en fase de inferencia con esta técnica, dado que el proceso de entrenamiento sería costoso y el objetivo de esta decisión de diseño es reducir la carga computacional.

Perturbaciones estocásticas

Otra opción para introducir diversidad en los datos durante la fase de inferencia es el uso de perturbaciones estocásticas. Este tipo de perturbaciones, orientadas a modelos numéricos, son empleadas por los modelos de conjunto del ECMWF IFS, los cuales utilizan una técnica conocida como Tendencias de Parametrización Perturbadas Estocásticamente (SPPT).

Esta técnica tiene en cuenta procesos físicos que ocurren a escalas menores, como la radiación o los procesos relacionados con las nubes. En cada estado, las tendencias (es

decir, los cambios de estos procesos en cada paso del tiempo) modifican las variables de entrada del modelo. Las perturbaciones no se calculan de manera determinista, sino que en su lugar se multiplican por un término de ruido para alterarlas. Este ruido se controla en función del tamaño de la tendencia, lo que permite mantener un equilibrio y evitar grandes valores.

Otra opción son las Perturbaciones Estocásticas Parametrizadas (SPP), en las que también se añade ruido aleatorio que varía en el espacio y en el tiempo con patrones específicos. En este caso, las perturbaciones se generan a partir de una distribución log-normal, de forma que los valores siempre sean positivos. Además, no se modifican directamente las variables atmosféricas, sino que se alteran las parametrizaciones a pequeña escala. Esta estrategia surgió como solución a limitaciones de SPPT, donde solo se varían las variables según la intensidad de las tendencias. En ese caso, existe el riesgo de que un cambio cercano a cero no afecte a la perturbación, impidiendo que haya variabilidad espacial y temporal. En cambio, SPP no aplica ningún componente multiplicativo, sino que usa la suma de los parámetros modificados para calcular la variable atmosférica, introduciendo la aleatoriedad en el modelo de manera más indirecta [Ollinaho et al., 2017].

Aunque el enfoque anteriormente descrito se aplica a variables atmosféricas, existen también opciones en modelos oceánicos como NEMO. En un estudio sobre las incertidumbres regionales en el Golfo de Vizcaya [Vervatis et al., 2025], se aplica SPP para perturbar parámetros físicos del modelo NEMO, mientras que SPPT se aplica sobre el componente biogeoquímico PISCESV2 y ciertas variables atmosféricas de ECMWF.

Las perturbaciones estocásticas podrían emplearse para introducir diversidad durante la inferencia. Sin embargo, esto implicaría trabajar con un conjunto de datos adicional basado en esta técnica o en la descarga de datos de parámetros de bajo nivel. La principal desventaja de este enfoque es la necesidad de realizar una nueva búsqueda y procesamiento de dichos conjuntos de datos, así como su integración en el modelo. Uno de los mayores retos fue la adaptación de la red a nuevos conjuntos de datos, por lo que trabajar con distintas fuentes desde archivos resulta complicado dentro del alcance de este trabajo. No obstante, esta posibilidad puede considerarse como una línea de investigación futura.

Otras estrategias

En este caso, se proponen estrategias que no están directamente relacionadas con modelos climáticos. Una de ellas consiste en aplicar los métodos de generación de ruido previamente descritos sobre los conjuntos de variables atmosféricas y pesos del modelo, en lugar de únicamente sobre los estados oceánicos. Como alternativa, una estrategia para fomentar la diversidad que no requiere de la modificación de datos es el uso de distintos puntos de inicialización del modelo durante las ejecuciones de inferencia. Esta técnica fue empleada en el modelo de lenguaje BERT [Devlin et al., 2019], donde cada ejecución se realiza con un conjunto de pesos obtenido en distintas etapas del entrenamiento del modelo.

6.2. Mecanismo de unificación

Los métodos de conjuntos requieren una técnica para unificar las salidas de todos los miembros, obteniendo así una única predicción, tal como se espera en los modelos oceanográficos. En este trabajo, se optó por implementar la opción más sencilla: el promedio de todos los miembros calculado mediante la siguiente fórmula:

$$\bar{f}_{t,l,i,j} = \frac{1}{M} \sum_{m}^{M} f_{t,l,i,j,m}, \tag{6.10}$$

donde M representa el número de miembros del conjunto, t la fecha de validez de la predicción, l el horizonte de predicción, i es el índice de latitud y j es el índice de longitud. La media del conjunto se calcula promediando las predicciones sobre la dimensión de los miembros, dejando fijas el resto de las dimensiones espaciales y temporales [Rasp et al., 2024].

6.3. Métricas de evaluación

Dada la naturaleza del proyecto, se usan dos conjuntos de métricas diferentes. Para evaluar la calidad de los distintos conjuntos de predicciones generados a partir de entradas contaminadas con distintos tipos de ruido, se utilizan métricas probabilísticas. Una vez unificados los resultados de los conjuntos, se aplican métricas deterministas clásicas para comparar las predicciones frente a una única serie generada por un modelo sin ruido en la entrada. Las métricas empleadas se extraen de la librería de `weatherbenchx`, la cual fue adaptada para poder cargar archivos `numpy` en lugar de `xarray`.

Cabe destacar que ninguna de estas métricas por sí sola resulta suficiente para concluir que conjunto es mejor. Es necesario realizar una evaluación basada en múltiples métricas y complementar el análisis con comparaciones más amplias [Rasp et al., 2024].

A lo largo de esta sección, se emplea una notación común para todas las métricas. La Tabla 6.3 presenta la correspondencia de símbolos utilizada, especificando el rango de cada índice y su significado.

Símbolo	**Rango**	**Descripción**
f	-	Predicción
o	-	Observación
t	$1, \ldots, T$	Tiempo de verificación
l	$1, \ldots, L$	Horizonte de predicción
i	$1, \ldots, I$	Índice de latitud
j	$1, \ldots, J$	Índice de longitud
m	$1, \ldots, M$	Índice del miembro del conjunto

Tabla 6.1: Descripción de los símbolos utilizados en las ecuaciones de la métricas. Fuente: [Rasp et al., 2024].

6.3.1. Métricas probabilísticas

En primer lugar, se definen las métricas probabilísticas empleadas en este trabajo, que incluyen el CRPS, el *spread-skill ratio* (con y sin sesgo) y el RMSE sin sesgo, extraídas de WeatherBench 2 [Rasp et al., 2024].

Continuous ranked probability score

El CRPS es una métrica compuesta por dos componentes, introducida por primera vez en [Gneiting et al., 2005]. A continuación, se describen ambos componentes por separado,

aplicando algunas simplificaciones [WeatherBenchX, 2025]. El primero es el CRPS skill, que evalúa la calidad de las predicciones con la siguiente fórmula:

$$\mathrm{CRPS}_{\mathrm{skill}} = \mathbb{E}|X - Y|, \tag{6.11}$$

donde X representa el resultado de un miembro del conjunto de predicciones e Y la observación real. El operador $\mathbb{E}$ denota el valor esperado (es decir, la media), mientras que $|\cdot|$ indica el valor absoluto. Cabe destacar que, cuanto menor sea este estadístico, mejor será la calidad de la predicción, puesto que es la media de las diferencias absolutas entre las predicciones y el valor real.

El segundo componente de la fórmula es la medida de dispersión, conocida como CRPS spread, cuya expresión es la siguiente:

$$\mathrm{CRPS}_{\mathrm{spread}} = \mathbb{E}[X - X'], \tag{6.12}$$

donde X' es el resultado de la predicción de otro miembro del mismo conjunto que X. Por lo tanto, esta medida es un indicador de cómo varían las predicciones entre ellas. A partir de estos dos componentes, se obtiene la siguiente fórmula de CRPS:

$$\mathrm{CRPS} = \mathbb{E}|X - Y| - \frac{1}{2}\mathbb{E}|X - X'|. \tag{6.13}$$

Como se observa el CRPS puede interpretarse como una medida del error esperado entre una distribución de probabilidad predicha y las observaciones reales, ajustada por la dispersión interna del conjunto de predicciones. Cuanto menor sea el CRPS, mejor será la capacidad predictiva del conjunto de modelos [Gneiting et al., 2005]. Por lo tanto, el CRPS busca un equilibrio entre predicciones cercanas a los valores observados y una dispersión controlada en el conjunto que permita captar la incertidumbre. Además, esta medida es igual al error medio absoluto (MAE) para el caso de una única predicción, ya que solo se mantiene el primer término de la ecuación.

No obstante, la fórmula anterior no es la empleada en este proyecto, dado que solo tiene en cuenta miembros muestreados del conjunto y no el conjunto completo. La expresión realmente utilizada es la siguiente:

$$\mathrm{CRPS}_l := \frac{1}{T}\sum_{t}^{T}\left(\frac{1}{M}\sum_{m=1}^{M}\left\|f^{(m)} - o\right\|_{t,l} - \frac{1}{2M(M-1)}\sum_{m=1}^{M}\sum_{\substack{n=1\\ n\neq m}}^{M}\left\|f^{(m)} - f^{(n)}\right\|_{t,l}\right). \tag{6.14}$$

La idea es similar a la ecuación (6.13), pero en este caso se realiza un promedio sobre las dimensiones de tiempo de verificación y de miembros del conjunto. Cabe destacar que tanto las predicciones como las observaciones reales ya han sido previamente promediadas en sus coordenadas espaciales. En el primer sumatorio interior se calcula el error medio cometido por los miembros del conjunto frente a las observaciones reales, para un horizonte de predicción l y tiempo de verificación t.

El componente de dispersión (segundo sumatorio) presenta una diferencia respecto a la formulación original, dado que no se usa el factor $\frac{1}{2}$, sino $\frac{1}{2M(M-1)}$. Esta fue una medida propuesta por [Zamo and Naveau, 2018] con el objetivo de obtener una estimación de CRPS no sesgada, también conocida como *fair* CRPS que se puede usar para conjuntos de más de dos miembros.

Al trabajar con un número limitado de predicciones generadas por un conjunto, se introduce un sesgo respecto a la distribución predictiva subyacente. Dicha distribución solo se podría evaluar con precisión si se contara con un conjunto infinito de miembros. Esta estimación del CRPS permite aproximarse a ese escenario [WeatherBenchX, 2025]. La solución se basa en calcular las diferencias entre todos los pares de miembros del conjunto y luego promediarlas. En este caso, se trabaja con pares repetidos (ya que, excepto cuando $n = m$, todas las demás combinaciones ocurren dos veces), por lo que se divide por el número de pares procesados ($M(M-1)$) multiplicado por dos, lo que ayuda a reducir el sesgo.

Finalmente, el sumatorio exterior promedia el CRPS para un horizonte temporal l específico a lo largo de los tiempos de verificación disponibles. En los modelos climáticos, es muy común calcular las métricas según el horizonte de predicción, en lugar de por días individuales, ya que permite evaluar cómo evoluciona la predicción a medio plazo.

Spread-skill ratio

Esta métrica evalúa la calibración del conjunto de predicciones. Se considera que un conjunto está bien calibrado cuando el error cometido se corresponde con la dispersión de sus predicciones respecto a la media. De hecho, el nombre de esta métrica podría traducirse como "relación entre dispersión y habilidad". La fórmula empleada utiliza la media de las predicciones del conjunto, que se puede describir con la ecuación (6.10). Una vez obtenida la media, es necesario definir la dispersión (no debe confundirse con él término de dispersión utilizado en el CRPS) del conjunto como:

$$\text{Spread}_l = \sqrt{\frac{1}{TIJ} \sum_{t}^{T} \sum_{i}^{I} \sum_{j}^{J} \text{var}_m\big(f_{t,l,i,j,m}\big)}, \tag{6.15}$$

donde $var_m(f_{t,l,i,j,m})$ corresponde a la varianza insesgada calculada sobre la dimensión del conjunto [WeatherBenchX, 2025], para cada punto espacial (i, j), tiempo de verificación t y horizonte de predicción l. Esta varianza se promedia sobre la latitud, longitud y tiempo de verificación. Finalmente, al aplicar la raíz cuadrada al promedio de varianzas, se obtiene la desviación estándar que representa la dispersión para un horizonte de predicción l específico, expresada en las unidades de la predicción. En la ecuación original de WeatherBench, se incluye una ponderación por tamaño de celda $w(i)$ (como la calculada en la ecuación (4.1)). Sin embargo, no se observaron diferencias significativas al usar esta ponderación, por lo que se decidió omitirla para ahorrar cómputo, ya que se calcula internamente durante la evaluación en las distintas llamadas. Esta misma decisión se aplicó para el cálculo del RMSE. La fórmula de la varianza empleada es:

$$\text{var}_m\big(f_{t,l,i,j,m}\big) = \frac{1}{M-1} \sum_{m}^{M} \left(f_{t,l,i,j,m} - \bar{f}_{t,l,i,j}\right)^2, \tag{6.16}$$

donde se observa que la varianza representa la dispersión de las predicciones individuales del conjunto respecto a su media. Esto supone una diferencia respecto al término de dispersión empleado en el CRPS, donde se calcula la diferencia entre pares de miembros del conjunto.

Finalmente, se define el *spread-skill ratio* como:

$$R_l = \frac{\text{Spread}_l}{\text{RMSE}_l(\bar{f})} \tag{6.17}$$

Como se observa en la ecuación anterior, el *spread-skill ratio* divide la dispersión por la raíz del error cuadrático medio cometido por la media del conjunto. Si el cociente es mayor que uno, indica que el modelo tiene mayor dispersión que error, lo que hace que el modelo sobreestime la incertidumbre, pudiendo compensar errores.

Si el valor es menor que uno, significa que el conjunto presenta una dispersión insuficiente en relación con el error. En este caso, si el RMSE es alto, las predicciones son demasiado similares respecto a la media, por lo que el modelo no refleja adecuadamente la incertidumbre ni representa bien los errores cometidos por la predicción promedio.

El mejor resultado es un valor cercano a uno, lo que indica una buena calibración del conjunto, ya que la dispersión representa adecuadamente los errores cometidos por la predicción.

Se puede distinguir que el CRPS y *spread-skill ratio* no evalúan los mismos aspectos del conjunto. El CRPS es una métrica que evalúa cuánto se aproxima la distribución de probabilidad predicha a las observaciones reales. Aunque incluye un término de dispersión, no mide directamente la calibración del conjunto, sino que busca que el conjunto refleje una incertidumbre adecuada, al mismo tiempo que logre un buen rendimiento predictivo.

El *spread-skill ratio* es una medida de calibración que se basa en la intercambiabilidad entre los miembros del conjunto y las observaciones reales. Bajo esta suposición, según [Wilks, 2011], "la distribución conjunta de los miembros del conjunto no cambiaría si se reemplaza cualquiera de ellos por la observación". Por tanto, comparar la dispersión del conjunto frente al RMSE de su media permite evaluar si su incertidumbre está bien representada. De esta forma, un valor cercano a uno indica que los miembros del conjunto son, en promedio, tan representativos como la observación misma [Fortin et al., 2014].

Es importante mencionar que, en este proyecto, el *spread-skill ratio* se calcula utilizando una estimación corregida del RMSE para reducir el sesgo (6.20), que se explica más en detalle en el siguiente apartado.

Raíz del error cuadrático medio sin sesgo

Como se muestra en la ecuación (6.17), el RMSE aparece en el denominador de la definición del *spread-skill ratio*. La expresión del RMSE para la métrica base es la siguiente:

$$\text{RMSE}_l = \sqrt{\frac{1}{TIJ}\sum_{t}^{T}\sum_{i}^{I}\sum_{j}^{J}\left(\bar{f}_{t,l,i,j} - o_{t,i,j}\right)^2}. \tag{6.18}$$

Esta métrica se calcula como el promedio del error cuadrático entre la predicción, representada por la media del conjunto (ecuación (6.10)), y la observación. Al igual que las demás métricas, el RMSE se evalúa para cada horizonte temporal (l), promediando los errores por las dimensiones de tiempo de verificación (T), latitud (I) y longitud (J).

También se utiliza una variante del RMSE que incorpora al cálculo la varianza definida en la ecuación (6.16), con el fin de convertirlo en una medida corregida por sesgo. Para ello primero se calcula el promedio de la varianza a lo largo de las dimensiones de tiempo de verificación, latitud y longitud, tal y como se define en la ecuación:

$$\text{var}_l = \frac{1}{TIJ} \sum_{t}^{T} \sum_{i}^{I} \sum_{j}^{J} \text{var}_m\big(f_{t,l,i,j,m}\big), \tag{6.19}$$

donde var_l representa el promedio de la varianza en el horizonte de predicción l. Una vez obtenido este término, el RMSE sin sesgo se expresa como:

$$\text{UnbRMSE}_l = \sqrt{\text{MSE}_l - \text{var}_l}, \tag{6.20}$$

donde el RMSE sin sesgo para cada horizonte de predicción se calcula restando la varianza promedio (var_l) al error cuadrático medio (MSE_l). En este caso, se utiliza el error cuadrático medio en lugar del RMSE para poder aplicar esta corrección, y luego se toma la raíz cuadrada del resultado [WeatherBenchX, 2025]. Esta versión corregida del RMSE es la que se aplica para calcular el *spread-skill ratio* sin sesgo, sustituyendo a RMSE_l en la ecuación (6.17).

6.3.2. Métricas deterministas

En este apartado se explican las métricas deterministas empleadas en este trabajo para evaluar la media del conjunto frente a las observaciones, así como el rendimiento de los modelos individuales. Estas métricas son el RMSE y el sesgo (*bias*), extraídas también de [Rasp et al., 2024].

Raíz del error cuadrático medio

La fórmula del RMSE se define como:

$$\text{RMSE}_l = \sqrt{\frac{1}{TIJ} \sum_{t}^{T} \sum_{i}^{I} \sum_{j}^{J} \left(f_{t,l,i,j} - o_{t,i,j}\right)^2}, \tag{6.21}$$

donde se mantiene la misma notación empleada en las métricas probabilísticas. Además, se puede observar que esta fórmula es prácticamente idéntica a la de la ecuación (6.18), pero usando una única predicción ($f_{t,l,i,j}$) en lugar de la media de los miembros del conjunto ($\bar{f}_{t,l,i,j}$).

Sesgo

La fórmula del sesgo se define como:

$$\text{Bias}_l = \frac{1}{TIJ} \sum_{t}^{T} \sum_{i}^{I} \sum_{j}^{J} \left(f_{t,l,i,j} - o_{t,i,j}\right), \tag{6.22}$$

donde se evalúa el sesgo por cada horizonte de predicción, calculando el promedio de las diferencias entre la predicción realizada por el modelo para un tiempo y ubicación geográfica específicos respecto al valor observado. Esta métrica permite saber cómo se suele comportar el modelo en sus predicciones en relación a las observaciones. En primer lugar, si el sesgo es positivo, significa que el modelo tiende a predecir valores mayores que los observados, provocando sobreestimación. Un sesgo es cercano a cero, sugiere que las predicciones del modelo, en promedio, se ajustan bien a las observaciones. Por último, un sesgo negativo indica que el modelo suele subestimar los valores observados.

6.4. Estrategia de entrenamiento

El entrenamiento en este proyecto es muy relevante, dado que se parte de un único modelo sobre el cual se realiza inferencia para llevar a cabo las pruebas. Esta idea justifica el uso de un período relativamente largo de datos para el entrenamiento. Sin embargo, en cuanto al desarrollo de la estrategia de entrenamiento, se comenzó utilizando la configuración predeterminada empleada en el trabajo original de SeaCast.

El primer entrenamiento se definió a 100 épocas con una tasa de aprendizaje inicial de 0.00001 y el optimizador AdamW. Además, se aplicaron cinco épocas de calentamiento con una estrategia de decaimiento cosenoidal para ajustar progresivamente la tasa de aprendizaje.

Fase de calentamiento

La fase de calentamiento (*warmup*) se emplea para evitar problemas de optimización que suelen producirse en las primeras épocas del entrenamiento, debido a los grandes cambios que experimenta la red en poco tiempo. En esta estrategia, la tasa de aprendizaje se incrementa de forma constante desde un valor inicial bajo hasta alcanzar un valor objetivo. El crecimiento es constante durante cada época, evitando grandes variaciones y mejorando la estabilidad del entrenamiento [Goyal et al., 2017]. En este caso, la tasa de aprendizaje comienza en 0.00001 y se eleva hasta 0.001.

Evolución de tasa de aprendizaje

Una vez finalizada la fase de calentamiento, la tasa de aprendizaje evoluciona siguiendo un decaimiento cosenoidal (*cosine decay schedule*) [Loshchilov and Hutter, 2016]. Esto permite mejorar el rendimiento en comparación con los métodos tradicionales, que reducen la tasa de aprendizaje dividiendo su valor por un factor cada ciertas épocas. Esta estrategia ayuda a reducir el sobreajuste, ya que los cambios en la tasa no ocurren de forma abrupta, sino siguiendo una curva suave. La principal ventaja de reducir la tasa de aprendizaje progresivamente es que en las primeras épocas la red aprende a generalizar los datos, mientras que los cambios en las últimas épocas no reciben tanta importancia, permitiendo ajustar pequeños detalles. La ecuación para la evolución de la tasa de aprendizaje es:

$$\eta_t = \eta_{\text{mín}} + \frac{1}{2}(\eta_{\text{máx}} - \eta_{\text{mín}}) \left(1 + \cos\left(\frac{t\pi}{T}\right)\right), \tag{6.23}$$

donde $\eta_{\text{mín}}$ y $\eta_{\text{máx}}$ representan los valores mínimo y máximo de la tasa de aprendizaje, t es la época en ejecución y T es el número total de épocas del entrenamiento. Cuando el número de épocas en ejecución es igual al número total, el término $cos(\pi)$ equivale a -1, lo que anula el segundo término y establece la tasa de aprendizaje mínima ($\eta_{\text{mín}}$). En este caso, no se especifica un valor mínimo explícito para la tasa de aprendizaje, por lo que podría llegar a ser 0. Además, la época actual (t) puede tomar valores decimales, lo que permite modificar la tasa durante los pasos intermedios del modelo.

Actualización de pesos de la red

Para actualizar los pesos de la red se utiliza AdamW [Loshchilov and Hutter, 2017]. Este optimizador realiza las actualizaciones a partir del gradiente de la función de pérdida, teniendo en cuenta la media y varianza móviles de los gradientes calculados previamente.

AdamW soluciona una limitación común de la regularización L2 tradicional, que tiende a aplicar una penalización baja en caso de pesos con grandes gradientes, debido a que la penalización se incorpora directamente en el cálculo del gradiente. Para solucionarlo, AdamW aplica un paso separado de decaimiento de pesos, controlado por un parámetro λ, que asegura que en cada actualización estos pesos se reduzcan de forma adecuada. Este optimizador ha mostrado una buena capacidad de generalización, así como también se complementa con métodos de tasa de aprendizaje variable, como el empleado en este entrenamiento.

Los parámetros empleados son los mismos que en SeaCast: $\beta_1 = 0{,}9$, $\beta_2 = 0{,}95$ y $\lambda = 0{,}1$. Los parámetros β_1 y β_2 modifican la influencia de las magnitudes históricas de los gradientes, específicamente de su media y varianza móvil. La función de pérdida utilizada en este caso corresponde al RMSE ponderado definido en la ecuación (5.9).

Configuración del modelo y detalles de entrenamiento

En cuanto a los parámetros específicos de la arquitectura de la red, se utilizó una capa oculta de 128 neuronas, lo que da lugar a representaciones vectoriales de dimensión 128 dentro de las GNNs y sus MLPs. El procesador del modelo está compuesto por cuatro capas. Respecto a la malla, se usaron tres niveles uniformes con resoluciones de 81, 27 y 9, respectivamente. El nivel de mayor resolución (malla inferior) incluye un total de 81 x 81 nodos, entre ellos aquellos ubicados sobre tierra firme. No obstante, se eliminaron los nodos y aristas que atraviesan superficies terrestres a lo largo de cierta distancia, lo que da lugar a la estructura descrita en la Tabla 6.2.

Grafo	**Nodos**	**Conexiones**
$\mathcal{G}_{M[1]}$	3,573	27,616
$\mathcal{E}^{[1\nearrow 2]}$ / $\mathcal{E}^{[2\searrow 1]}$	-	7,146
$\mathcal{G}_{M[2]}$	398	2,886
$\mathcal{E}^{[2\nearrow 3]}$ / $\mathcal{E}^{[3\searrow 2]}$	-	796
$\mathcal{G}_{M[3]}$	45	270
Malla total	4016	38,714
$\mathcal{G}_{G2M}$	-	49,061
$\mathcal{G}_{M2G}$	-	49,061
Rejilla latitud-longitud	49,061	-

Tabla 6.2: Número de nodos y aristas del grafo utilizado.

En cuanto a los pasos autorregresivos, se definió que el incremento comenzara a partir del 80 % del entrenamiento, es decir, a partir de la época 80 (de un total de 100). El máximo de pasos autorregresivos fue fijado en 4.

Con esta configuración inicial, el modelo contaba con cerca de 5.6 millones de parámetros entrenables y un tamaño total de 22.395 GB, lo que limitaba la posibilidad de dividir los datos en batches para su procesamiento en paralelo. Por esta razón, los datos diarios se procesaron de forma secuencial, utilizando un tamaño de batch igual a uno. El tamaño en memoria del modelo hacía que entrenar este modelo en un ordenador portátil provocara un error por saturación de memoria. Por lo tanto, se utilizó el equipo del laboratorio, que contaba con 32 GB de memoria RAM.

Este entrenamiento sufrió una interrupción debido al reinicio del ordenador del laboratorio donde se estaba ejecutando. Se detuvo en la época 96, lo cual fue suficiente para

extraer una serie de conclusiones.

La Figura 6.4 muestra que el modelo llega a alcanzar una pérdida cercana a 0.5 antes de la época 80. Sin embargo, al incrementar los pasos autorregresivos a dos, se observa un aumento repentino del RMSE, que alcanza un valor 1.2. Cabe destacar que la gráfica está parcialmente recortada, ya que en la época 85 los pasos autorregresivos se incrementaron nuevamente en uno, provocando otro aumento en la pérdida que habría dificultado la visualización del valor mínimo alcanzado. Es normal que el error del modelo incremente al utilizar sus propias predicciones, ya que los dos estados iniciales dejan de ser observaciones reales. No obstante, aunque el error sufra un crecimiento temporal, esta técnica puede ser beneficiosa para mejorar las predicciones del modelo a medio plazo, ya que se observa que el error desciende tras dicho incremento.

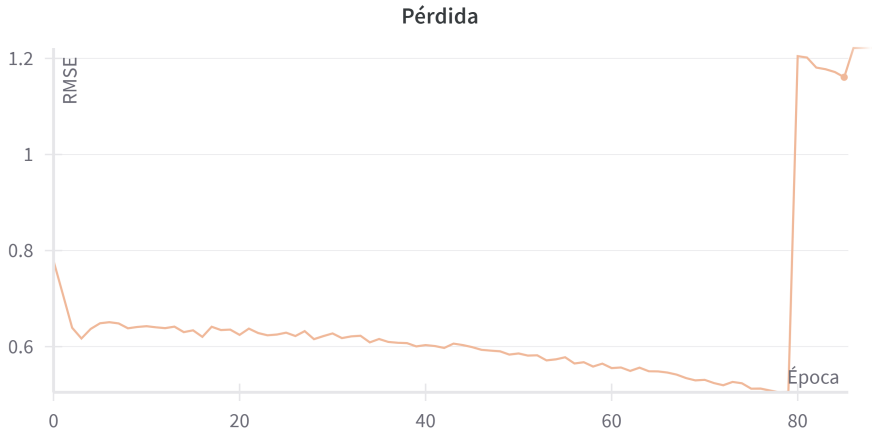


Figura 6.4: Pérdida RMSE en los datos de entrenamiento por época durante el primer entrenamiento.

La Figura 6.5 muestra el tiempo consumido en horas durante el entrenamiento, donde se observa que casi la mitad corresponde a la fase autorregresiva. Además, el entrenamiento estuvo en ejecución durante 6 días y 8 horas antes de detenerse.

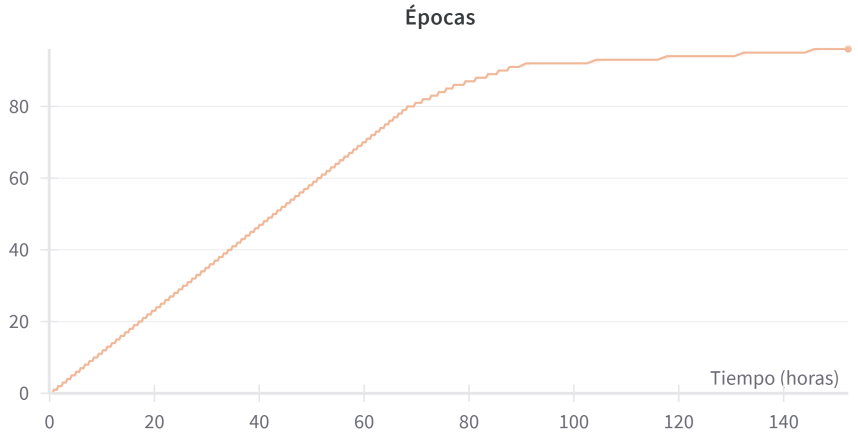


Figura 6.5: Tiempo consumido en horas durante el primer entrenamiento.

Debido al largo tiempo de ejecución de la fase autorregresiva, se optó por descartar su uso en los siguientes entrenamientos. Por otro lado, la inferencia sobre los datos de prueba

con los pesos de este modelo requería cerca de 12 horas en el equipo portátil personal, lo que dificultaba la realización de múltiples pruebas para los conjuntos de predicciones.

El entrenamiento final mantiene los mismos parámetros que el anterior, con la excepción de los pasos autorregresivos que se redujeron a uno. Además, se detectó que, en el código de generación de la malla de SeaCast, se estaban utilizando 16 vecinos para establecer las conexiones entre los niveles de la jerarquía. Esta configuración podría haber sobrecargado la estructura del modelo, lo que explicaría los largos tiempos de ejecución. En este nuevo entrenamiento, el número de vecinos se redujo a uno, obteniendo la estructura definida en la Tabla 6.2. Este cambio, sin embargo, no provocó una reducción en la memoria ocupada por el modelo, ya que no modifica el número de parámetros entrenables. Por último, el número total de épocas se aumentó a 150.

Este entrenamiento tuvo una duración total de 4 días y 5 horas sin interrupciones, resultando considerablemente más rápido que el anterior, a pesar de contar con más épocas, debido a la ausencia de pasos autorregresivos. La Figura 6.6 muestra que la pérdida en el conjunto de entrenamiento alcanzó un valor de 0.46 en la última época, donde parece cercano a converger.

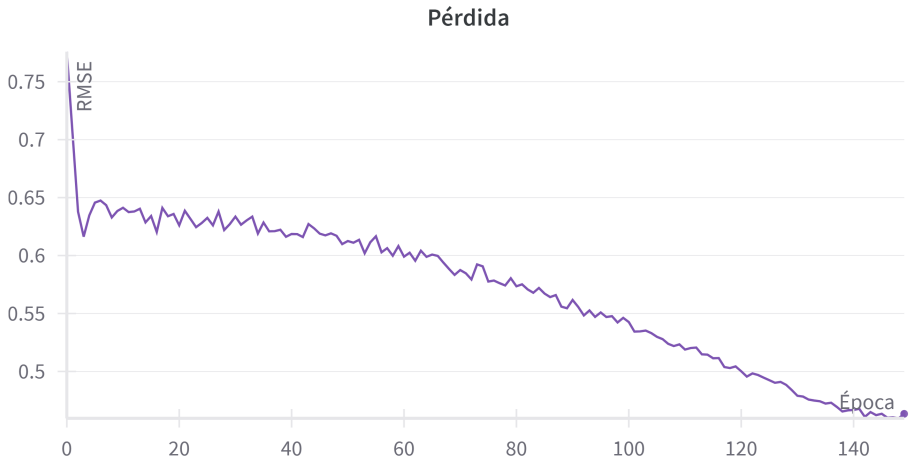


Figura 6.6: Pérdida RMSE en datos de entrenamiento por época en el entrenamiento final.

La Figura 6.7 muestra la pérdida promedio en el conjunto de validación, compuesto por datos no vistos por el modelo durante el entrenamiento. Se observa que el rendimiento en validación es inferior al modelo anterior, lo cual era esperado, ya que durante esta fase se realizan predicciones con hasta cuatro pasos autorregresivos. Esta pérdida promedio resume el rendimiento de los modelos frente a distintos números de pasos autorregresivos. A pesar de que el modelo final es peor bajo este criterio, la diferencia no es significativa. El modelo anterior alcanzó una pérdida de 2.33, mientras que el modelo final obtuvo 2.37.

Este último modelo realiza todas las predicciones sobre el conjunto de pruebas de dos años en aproximadamente dos horas y media. Su rendimiento, comparable al de un modelo entrenado durante más tiempo, junto con la rapidez en la fase de inferencia, motivaron su elección como modelo final.

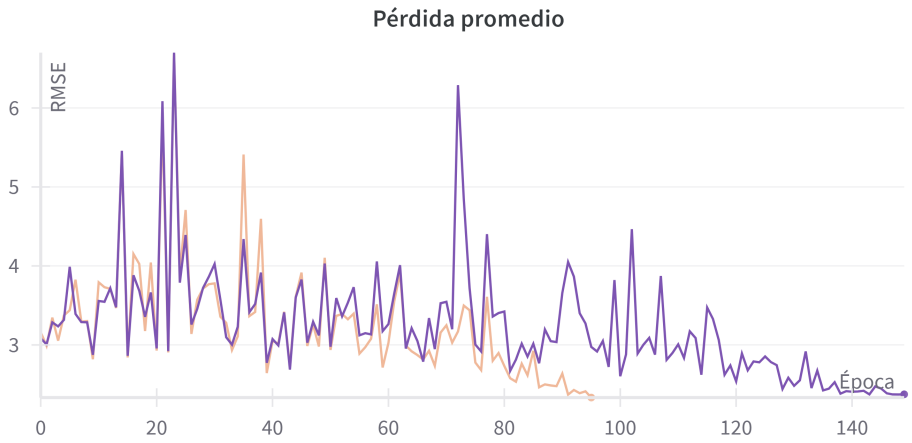


Figura 6.7: Pérdida promedio en datos de validación por época en los entrenamientos.

6.5. Estrategia de pruebas con conjuntos de predicciones

Para diseñar las pruebas de conjuntos de predicciones, se optó por realizar inicialmente la inferencia utilizando un único día como entrada del modelo en la fase de evaluación. Esta estrategia permite obtener 15 predicciones a futuro desde esta muestra, lo cual facilita una evaluación rápida de los distintos tipos de ruido de forma exploratoria. Durante esta fase, el objetivo no es obtener un resultado definitivo, sino descartar tipos de ruido que tengan un bajo rendimiento según las métricas. Se consideran tanto las métricas probabilísticas, para evaluar el comportamiento del conjunto y sus miembros individuales, como las métricas deterministas sobre la combinación de las predicciones del conjunto. Los ruidos que superaron esta etapa preliminar se probaron más tarde con el conjunto de evaluación completo, aunque se conservaron como referencia algunos ruidos cuyo rendimiento no fue tan bueno. El criterio de selección de estos ruidos consistió en quedarse con los más destacados según el conjunto de métricas.

Por un lado, se evaluaron las predicciones con ruido Gaussiano, y por otro, las predicciones que emplearon ruido de Perlin junto con Perlin fractal. Como se ha comentado a lo largo del documento, la estrategia de conjuntos utilizada fue Bagging, perturbando los estados oceánicos iniciales del modelo con ruido. Cada conjunto estuvo compuesto por cinco miembros, aunque se hicieron experimentos de diez miembros con los ruidos más prometedores. En cuanto a Perlin, en las funciones bidimensionales los ruidos son generados para una cuadrícula de 300 x 300 (latitud, longitud), iterando para los dos estados iniciales de cada archivo. Por otro lado, las funciones tridimensionales crean ruido con la forma 2 x 300 x 300, incluyendo la primera dimensión del número de días.

Las pruebas de ruido Gaussiano resultaron más sencillas de parametrizar, dado que su función generadora solo requiere dos parámetros: la media y la desviación estándar de la distribución. La Tabla 6.3 muestra las configuraciones de ruido Gaussiano empleadas en la primera fase de pruebas. No se consideraron valores de desviación estándar mayores a uno, ya que durante el desarrollo de los módulos de ruido se comprobó que afectan negativamente a las predicciones. En este caso, la función opera sobre los datos sin estructurar por latitud y longitud, por lo que el ruido se genera con forma 2 x 49,061, donde dos corresponde al número de días y 49,061 al total de puntos oceánicos.

Respecto a los ruidos de Perlin, primero se describen las pruebas realizadas con la

Nombre del ruido	**Media (μ)**	**Desviación estándar (σ)**
Gaussiano (μ=0, σ=0.1)	0.0	0.1
Gaussiano (μ=0, σ=0.05)	0.0	0.05
Gaussiano (μ=0, σ=0.01)	0.0	0.01

Tabla 6.3: Configuración de los ruidos Gaussianos empleados en la fase de pruebas.

versión clásica, aunque la mayoría de las configuraciones de ruido corresponden a la variante fractal, ya que era la opción empleada por Pangu-Weather. La Tabla 6.4 muestra dos configuraciones que varían en la resolución aplicada a los ejes de latitud y longitud. P_res_2x3x3 presenta un menor nivel de detalle, puesto que los patrones de ruido son mayores en el espacio. En cambio, P_res_2x12x12 usa una mayor resolución en los ejes geográficos, generando más patrones y, por lo tanto, ruido más complejo. Ambas configuraciones indican que el ruido debe ser coherente en el eje temporal (mediante el parámetro `tileable`), lo que garantiza una transición suave entre los dos estados ruidosos.

Nombre del ruido	**Resolución**	**Repetible**
P_res_2x3x3	(2, 3, 3)	(T, F, F)
P_res_2x12x12	(2, 12, 12)	(T, F, F)

Tabla 6.4: Configuración de los ruidos de Perlin empleados en la fase de pruebas.

Respecto a las configuraciones de ruido de Perlin Fractal, se siguieron los ajustes encontrados en el artículo de Pangu-Weather, aunque con algunas adaptaciones. En primer lugar, se evaluó si existía alguna diferencia al aplicar Perlin Fractal como si fuera la función de ruido de Perlin base. Esto se realizó con la configuración PF_res_1x3x3, donde la única modificación fue la resolución en el eje temporal, utilizando una única octava, lo que genera una iteración de ruido. Sin embargo, esta prueba confirmó que una resolución de 1 en la primera dimensión, junto con la opción de hacer el ruido coherente en ese eje, impiden generar variabilidad a lo largo del tiempo. Por lo tanto, el experimento de PF_res_1x3x3 no es de gran ayuda, puesto que se aplica el mismo ruido sobre los dos estados iniciales, aunque varía entre los miembros del conjunto. Por esta razón, no se considera en los resultados de este proyecto.

En cuanto al resto de pruebas, todas se basaron en los parámetros propuestos por Pangu-Weather, ajustando la resolución del ruido para adaptarse al tamaño distinto de los datos, tal como se explicó en la sección 6.1. En este caso, la función empleada genera ruido en dos dimensiones y las pruebas giran en torno a la configuración base PF_res_15x15, que emplea una resolución de 15 en cada eje espacial. Además, se especificó que el ruido no sufra cambios bruscos en el eje de la longitud. Se utilizó una persistencia de 0.5 (factor que regula la escala de la amplitud del ruido), y un total de tres octavas para aumentar la complejidad espacial.

Aunque en las especificaciones de Pangu-Weather no se señala la lacunaridad explícitamente, al analizar la evolución de las resoluciones utilizadas (12, 24, 48), se puede concluir que su valor es de 2 [Bi et al., 2023a]. Además, como se mencionó anteriormente en la sección 6.1, en el repositorio de GitHub de Pangu-Weather se incorpora un argumento adicional que no pertenece al algoritmo original. Este parámetro pondera el ruido final de Perlin fractal con un coeficiente de escalabilidad de ruido (α), cuyo valor por defecto es de 0.2 [Bi et al., 2023b].

El resto de las configuraciones se derivan de esta, modificando un parámetro en cada caso. PF_res_5x5 reduce la resolución inicial para ver en qué medida afecta. La configuración PF_res_15x15_without_tileable hace que el ruido varíe bruscamente en cada generación, sin seguir ningún patrón por longitud. En cuanto a PF_res_15x15 (α=0.05), modifica el parámetro de escalabilidad del ruido, reduciéndolo a 0.05 con el objetivo de generar una menor cantidad de ruido final. Por el contrario, la configuración PF_res_15x15 (α=0.4) aumenta la escalabilidad de este ruido, de forma que la magnitud del ruido sea similar a la de los ruidos de Perlin (ver apéndice A).

La lacunaridad y persistencia no se modifican, ya que son los valores utilizados son los recomendados tanto en el paquete de las funciones [Vigier, 2018] como en Pangu-Weather [Bi et al., 2023b]. Además, modificar la lacunaridad requiere de un análisis cuidadoso, ya que implica calcular cómo evolucionaría la resolución a lo largo de las octavas, asegurando que sea compatible con la forma del ruido.

ID	**Resolución**	**Repetible**	**Persistencia**	**Octavas**	**Escala**	**Lacunaridad**
A	(1, 3, 3)	(T, F, F)	0.5	1	1.0	2.0
B	(15, 15)	(F, T)	0.5	3	0.2	2.0
C	(5, 5)	(F, T)	0.5	3	0.2	2.0
D	(15, 15)	(F, F)	0.5	3	0.2	2.0
E	(15, 15)	(F, T)	0.5	3	0.05	2.0
F	(15, 15)	(F, T)	0.5	3	0.4	2.0

Tabla 6.5: Configuración de los ruidos de Perlin fractal empleados en la fase de pruebas.

Los identificadores corresponden a:
A: PF_res_1x3x3, **B**: PF_res_15x15, **C**: PF_res_5x5, **D**: PF_res_15x15_without_tileable, **E**: PF_res_15x15 (α=0.05), **F**: PF_res_15x15 (α=0.4).

Capítulo 7

Resultados

En este capítulo se presentan los resultados de las pruebas descritas en la sección 6.5. A diferencia del flujo de pruebas mencionado anteriormente, se comienza evaluando el rendimiento de los conjuntos de predicciones frente a una predicción determinista. A continuación, se discuten las diferencias observadas en las pruebas preliminares respecto a los distintos tipos de ruido, analizando tanto el ruido Gaussiano como el ruido de Perlin y destacando cómo los parámetros específicos de cada tipo de ruido afectan al rendimiento de los conjuntos de predicciones con estados iniciales perturbados. Finalmente, se analiza el impacto del tamaño del conjunto en su rendimiento, empleando un conjunto de pruebas más reducido. A lo largo de este capítulo se emplea la notación de experimentos creada en las Tablas 6.3, 6.4 y 6.5, por ello se recomienda consultarlas como referencia.

7.1. Comparación de conjuntos de predicciones y predicción determinista

Durante la última fase de pruebas, se realizaron cinco predicciones para conformar cada conjunto, considerando distintos tipos de ruido para alterar las condiciones iniciales. Las configuraciones de ruido evaluadas con el conjunto completo de datos de prueba fueron: gaussiano (μ=0, σ=0.01), Gaussiano (μ=0, σ=0.1), PF_res_15x15 (α=0.05), P_res_2x12x12 y P_res_2x3x3. A continuación, se analiza el rendimiento de la combinación de predicciones para cada configuración en comparación con una única predicción realizada por el modelo base sin perturbación en los estados iniciales.

La Tabla 7.1 muestra un resumen del empeoramiento porcentual en el rendimiento obtenido al combinar los conjuntos de predicciones correspondientes a las distintas configuraciones. Esta mejora se calcula en relación a la predicción determinista sin ruido, la cual se incluye como referencia. La métrica evaluada es el RMSE (ecuación (6.21)), promediado en tres horizontes temporales distintos (1, 5 y 15 días).

Como se observa, ninguno de los conjuntos unificados logra superar a la predicción determinista en el RMSE en los tres horizontes temporales, siendo la configuración Gaussiano (μ=0, σ=0.01) la única que iguala su desempeño en el día quince de predicción. Este resultado era esperable, dado que el modelo fue entrenado de forma determinista con estados iniciales libres de ruido. Por lo tanto, la predicción que no altera los datos de entrada tiende a aproximarse más a los valores esperados. Sin embargo, aunque no se mejora el rendimiento en términos de RMSE respecto a la predicción determinista, se observan algunos fenómenos interesantes.

En primer lugar, se observa que el RMSE empeora a medida que avanza el horizonte de predicción. Este comportamiento también era esperable, ya que el modelo se alimenta exclusivamente de sus propias predicciones a partir del tercer día. Esta tendencia se presenta tanto en los conjuntos de predicciones como en la predicción determinista. Por otro lado, los conjuntos de predicciones presentan un mayor deterioro del RMSE respecto a la predicción determinista en el primer día de predicción. Esto se debe a que, aunque utilicen observaciones reales como entrada, estas han sido contaminadas con ruido, lo que provoca un error inicial más alto en comparación con el modelo determinista.

En las configuraciones P_res_2x12x12, P_res_2x3x3 y Gaussiano (μ=0, σ=0.1), se observa que este empeoramiento inicial es más pronunciado. Esto se debe a que estas configuraciones introducen una mayor variabilidad en los datos iniciales en comparación con las otras dos configuraciones de la Tabla 7.1. Este aspecto se aborda con mayor detalle en la sección 7.2. Aunque el empeoramiento inicial sea mayor, se aprecia cómo a medida que avanza el horizonte de predicción este efecto se vuelve más débil. Por ejemplo, en el día 5 el mayor empeoramiento es del 15.90 %, mientras que en la predicción del primer día era del 64.01 %. Para el último día de la ventana de predicción el rendimiento de los modelos parece estabilizarse, de forma que las diferencias entre los conjuntos y el modelo determinista son mínimas.

Esto respalda la hipótesis de que los conjuntos de predicciones tienden a corregir errores a medida que avanza el horizonte temporal, beneficiándose de la diversidad introducida en las condiciones iniciales. Esta diversidad facilita la exploración de un abanico más amplio de posibles estados futuros, lo cual resulta especialmente beneficioso a largo plazo, cuando la incertidumbre es mayor debido a que el modelo se alimenta de sus propias predicciones en lugar de usar información reciente.

Configuración	**1 día**	**5 días**	**15 días**
Determinista	0.1086 (ref)	0.3083 (ref)	0.5855 (ref)
Gaussiano (μ=0, $\sigma = 0{,}01$)	0.55 %	0.03 %	0.00 %
Gaussiano (μ=0, $\sigma = 0{,}1$)	28.96 %	5.36 %	0.43 %
PF_res_15x15 ($\alpha = 0{,}05$)	1.56 %	0.16 %	0.02 %
P_res_2x12x12	64.01 %	15.90 %	1.62 %
P_res_2x3x3	61.17 %	14.83 %	1.47 %

Tabla 7.1: Comparación del empeoramiento porcentual del RMSE promedio en distintos horizontes temporales para la unificación de cada configuración de ruido, con respecto a una predicción determinista sin ruido (valores de RMSE de predicción determinista indicados como referencia).

La Figura 7.1 muestra la evolución promedio del RMSE y del sesgo a lo largo del horizonte de predicción para cada configuración evaluada. Como se ha mencionado, aunque no se observa una mejora clara en RMSE respecto al modelo determinista, sí se aprecia una convergencia progresiva entre ambos tipos de modelos hacia el final del horizonte temporal.

Una de las principales observaciones es que todos los modelos presentan un sesgo negativo, lo que indica que las temperaturas tienden a subestimarse (es decir, se pronostican valores menores de los reales). El sesgo también empeora a lo largo del horizonte de predicción, reflejando una acumulación del error. Sin embargo, algunos conjuntos de predicciones muestran una reducción del sesgo en comparación con el modelo determinista, especialmente en las predicciones más lejanas. Esto sugiere que las perturbaciones

iniciales ayudan a corregir el sesgo sistemático del modelo, aunque no es suficiente para superar el desempeño de la predicción determinista en términos de RMSE. En cualquier caso, las diferencias observadas no son especialmente significativas y el rendimiento entre las distintas configuraciones es comparable al determinista.

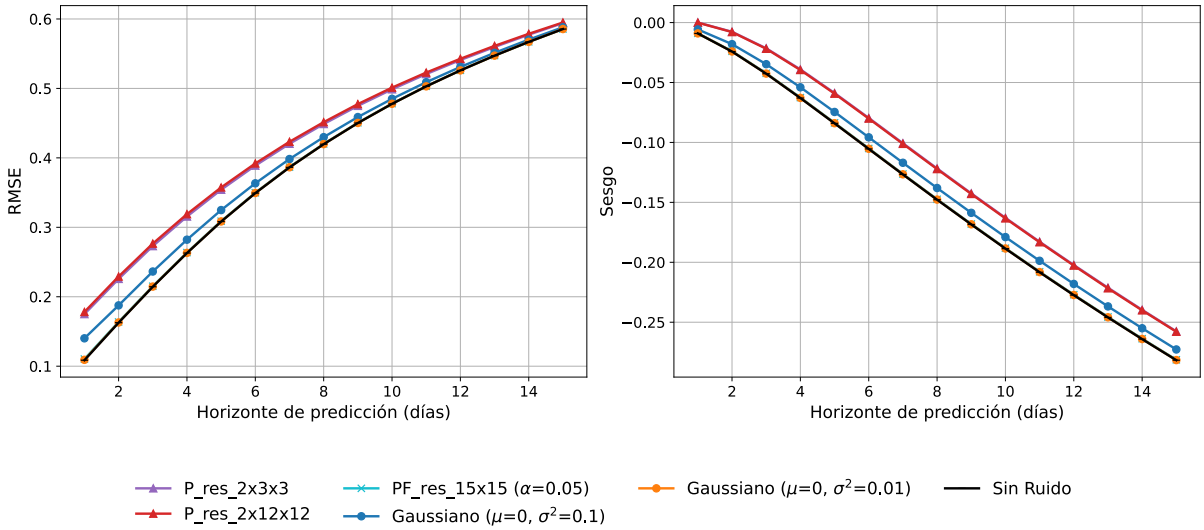


Figura 7.1: Gráficas del RMSE y del sesgo promedio por horizonte de predicción, comparando la combinación de distintos conjuntos de predicciones con un modelo determinista.

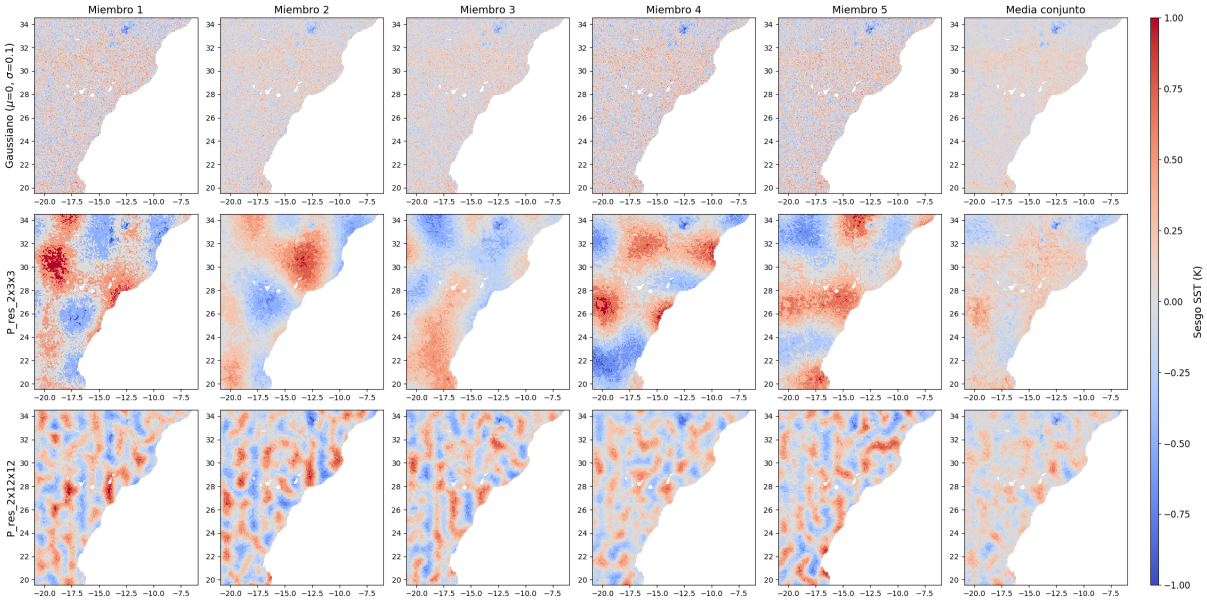


Figura 7.2: Mapas del sesgo de SST para predicciones a un día partiendo desde el día 2 de enero de 2022. Cada fila corresponde a una configuración diferente de ruido de entrada y cada columna representa un miembro del conjunto, con la última columna mostrando la media del conjunto.

La Figura 7.2 muestra el sesgo (ecuación 6.22) de las predicciones a un día de horizonte temporal para diferentes miembros dentro de varios conjuntos, así como para la

media de cada conjunto. Estas predicciones corresponden al día 3 de enero de 2022, con inicio de la predicción el 2 de enero. Los mapas reflejan lo comentado previamente sobre los distintos tipos de ruido, donde la primera predicción muestra una mayor dispersión entre los miembros del conjunto. Estas desviaciones ilustran los problemas que sufren los modelos probabilísticos en predicciones a corto plazo, donde la influencia del ruido es más notable, incrementando el RMSE en los primeros pasos temporales en comparación con el modelo sin ruido. Además, en la columna correspondiente a la media del conjunto estas diferencias se suavizan, mostrando cómo la incertidumbre aportada por cada miembro de la predicción contribuye a una salida unificada más robusta.

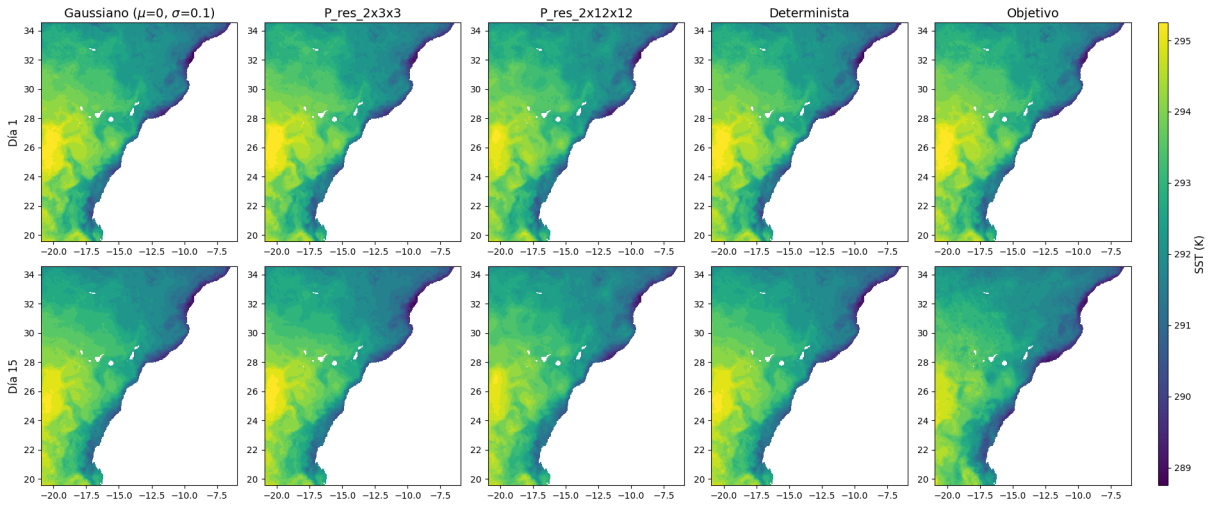


Figura 7.3: Mapas del SST para predicciones a uno y 15 días partiendo desde el 2 de enero de 2022. Cada columna corresponde a una configuración diferente de ruido de entrada y cada fila representa un horizonte de predicción. Las dos últimas columnas a la derecha corresponden a la predicción determinista y al estado objetivo.

La Figura 7.3 muestra las predicciones de SST obtenidas mediante la unificación de distintas configuraciones de conjuntos, así como las del modelo determinista y las observaciones reales (*target*). Esta imagen permite contextualizar la intensidad del ruido añadido en los estados iniciales, ya que se aprecia que las diferencias en las predicciones son bastante pequeñas y principalmente se concentran en la zona del afloramiento costero (costa africana) y en la parte del océano abierto al oeste de Canarias.

7.2. Comparación de configuraciones de ruido

En cuanto a la comparación entre las distintas configuraciones de ruido aplicadas a la entrada del modelo, esta se llevó a cabo en la fase preliminar mediante la obtención de 15 predicciones a partir de un único día de inicio (2 de enero de 2022). Las pruebas se organizaron en función del tipo de ruido utilizado, ya sea Gaussiano o de Perlin. En primer lugar, se presenta la evaluación de las diferentes configuraciones correspondientes al ruido Gaussiano. Posteriormente, se analizan comparaciones similares para las entradas con ruido de Perlin. Finalmente, se exponen conclusiones comunes derivadas tras el análisis de ambos tipos de ruido.

7.2.1. Ruido Gaussiano

Las configuraciones seleccionadas de ruido Gaussiano consisten en la extracción de muestras de una distribución normal con media cero ($\mu = 0$), variando la desviación estándar de dicha distribución. La Figura 7.4 muestra la evolución del CRPS (ecuación (6.14)) y sus componentes: el CRPS skill (ecuación (6.11)) y CRPS spread (ecuación (6.12)). En cuanto al CRPS, valores más cercanos a cero indican un mejor rendimiento del conjunto, ya que reflejan un equilibrio entre precisión (bajo error de la distribución predicha) y una dispersión adecuada entre las predicciones del conjunto, lo que favorece a los conjuntos de predicciones que exploran más alternativas con un error controlado.

En este caso, se observa que las predicciones cuyos estados iniciales han sido perturbados con un ruido Gaussiano de desviación estándar igual a 0.1 (caso Gaussiano (μ=0, σ=0.1)) presentan un CRPS mayor al comienzo del horizonte de predicción, cuando la predicción se basa en los dos estados iniciales ruidosos. Este valor alto se relaciona con un mayor error inicial, como se señaló en la sección anterior. Esto se confirma al analizar la gráfica del CRPS skill, donde dicha configuración presenta un mayor desacuerdo entre la distribución predicha y las observaciones reales en las etapas iniciales. El CRPS spread contribuye a reducir este error, aunque no al nivel de CRPS alcanzado por las otras configuraciones (Gaussiano (μ=0, σ=0.05) y Gaussiano (μ=0, σ=0.01)). Es importante tener presente que esta dispersión es dividida por el total de combinaciones de los modelos, por lo que al error ser tan grande no se contrarresta mucho. Estos conjuntos con una desviación estándar menor introducen menos ruido en los estados iniciales, lo que favorece a las predicciones cercanas al tiempo de inicio.

Sin embargo, a pesar de presentar un CRPS inicial más bajo, estas configuraciones experimentan un cambio de tendencia a lo largo del horizonte de predicción, llegando a mostrar el peor rendimiento al final del mismo. Como resultado, para el día 15 de predicción, la configuración con desviación estándar igual a 0.1 presenta el mejor CRPS con un valor cercano a 0.175.

Esto puede entenderse al analizar la gráfica de CRPS skill en la Figura 7.4, donde se observa que el error cometido por las predicciones de los conjuntos con menor perturbación crece de manera más pronunciada que en la configuración de desviación estándar igual a 0.1. Esta tendencia provoca que las diferencias en el error de las distintas configuraciones sean mínimas al final del horizonte temporal. Sin embargo, al ser su dispersión (CRPS spread) más baja, estos errores no se corrigen adecuadamente. Por ello, pese a que la dispersión entre los miembros sufre un ligero descenso desde el sexto día, el modelo con mayor ruido introducido termina siendo el que mejor representa la distribución predicha frente a las observaciones reales, considerando dicho nivel de dispersión. Por lo tanto, el ruido introducido debe tener cierta intensidad para favorecer la exploración de estados y la compensación de errores, aunque esta debe ser controlada.

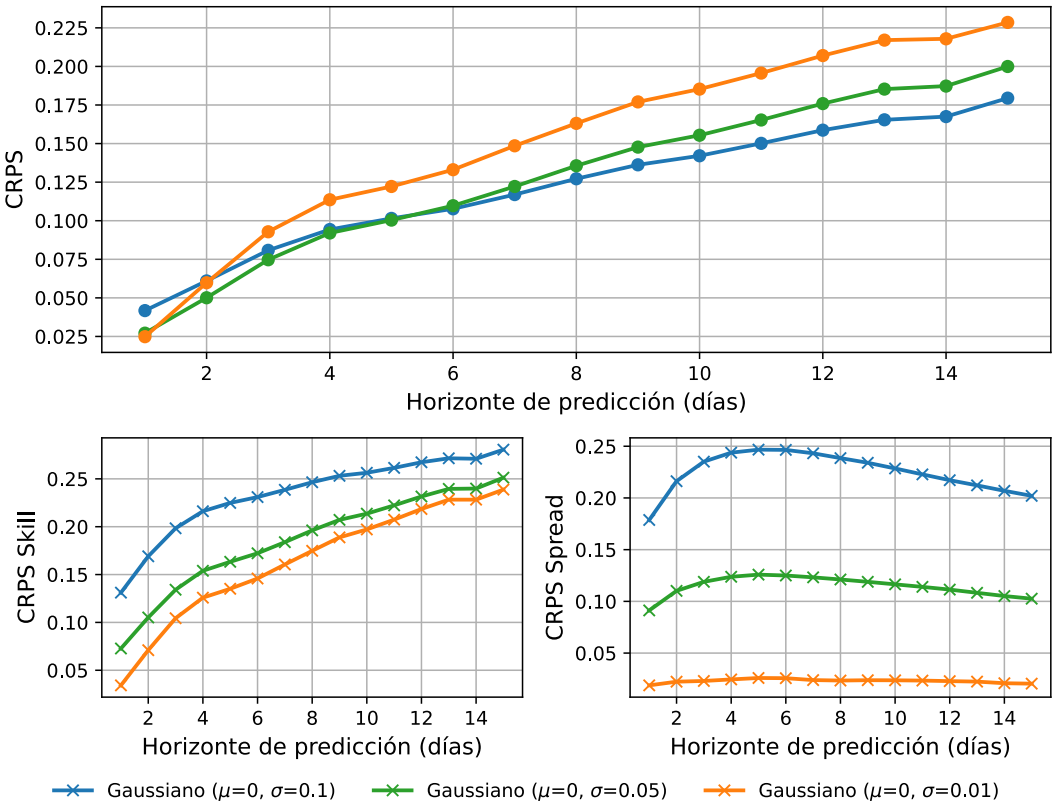


Figura 7.4: Gráfica del CRPS y sus componentes, promediados para cada horizonte de predicción, que compara los distintos conjuntos de predicciones con ruido Gaussiano introducido.

La Figura 7.5 muestra la evolución del *spread-skill ratio* (ecuación (6.17)) y sus componentes: la desviación estándar (ecuación 6.15) y el RMSE sin sesgo (ecuación 6.20). La línea horizontal en el valor 1 del eje y representa el umbral de referencia. Este sería el caso donde la dispersión de las predicciones del conjunto respecto a su media es capaz de explicar completamente los errores de dicha media, lo que indica una calibración perfecta. En ese caso, incluso sería posible sustituir una de las predicciones por la observación real sin alterar el resultado final.

En este caso, se observa que la configuración con desviación estándar igual a 0.01 presenta un comportamiento subdisperso a lo largo de todo el horizonte de predicción, con una dispersión insuficiente respecto al error cometido por su media. Esto se comprueba al comparar la gráfica de la desviación estándar respecto al RMSE, donde se observa que es siempre un valor menor. Las otras dos configuraciones comienzan siendo sobredispersas, con una dispersión mayor respecto a su media. Sin embargo, se detecta un patrón común en la evolución de los *spread-skill ratio*, ya que a partir del sexto día comienzan a estabilizarse. Esto concuerda con el hecho de que la desviación estándar de los conjuntos crece hasta el día 6, seguido de un descenso progresivo. Además, en este punto también se detecta que el valor de RMSE supera a de la desviación estándar, lo que explica el coeficiente menor que uno para el caso de desviación estándar igual a 0.1. Cabe destacar que este punto de inflexión coincide con el momento en el que esta configuración cruza la línea de referencia, donde el conjunto estaría perfectamente calibrado. Por lo tanto, pese a que todas las

configuraciones acaban siendo subdispersas, se puede deducir que con una perturbación algo mayor pero controlada, sería posible conseguir que la calibración del conjunto sea cercana a uno al final.

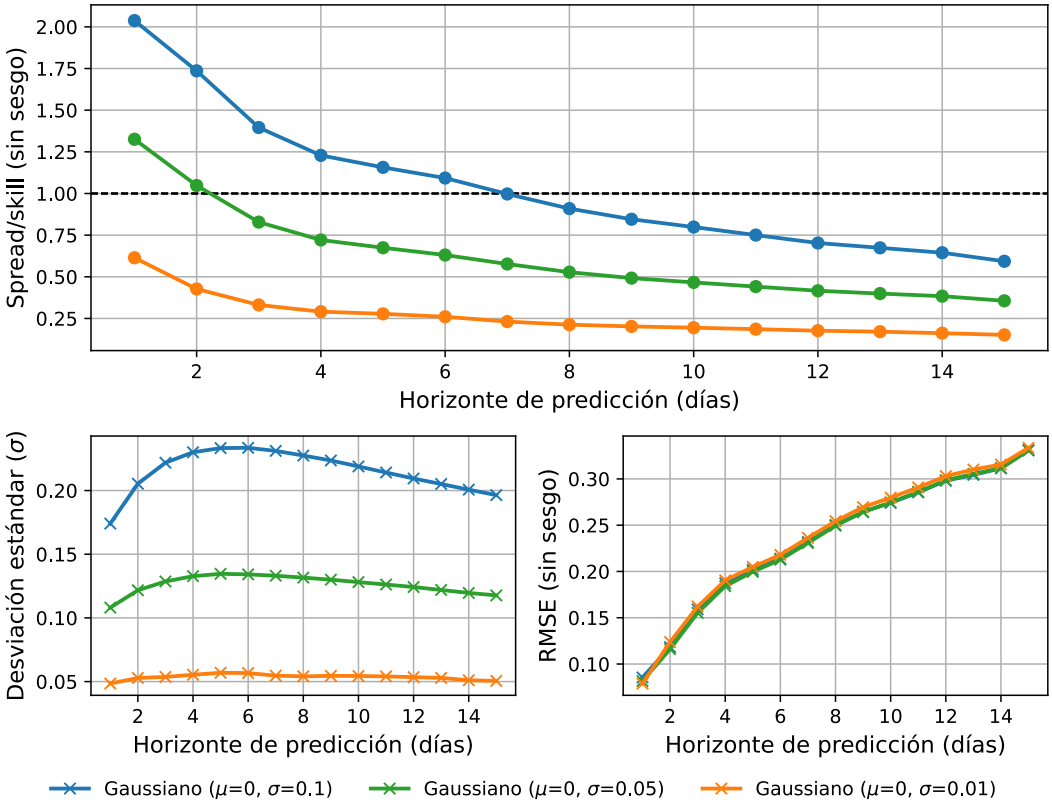


Figura 7.5: Gráfica del *spread-skill ratio* sin sesgo y sus componentes, promediados para cada horizonte de predicción, que compara los distintos conjuntos de predicciones con ruido Gaussiano introducido.

En la Figura 7.5 se observa que la corrección del RMSE mediante la varianza provoca que las trayectorias de los tres experimentos prácticamente coincidan. No obstante, en la Figura 7.6 se aprecia una diferencia en el RMSE al unificar los modelos, similar a la reflejada en la Tabla 7.1. La configuración con mayor dispersión presenta un RMSE inicial más alto que, al final del horizonte de predicción, se aproxima al valor de los modelos con menor ruido aplicado. El RMSE sin sesgo iguala estas trayectorias al aproximarse al caso ideal de un tamaño de conjunto infinito, lo que permite realizar una evaluación más justa de la calibración del conjunto mediante el *spread-skill ratio*, ya que reduce el efecto de contar con pocos miembros, como ocurre en este caso.

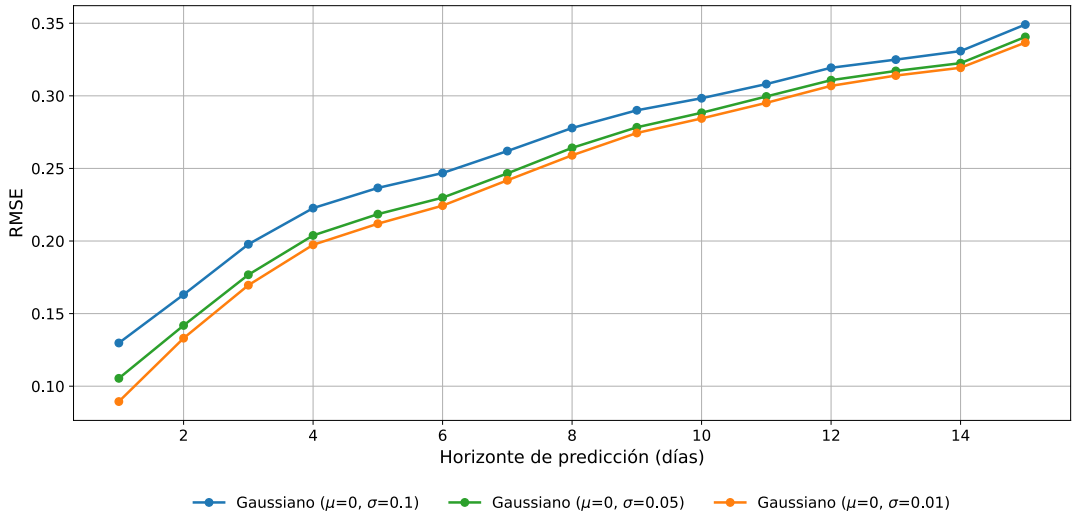


Figura 7.6: Gráfica de RMSE promediado por horizonte de predicción comparando el rendimiento de la unificación de las configuraciones de ruido Gaussiano.

7.2.2. Ruido de Perlin

El análisis de las configuraciones con ruido de Perlin resulta más complejo que el de aquellas basadas en ruido Gaussiano, debido al mayor número de variantes. Por ello, se opta por explicar su comportamiento de forma progresiva, a medida que se abordan las distintas métricas. Al igual que con el ruido Gaussiano, se comienza revisando el CRPS.

En la Figura 7.7 se muestra la gráfica del CRPS y sus componentes para las distintas configuraciones de ruido de Perlin. Aunque se intenta describir claramente cada configuración a lo largo del texto, las Tablas 6.4 y 6.5 pueden servir como referencia para consultar los parámetros exactos usados para generar las perturbaciones en cada experimento.

En primer lugar, los resultados correspondientes a los primeros días de predicción para el CRPS muestran que las configuraciones con ruido de Perlin base (P_res_2x3x3 y P_res_2x12x12) presentan un rendimiento inicial inferior. Sin embargo, se observa que estos experimentos acaban siendo los mejores al final del horizonte temporal en comparación con las configuraciones de Perlin fractal, siguiendo una tendencia similar al ruido Gaussiano (μ=0,σ^2=0.1) frente a sus variantes alternativas. Las configuraciones con ruido de Perlin fractal comienzan con valores de CRPS iniciales cercanos al ideal, pero su rendimiento se deteriora con el paso del tiempo.

Una de las diferencias entre ambos grupos es la complejidad del ruido. El ruido de Perlin fractal emplea múltiples iteraciones (octavas) para refinar el patrón, cada una de ellas con una resolución mayor. Sin embargo, la influencia de estas últimas octavas es menor, por lo que se agregan únicamente detalles finos. Por el contrario, el ruido de Perlin base es más sencillo, ya que genera el ruido una única vez con la resolución de entrada. Además, cabe recordar que las configuraciones de ruido de Perlin fractal incorporan un factor de escalabilidad del ruido, representado por α, que se aplica en la salida para atenuar su impacto.

Como se observa en la Figura 7.7, las configuraciones con ruido de Perlin base, al no estar afectadas por ningún parámetro de escalabilidad, muestran una mayor dispersión entre los miembros del conjunto. Esto implica que, si bien el error inicial de sus predicciones es elevado, este no crece tan rápidamente como en el caso de las configuraciones con

ruido de Perlin Fractal, y se ve parcialmente compensado por una mayor dispersión. Esto confirma el patrón observado con el ruido Gaussiano, donde una intensidad de ruido mayor controlada mejora la distribución de probabilidad predicha al considerar la incertidumbre, a pesar de un valor inicial peor.

El caso de PF_res_15x15 (α=0.4) resulta especialmente relevante, ya que permite observar cómo afecta la intensidad del ruido respecto al resto de configuraciones de Perlin fractal. Al tener una mayor escalabilidad (frente al α=0.2 por defecto y al α=0.05 de PF_res _15x15 (α=0.05)), su dispersión entre miembros es también superior, lo que provoca una tendencia del error similar a la de las configuraciones Perlin base, aunque algo menor. Esta configuración alcanza el mejor CRPS del grupo de Perlin fractal para el horizonte de predicción de 15 días.

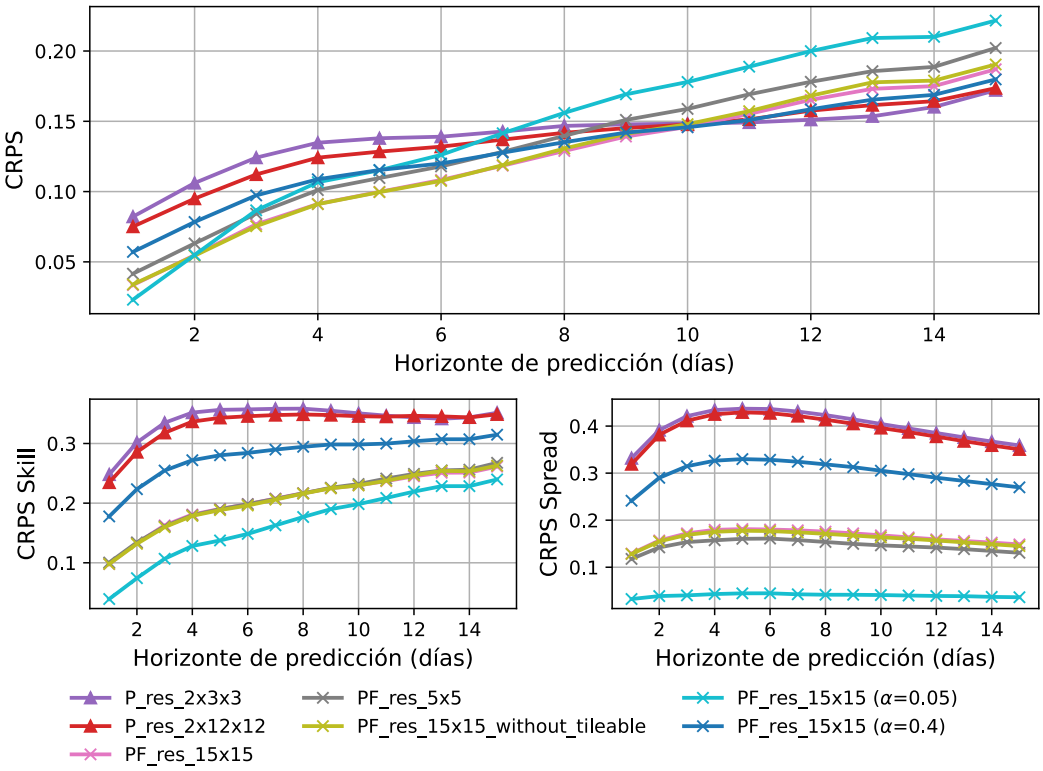


Figura 7.7: Gráfica del CRPS y sus componentes, promediados para cada horizonte de predicción, que compara los distintos conjuntos de predicciones con ruido de Perlin introducido.

La elección de la escalabilidad del ruido en 0.4 se realizó precisamente con el objetivo de que la magnitud del ruido fractal fuera comparable a la de los ruidos de Perlin base (ver apéndice A). Sin embargo, se observa que la dispersión entre miembros es menor en este caso, a pesar de utilizar una intensidad de ruido similar. Esto podría sugerir que el ruido de Perlin fractal no es tan beneficioso debido al refinamiento de ruido mediante octavas, el cual genera patrones espaciales complejos y con menor coherencia estructural respecto a los ruidos de Perlin base, afectando así a la incertidumbre.

En cuanto al *spread-kill ratio* sin sesgo, la Figura 7.8 muestra que las configuraciones de Perlin base vuelven a destacar. Aunque presentan sobredispersión a lo largo del horizonte

temporal, para el día 15 de predicción alcanzan un valor cercano a uno, lo que indica una calibración casi perfecta. Además, se observa que en todos los conjuntos la desviación estándar comienza a disminuir desde el sexto día, como ya ocurría con el ruido Gaussiano. Este descenso, junto con un incremento menos pronunciado del RMSE sin sesgo, provoca una estabilización del *spread-skill ratio*. Por el contrario, debido a la baja intensidad del ruido inicial, la mayoría de las configuraciones de Perlin fractal acaban siendo subdispersas durante la mayor parte del período de predicción. Esto indica que las predicciones son demasiado similares respecto a la media, mientras el error cometido es mayor.

La configuración PF_res _15x15 (α=0.4) sigue una tendencia similar a los ruidos de Perlin base, aunque acaba siendo subdispersa. En este caso, se observa que la configuración P_res_2x3x3 comparte un valor inicial muy similar, ya que, pese a que su desviación estándar es mayor, su RMSE sin sesgo es más elevado especialmente al inicio con respecto a los demás conjuntos. Este comportamiento del RMSE sin sesgo pone el foco sobre la importancia de las resolución espacial aplicada al ruido, ya que los peores resultados tienden a darse en configuraciones con resoluciones más bajas, como P_res_2x3x3 y PF_res_5x5.

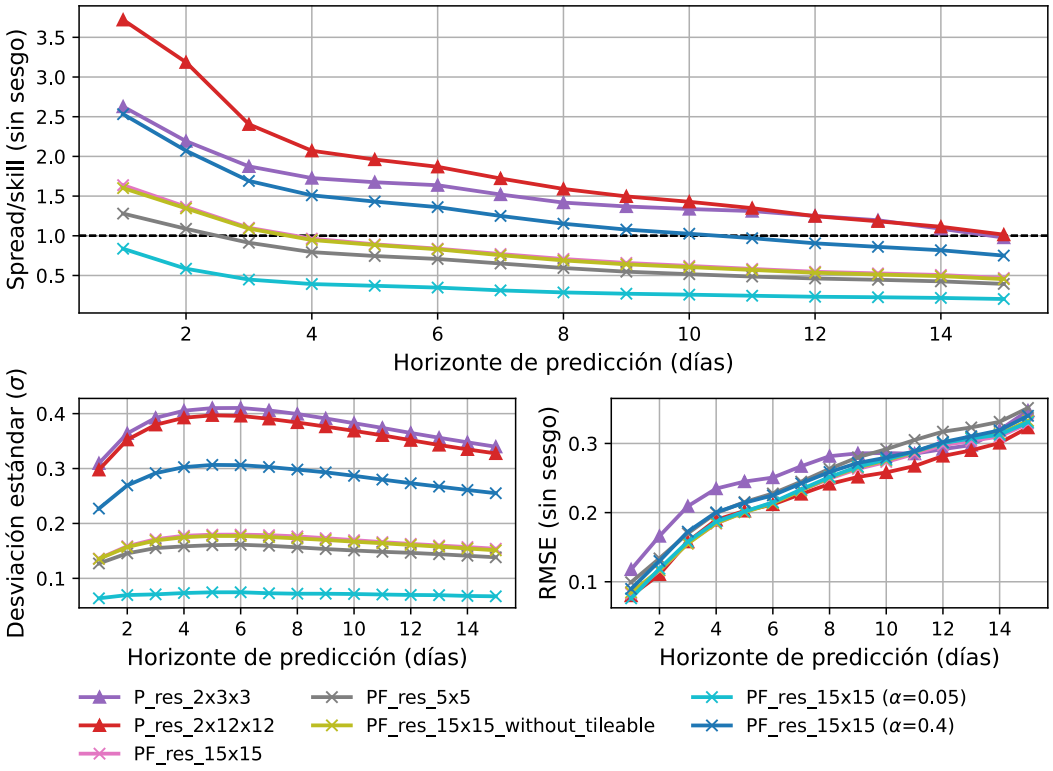


Figura 7.8: Gráfica del *spread-skill ratio* sin sesgo y sus componentes, promediados para cada horizonte de predicción, que compara los distintos conjuntos de predicciones con ruido de Perlin introducido.

En cuanto al RMSE sobre la unificación de las predicciones, la Figura 7.9 muestra que los ruidos de mayor intensidad vuelven a presentar el peor rendimiento a lo largo del horizonte temporal. Sin embargo, se confirma lo observado con el RMSE sin sesgo, ya que la resolución del ruido aplicado influye directamente en la pérdida del modelo. Las configuraciones con menor resolución generan menos patrones espaciales, lo que re-

duce la diversidad del conjunto en distintas zonas. Este comportamiento es especialmente destacable, ya que, a pesar de tener una menor magnitud de ruido, estas configuraciones presentan peores resultados. Esto sugiere que la cantidad y el detalle de los patrones espaciales puede ser un factor más relevante que la intensidad del ruido. Por el contrario, las resoluciones más altas producen patrones más detallados, lo que puede ayudar a representar mejor la incertidumbre en procesos a pequeña escala como el afloramiento costero.

A lo largo de esta sección, no se ha detectado que el parámetro de coherencia del ruido tenga un impacto relevante.

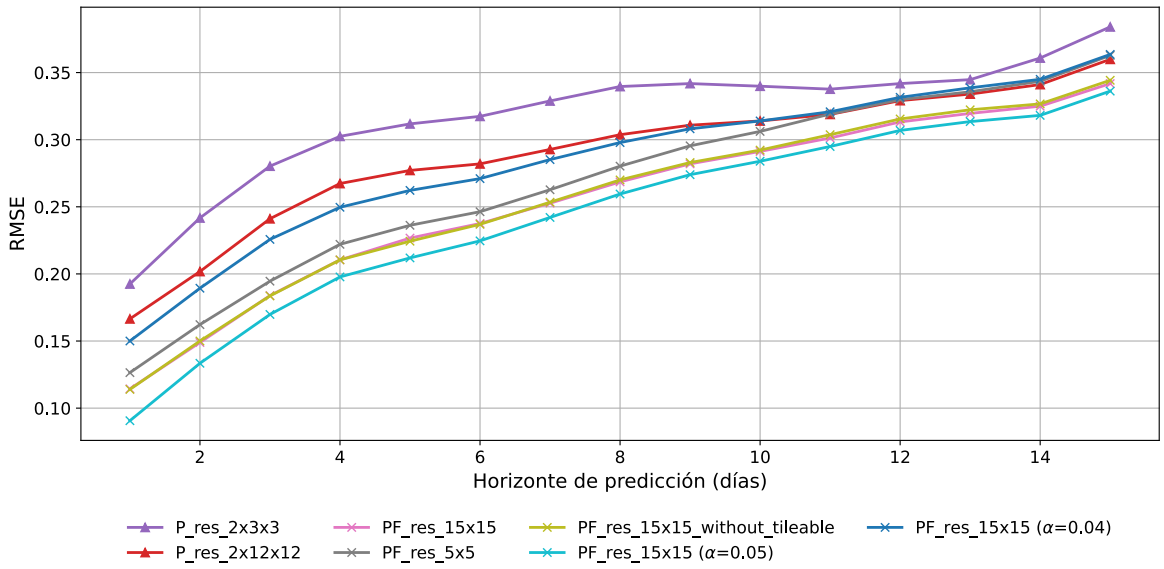


Figura 7.9: Gráfica de RMSE promediado por horizonte de predicción comparando el rendimiento de la unificación de las configuraciones de ruido de Perlin.

7.2.3. Comparación entre tipos de ruido

Los ruidos más prometedores se sometieron a una prueba adicional, en la que se realizaron predicciones partiendo desde el mismo día que las anteriores, pero con un aumento de los miembros del conjunto a 10. Sin embargo, en este caso, los resultados no mostraron diferencias significativas, observándose cambios mínimos en las tendencias anteriores.

En la evaluación final, se seleccionaron los ruidos con mejor rendimiento según las métricas anteriores para cada grupo analizado, junto con uno representativo de cada tipo de ruido con peores resultados, a modo de comparación. En la Figura 7.10 se muestra la evolución de las métricas probabilísticas CRPS, RMSE sin sesgo y *spread-skill ratio*, promediadas por horizonte de predicción y evaluadas sobre el conjunto de evaluación completo de dos años.

En estas gráficas se corroboran los resultados observados en las pruebas preliminares, donde se detectó que la intensidad controlada del ruido era un factor relevante para que el conjunto fuera lo suficientemente diverso (por ejemplo con el caso de Gaussiano desviación estándar 0.1). En comparación con aquellas gráficas, se observa que al promediar por un

mayor número de días de inicio de la predicción, las métricas se deterioran de forma general y las diferencias son menores. Pese a ello, se observa que los ruidos de Perlin tienen un mejor rendimiento respecto a las configuraciones de ruido Gaussiano. Aunque existe una diferencia en la distribución estadística del ruido entre las configuraciones de ambos grupos, las intensidades de ruido aplicadas son similares en aquellas que presentan un rendimiento comparable (como el ruido Gaussiano σ=0.1 y el Perlin base P_res_12x12). Esto sugiere que los modelos se benefician de perturbaciones iniciales que combinan una intensidad de ruido adecuada con cierta estructura espacial.

Respecto al RMSE sin sesgo, las diferencias entre configuraciones son mínimas. No obstante, se observa nuevamente la tendencia de que los ruidos de Perlin base y el Gaussiano de mayor desviación estándar, pasan a ser los de mejor rendimiento en el horizonte temporal de 15 días. Una vez más, el ruido de Perlin destaca levemente sobre el Gaussiano, resaltando la importancia de los patrones espaciales en las perturbaciones iniciales del conjunto.

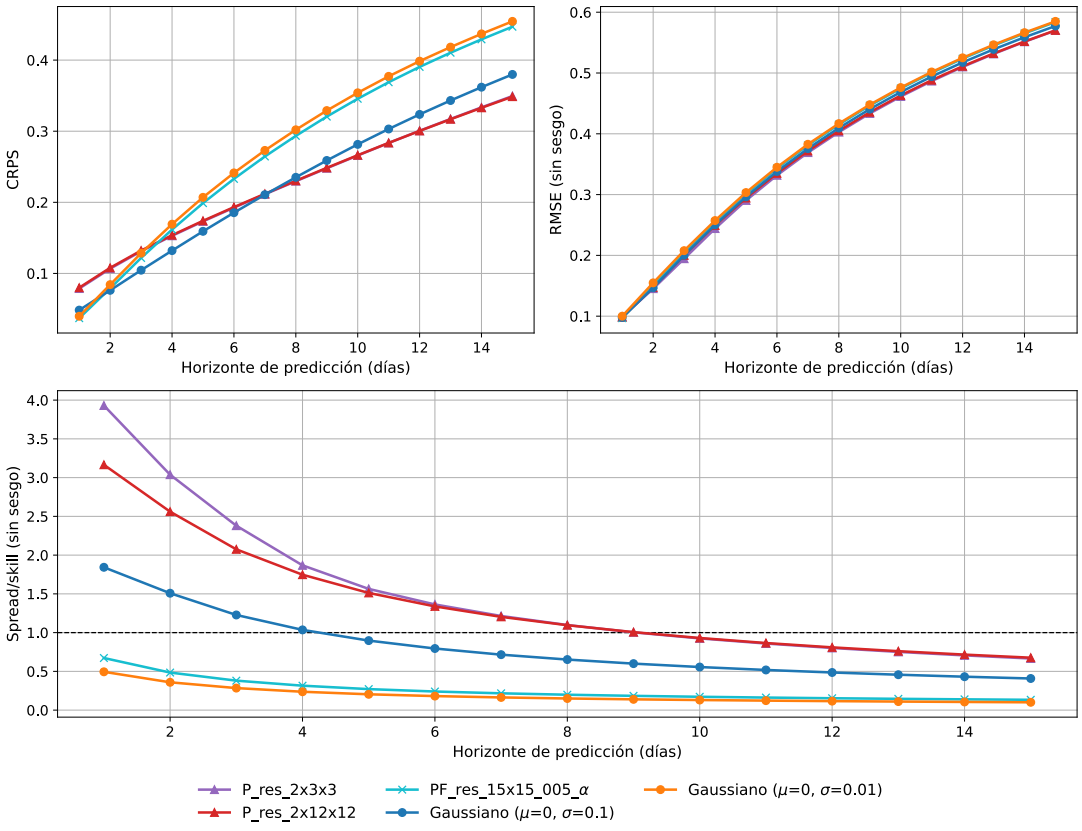


Figura 7.10: Gráficas del CRPS, RMSE sin sesgo y *spread-skill ratio* promediadas por horizonte de predicción, comparando los distintos conjuntos de predicciones con ruido añadido, evaluados sobre el conjunto completo de datos de prueba.

Capítulo 8

Conclusiones y trabajo futuro

Este capítulo tiene el propósito de dar una visión global de los resultados obtenidos y los objetivos conseguidos. El análisis no se limita exclusivamente las métricas presentadas en el capítulo 7, sino que también considera el trabajo realizado para adaptar SeaCast a la región de estudio, el punto de partida que ofrecen los resultados obtenidos, así como las limitaciones enfrentadas y líneas futuras de trabajo que podrían mejorar los resultados.

8.1. Análisis global de resultados

En cuanto a la adaptación de SeaCast, el principal reto encontrado fue la diferencia entre productos de datos. La exploración y la comparación del producto usado en la zona de estudio frente al de SeaCast, permitieron obtener una visión de la gran cantidad de variables y datos auxiliares que puede manejar un modelo climático. La flexibilidad demostrada por SeaCast en su adaptación a esta región demostró que se podría emplear para otros dominios específicos, facilitando la creación de modelos en zonas de interés que no dispongan de modelos climáticos globales, siempre y cuando sea posible contar con datos de alta resolución.

Respecto a los resultados obtenidos con los conjuntos de predicciones, si bien no son completamente positivos, ya que su uso no ha mejorado el error en comparación con las predicciones deterministas, sí se pueden extraer algunas conclusiones valiosas que pueden servir como base para futuros trabajos.

En primer lugar, el modelo base utilizado era relativamente sencillo, tanto en el número de variables como en su proceso de entrenamiento (sin pasos autorregresivos). Además, este modelo fue entrenado de forma determinista, sin perturbaciones en los datos iniciales. Por ello, el hecho de que varios conjuntos de predicciones con perturbaciones en los estados iniciales hayan alcanzado un rendimiento similar al modelo sin ruido es una señal prometedora. Esto sugiere que no es imprescindible contar con grandes conjuntos de datos modificados ni con arquitecturas complejas o conjuntos de modelos sofisticados para, al menos, igualar el desempeño del modelo base a largo plazo. Esto verifica también que los conjuntos de predicciones, pese a tener un peor desempeño en las primeras etapas de la predicción, compensan la diversidad de los errores entre modelos, haciendo que la media de estos sea similar a la predicción sin ruido. Si bien no se ha podido verificar con certeza, es posible que, a un horizonte temporal más amplio, las mejores configuraciones de conjuntos identificadas logren un rendimiento superior a un único modelo determinista.

En segundo lugar, se ha identificado qué tipo de perturbación resulta más efectiva desde el punto de vista del conjunto de predicciones para este modelo oceanográfico.

Se concluye que las perturbaciones iniciales generadas con un nivel moderado de ruido y con patrones espaciales estructurados (como el ruido de Perlin) son más beneficiosas que aquellas generadas con ruido disperso (como el ruido Gaussiano), que carecen de un componente espacial claro. Además, se ha verificado que iterar sobre patrones espaciales de ruido, aplicando detalles de ruido más fino en pasos sucesivos (como en el caso de Perlin fractal), no aporta mejoras significativas. Dentro de los patrones espaciales, aquellos que presentan una mayor resolución, y, por lo tanto, una mayor cantidad de patrones, tienden a ofrecer mejores resultados. Esto podría ser indicativo de que este tipo de ruido permite introducir mayor diversidad en zonas a pequeña escala.

8.2. Cumplimiento de objetivos

En cuanto a los objetivos, se logró la adaptación de SeaCast a una nueva zona de estudio, tal y como se detalló en la sección anterior. Asimismo, se profundizó en el funcionamiento del modelado climatológico en trabajos previos, como Neural-LAM o GraphCast (antecesores de SeaCast), identificando las diferencias entre ellos y cuáles son las ventajas que supone la evolución en la definición de mallas.

También se adquirió un conocimiento sólido sobre las GNNs, una herramienta que no se había utilizado previamente en el contexto de la formación previa. Esta técnica demostró ser eficaz en el contexto de predicción oceanográfica, destacándose por su capacidad para poder capturar y modelar fenómenos en contextos espaciales irregulares.

Otra lección importante es el manejo de grandes volúmenes de datos. Durante el desarrollo de este proyecto se trabajó con conjuntos de datos multidimensionales, lo que permitió comprender su estructura y modelarlos desde archivos `numpy` para poder trabajar con la librería de métricas de WeatherBench-X. Otra consecuencia de la gestión de grandes volúmenes de datos fue el desarrollo de métodos y técnicas para procesar el cálculo de métricas por secciones, de forma que pudieran computarse sin errores de memoria.

Por otro lado, el aprendizaje por conjuntos ha permitido explorar un nuevo enfoque, que, si bien ha estado presente tradicionalmente en modelos como Random Forest, no se había trabajado durante el grado con técnicas de aprendizaje profundo hasta el desarrollo de este proyecto. Este enfoque también facilitó la introducción a las métricas de evaluación probabilísticas, que buscan determinar el error entre la distribución predicha y las observaciones reales (CRPS), así como evaluar si el conjunto representa adecuadamente la incertidumbre (*spread-skill ratio*). Además, se ha profundizado en las diversas estrategias para crear conjuntos de modelos o predicciones diversas, que se pueden ajustar a los requisitos computacionales y necesidades del problema.

Por último, no se logró demostrar una mejora clara en la capacidad predictiva al usar conjuntos de modelos en una zona que, en principio, podría beneficiarse de ellos debido a la presencia de afloramiento costero. Pese a ello, se han conseguido identificar tipos de perturbaciones iniciales que pueden servir como punto de partida para proseguir esta investigación desde enfoques alternativos, que no se limiten exclusivamente a la perturbación de datos oceanográficos iniciales en inferencia o con una mayor cantidad de variables. Por otra parte, aunque se contemplaba la posibilidad de realizar ajustes de hiperparámetros sobre el modelo, esto no fue posible debido a las limitaciones computacionales, al igual que la implementación de técnicas de aprendizaje por conjuntos que funcionaran de forma paralela (por ejemplo, el Stacking). En compensación, sí que se trabajó en paralelo entrenando modelos y realizando pruebas de inferencia con pesos previamente entrenados, con el objetivo de acelerar el desarrollo. Aun así, los resultados obtenidos en las etapas

iniciales fueron limitados, debido al tiempo requerido para ejecutar predicciones en el equipo disponible.

8.3. Limitaciones del trabajo

El principal reto abordado en este trabajo ha sido el uso de recursos computacionales, ya que, pese a la potencia del ordenador empleado para el entrenamiento, el modelo seguía tardando una cantidad excesiva de tiempo en entrenar. Como referencia, el entrenamiento de SeaCast originalmente contó con 32 GPUs disponibles [Holmberg et al., 2024]. Esta limitación obligó a descartar el uso de pasos autorregresivos durante la predicción, ya que aumentaba de manera significativa el tiempo de ejecución. Además, también se redujeron la cantidad de datos de entrenamiento en comparación a los empleados en SeaCast. De la misma manera, también se limitaron las conexiones entre mallas del modelo para una ejecución más ligera. Todos estos ajustes pudieron haber afectado al rendimiento del modelo, especialmente en la fase de evaluación donde se alimenta de sus propias predicciones tras la realización de dos predicciones iniciales. A causa de esto tampoco se consideraron opciones de aprendizaje por conjuntos introduciendo la diversidad en la fase de entrenamiento, ya que hubiera sido muy costoso realizar el entrenamiento individual para pruebas de cada tipo de ruido o modificación realizada. Aunque en menor medida, el tiempo de ejecución de las predicciones también dificultó la realización de la tarea, por lo que se optó por realizar pruebas preliminares más cortas que permitieran obtener unas primeras conclusiones. Asimismo, se debe considerar el número reducido de miembros de los conjuntos, aunque se ha tratado de mejorar la interpretación con el uso de métricas sin sesgo siempre que fue posible.

Otra de las limitaciones es la disponibilidad de variables oceanográficas en la zona de estudio. Si bien existen conjuntos de datos de buena resolución, la cantidad de variables disponibles es baja en comparación a los datos del Mediterráneo con los que trabajaba SeaCast originalmente. Esto también afecta los resultados obtenidos en el análisis de conjuntos de predicciones, dado que las perturbaciones se aplicaron sobre una única variable. Aunque los resultados parecen ser generalizables a otro tipo de variables climáticas, se recomienda precaución en cuanto a la magnitud e intensidad del ruido en función del rango de valores de las variables sobre las que se trabaja.

8.4. Líneas de trabajo futuro

En cuanto a las líneas de trabajo futuro, este estudio puede servir como referencia para comenzar otras investigaciones usando técnicas similares de perturbación de condiciones iniciales. Una posible dirección, en caso de contar con la capacidad computacional suficiente, sería entrenar el modelo base para la inferencia con un límite de cuatro pasos autorregresivos. Esto podría mejorar la precisión de las predicciones, ya que el modelo empleado en este trabajo solo fue entrenado únicamente usando un paso autorregresivo, limitando su capacidad para alimentarse de sus propios resultados y generar predicciones consistentes a largo plazo.

Otra línea de mejora está relacionada con las perturbaciones iniciales aplicadas durante las ejecuciones de inferencia. En este proyecto, el ruido se añadió exclusivamente a los estados oceanográficos iniciales del modelo. Considerando que se trabajó con una sola variable, esta diversidad podría no ser suficiente para captar fenómenos complejos como

el afloramiento costero, que probablemente requieran de un mayor número de variables para explicarlos adecuadamente. Este escenario también permitiría validar las conclusiones extraídas en este trabajo sobre cuáles son los tipos de ruido más beneficiosos para crear conjuntos de predicciones a mayor escala. El uso de perturbaciones estocásticas podría ser beneficioso en este caso, aunque esto dependería de la disponibilidad de información muy específica, la cual podría no estar fácilmente accesible.

Además, se podrían emplear otros tipos de ruido o técnicas de conjuntos en inferencia, como las descritas en la sección 6.1.4. Algunas de las opciones incluyen el uso de predicciones rezagadas, la perturbación de estados iniciales de forzamiento o la modificación de los pesos del propio modelo para distintas ejecuciones durante inferencia. Otro ejemplo es GenCast, donde se parte de un estado de ruido puro para obtener una predicción calculando la diferencia respecto al estado anterior, lo cual podría usarse como estado inicial perturbado. Sin embargo, este componente requiere un entrenamiento independiente.

Una de las ventajas de las técnicas de aprendizaje por conjuntos es su gran flexibilidad, por lo que futuros trabajos podrían basarse en introducir la diversidad en la fase de entrenamiento en lugar de inferencia, en caso de contar con los recursos computacionales necesarios. Esto permitiría crear un modelo base realmente probabilístico, que podría ayudar a mejorar las predicciones de los conjuntos frente a técnicas deterministas. Este enfoque también permitiría probar otras técnicas de combinación de modelos, como los metamodelos. Además, si se modificara la arquitectura de la GNN empleada, sería posible estudiar cómo afecta el uso de modelos base que combinan sus salidas tras la última capa, en comparación con aquellos que lo hacen entre cada nivel de resolución de la malla, antes de los distintos procesos de paso de mensajes. Esta fue una de las limitaciones identificadas en la sección 2.8.

Aunque en este trabajo no se han obtenido mejoras significativas al aumentar el número de miembros del conjunto, es probable que esto se deba al reducido número de predicciones empleadas, condicionado por las limitaciones computacionales. Esta deficiencia se evidenció con el análisis del RMSE sin sesgo, donde en varios escenarios los resultados cambiaron drásticamente. Muchas técnicas de conjuntos requieren un número elevado de miembros. Por ejemplo, AIFS ENS emplea 51 predicciones distintas, mientras que GenCast utilizó 50 durante su evaluación. Por ello, una alternativa sería aumentar considerablemente el número de miembros del conjunto, para validar el funcionamiento de las distintas configuraciones de ruido extraídas de una mayor muestra.

Por lo tanto, este trabajo pretende servir como una referencia inicial en el estudio de los tipos de ruido aplicados durante la fase de inferencia como mecanismo para introducir diversidad en conjuntos de predicciones. No obstante, como se ha expuesto a lo largo de esta sección, las posibilidades de configuración e incorporación de diversidad en este tipo de técnicas son amplias. Como consecuencia, futuras líneas de investigación podrían orientarse a mejorar los resultados obtenidos en este proyecto utilizando las mismas técnicas, o bien explorar cualquiera de las otras opciones mencionadas.

Apéndice A

Comparación de tipos de ruido

En este apéndice se muestran ejemplos de la perturbación en los estados iniciales añadida por las distintas configuraciones de ruido utilizadas para conformar un conjunto de predicciones, que se detallaron en la sección 6. Se realiza un análisis tanto espacial del ruido como de su distribución estadística para los distintos tipos de ruido, agrupando las distintas configuraciones según su origen (Gaussiano, de Perlin o de Perlin fractal). Se emplea la notación creada en las Tablas 6.3, 6.4 y 6.5, por lo que se recomienda usarlas como referencia.

A.1. Configuraciones de ruido Gaussiano

La Figura A.1 muestra las diferencias entre las distintas configuraciones de ruido Gaussiano aplicadas. La configuración de Gaussiano (μ=0, σ=0.1) es la que introduce una mayor magnitud de ruido de las tres, lo que explica su mejor capacidad para representar la incertidumbre en un conjunto de predicciones. En cambio, las configuraciones con una desviación estándar reducida generan perturbaciones menos significativas, asemejándose más a las predicciones deterministas. Además, se puede apreciar que el ruido generado en todos los casos no presenta ningún componente espacial, ya que se trata de ruido completamente aleatorio y no estructurado.

Por otro lado, la Figura A.2 muestra las funciones de densidad asociadas a cada configuración. En todas ellas se observa una distribución con forma de campana centrada en cero (debido al uso de μ=0), característica de este tipo de ruido. La diferencia entre las configuraciones se encuentra en su desviación estándar. Cuanto menor sea esta, más estrecha y concentrada es la distribución (como en el caso Gaussiano(μ=0, σ=0.01)). Se observa como el aumento de la desviación estándar provoca que la dispersión de los valores aumente, donde el caso σ=0.1 llega a alcanzar valores cercanos a 0.4 en las colas de la distribución.

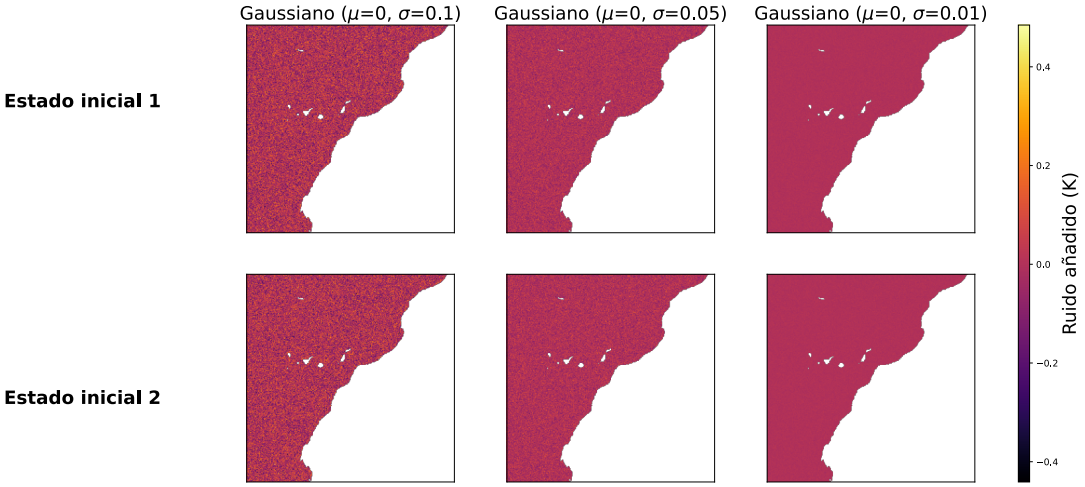


Figura A.1: Mapas del ruido añadido a la SST en dos estados iniciales diferentes, utilizando configuraciones de ruido Gaussiano.

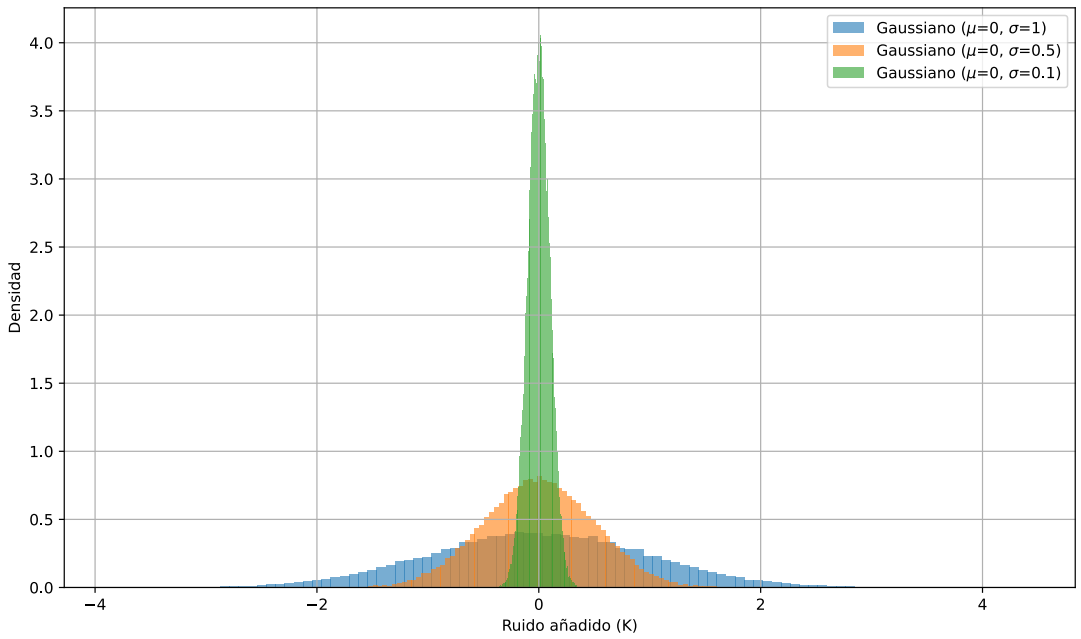


Figura A.2: Distribuciones del ruido añadido a la SST para diferentes configuraciones de ruido Gaussiano. Se representa un histograma normalizado de los valores aplicados al primer estado inicial.

A.2. Configuraciones de ruido de Perlin

La Figura A.3 muestra una comparación del ruido de Perlin añadido en dos estados iniciales, correspondiente a las configuraciones P_res_2x12x12 y P_res_2x3x3. En estas imágenes se observa claramente el efecto de la resolución en el patrón de ruido generado.

La configuración con resolución tres en los ejes espaciales genera patrones de ruido menos detallados y más amplios, ya que divide los datos originales en un número menor de celdas. Esto resulta en una interpolación de valores más suave. Por otro lado, la configuración con resolución 12 crea un mayor número de celdas por eje, lo que da lugar a patrones de ruidos más finos y detallados, además de una intensidad ligeramente mayor debido a la transición más rápida entre valores dentro de las celdas. Esta mayor intensidad se explica porque, al aumentar la resolución, el ruido se genera en escalas espaciales más pequeñas, lo que provoca variaciones más grandes entre puntos vecinos.

La Figura A.4 muestra como la distribución de los datos cambia ligeramente en comparación a los ruidos Gaussianos, ya que no presenta una forma de campana de Gauss tan definida. Pese a ello, se observa como la mayoría de los valores de la distribución se concentran en alrededor de cero y los valores extremos ocurren con menor frecuencia, lo que refleja el suavizado de la función de interpolación. Además, se aprecia que un aumento de la resolución está asociado con una mayor intensidad de ruido, aunque estos ruidos se encuentran en la cola de la distribución y por lo tanto no tienen tanto peso. Sin embargo, si se puede observar que la distribución correspondiente a la resolución 12 en los ejes espaciales es más achatada y dispersa, lo que implica una mayor cantidad de ruido, aunque de forma ligera, en comparación con la resolución tres.

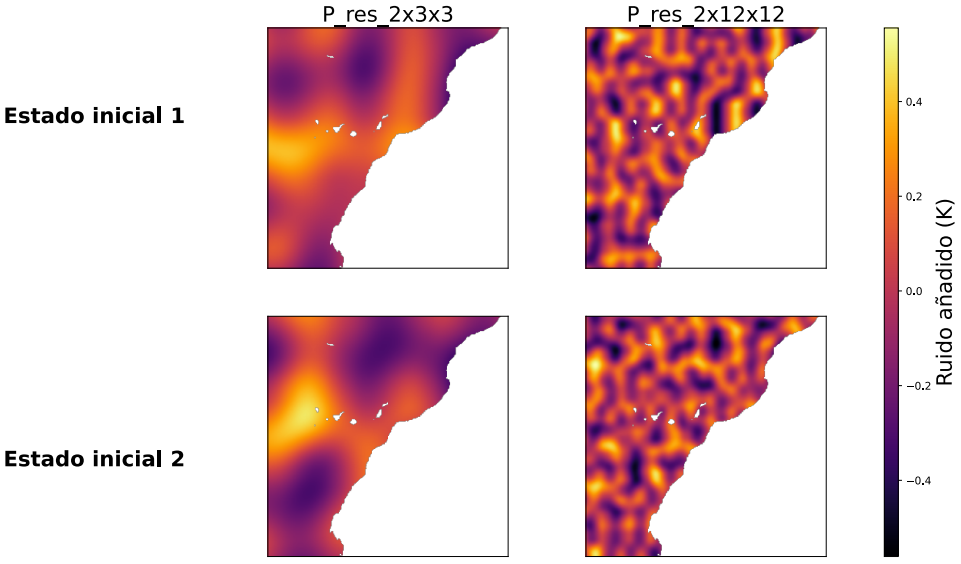


Figura A.3: Mapas del ruido añadido a la SST en dos estados iniciales diferentes, utilizando configuraciones de ruido de Perlin base.

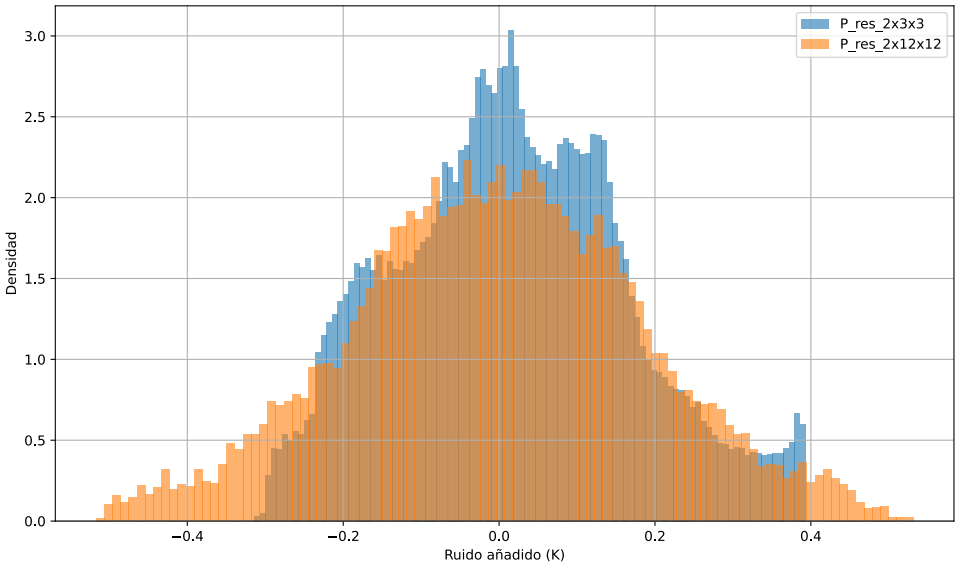


Figura A.4: Distribuciones del ruido añadido a la SST para diferentes configuraciones de ruido de Perlin base. Se representa un histograma normalizado de los valores aplicados al primer estado inicial.

A.3. Configuraciones de ruido de Perlin fractal

En la Figura A.5 muestra las diferencias entre las configuraciones de ruido Perlin fractal, donde se observa un patrón similar al ruido de Perlin base en cuanto a la influencia de la resolución espacial. El ruido con una resolución espacial en cada eje de cinco (PF_res_5x5) genera patrones espaciales más amplios, mientras que las configuraciones con resolución inicial de 15 presentan una mayor complejidad debido a la mayor cantidad de patrones generados.

Respecto al parámetro de escalabilidad del ruido (α), se observa que, al usar el valor por defecto de 0.2 en las configuraciones de PF_res_15x15, PF_res_15x15_without_tileable y PF_res_5x5, las magnitudes del ruido generado se igualan a pesar de las diferencias en resolución. Esto sugiere que la comparación entre estas configuraciones es algo más equilibrada, pues lo que realmente se evalúa es el impacto de la cantidad de patrones del ruido sobre el rendimiento de los conjuntos. Sin este parámetro, se observó que las configuraciones generaban ruidos mayores a uno, que es una perturbación muy grande para el ámbito de predicción.

En cuanto a la coherencia del ruido en uno de los ejes (`tileable`), no se observan grandes diferencias entre la configuración de referencia de Pangu-Weather (PF_res_15x15), donde el ruido es coherente en el eje longitudinal, y la configuración sin coherencia (PF_res_15x15_without_tileable). Una posible explicación es que el ruido generado presenta una complejidad considerable, a causa de la iteración mediante octavas y una alta resolución inicial, lo que dificulta la identificación de patrones dentro de un mismo estado inicial.

En contraste, la Figura A.3 muestra que dos perturbaciones iniciales generadas con ruido de Perlin base, aunque distintas entre sí, comparten un patrón general reconocible. Esto confirma que el ruido mantiene cierta coherencia en el eje temporal. Es importante distinguir entre el comportamiento de coherencia para la versión de tres dimensiones de Perlin y la de dos dimensiones. Mientras que en la de dos dimensiones se busca evitar

transiciones bruscas en los valores a lo largo del eje longitudinal, en la función de tres dimensiones el objetivo es mantener una transición suave entre los dos estados iniciales. Esta propiedad se observa con mayor claridad en la configuración P_res_2x3x3, donde se aprecia una mayor continuidad entre los estados generados. Sin embargo, en la configuración P_res_2x12x12 es algo más difícil de identificar, nuevamente debido a la mayor complejidad del patrón introducida por una resolución de ruido más elevada.

En el caso del ruido de Perlin fractal, se trabaja únicamente con funciones de dos dimensiones. Debido a que la resolución del ruido aumenta con cada iteración, al aplicar ruido sobre múltiples estados iniciales, la resolución en el eje temporal escala rápidamente, superando la longitud real del propio eje. Por ello, no se utiliza una función tridimensional en ningún momento, salvo en la prueba fallida de la configuración PF_res_1x3x3.

La modificación del parámetro de escalabilidad de ruido hace que la configuración PF_res_15x15 (α=0.05) presente intensidades de ruido similares a las del ruido Gaussiano con baja desviación estándar. Por otro lado, la configuración (α=0.4) iguala las intensidades de ruido obtenidas tanto con las configuraciones de ruido de Perlin base como con el ruido Gaussiano con desviación estándar igual a 0.1. Esta equivalencia es fundamental para permitir una comparación más justa entre los distintos tipos de conjuntos de predicciones.

La Figura A.6 muestra cómo la distribución de los datos está determinada por el parámetro de escalabilidad de ruido aplicado a las distintas configuraciones. Se observa que la dispersión es cercana a 0.2 para las configuraciones que utilizan el valor por defecto, lo que confirma que el ajuste de escalabilidad corrige el efecto de emplear diferentes resoluciones. Por ejemplo, la configuración PF_res_5x5 presenta una distribución muy similar al resto de configuraciones con esta escalabilidad, a pesar de tener una menor resolución. En cambio, la configuración PF_res_15x15 (α=0.05) tiene una distribución más estrecha, con valores más concentrados en torno a 0. Por otro lado, la configuración PF_res_15x15 (α=0.4) presenta una distribución más dispersa debido a su mayor escalabilidad, asimilándose a las distribuciones de ruido de las configuraciones de Perlin base mostradas en la Figura A.4.

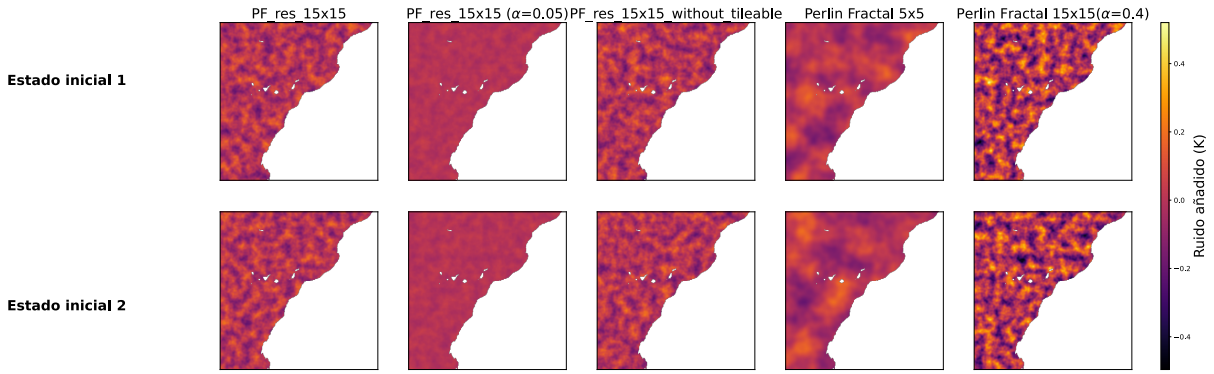


Figura A.5: Mapas del ruido añadido a la SST en dos estados iniciales diferentes, utilizando configuraciones de ruido de Perlin fractal.

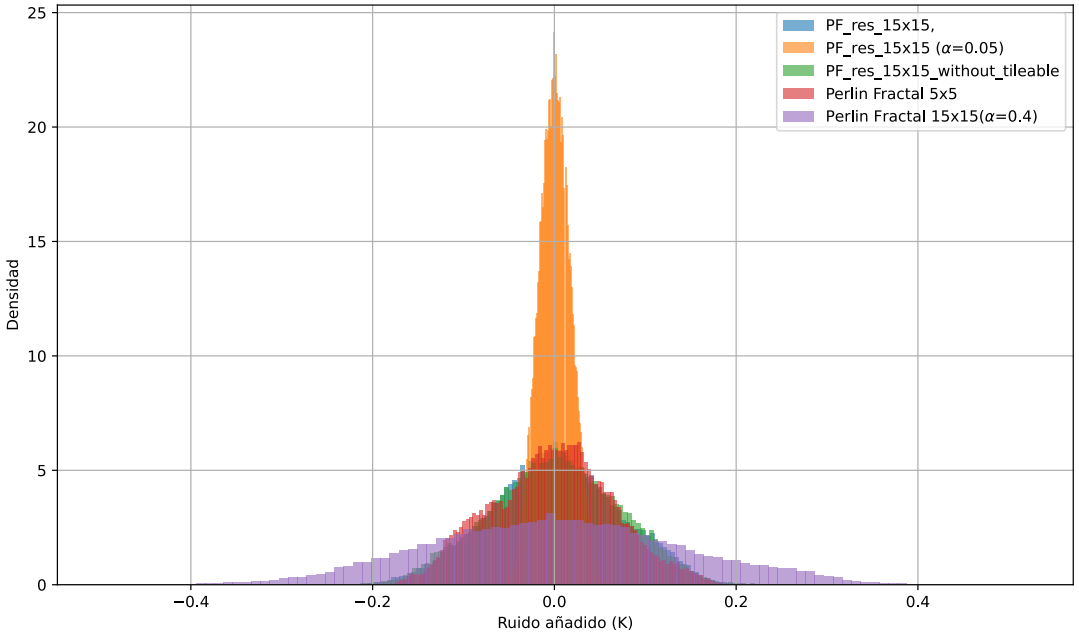


Figura A.6: Distribuciones del ruido añadido a la SST para diferentes configuraciones de ruido de Perlin fractal. Se representa un histograma normalizado de los valores aplicados al primer estado inicial.

Apéndice B

Competencias específicas cubiertas

En este apéndice se presentan las competencias específicas del plan de estudios del Grado en Ciencia e Ingeniería de Datos que se alinean con este proyecto. A lo largo del desarrollo de este trabajo, uno de los propósitos principales ha sido reforzar y aplicar los conocimientos y habilidades adquiridos durante la carrera. Por lo tanto, la siguiente lista actúa como una referencia de las competencias alcanzadas, acompañada con su justificación basada en las tareas realizadas durante el proyecto.

- **ED3:** Capacidad para conocer y desarrollar técnicas de aprendizaje computacional y diseñar e implementar aplicaciones y sistemas que las utilicen, incluyendo las dedicadas a extracción automática de información y conocimiento a partir de grandes volúmenes de datos.

  En este proyecto se realizó un estudio profundo del modelo SeaCast y su arquitectura, con el fin de adaptarlo al caso de uso específico. Esto incluyó la modificación de las entradas esperadas por la red, así como la actualización del código para permitir que el entrenamiento se ejecute en una única GPU. Además, se incorporó un componente que permite perturbar las condiciones iniciales empleadas por el modelo, con el objetivo de crear conjuntos de predicciones diversas y aplicar técnicas de aprendizaje por conjuntos. Los datos oceanográficos manejados son multidimensionales y de gran volumen, lo que requirió implementar una estrategia eficiente para su procesamiento y el cálculo de métricas de evaluación de los conjuntos.

- **ED4:** Capacidad de identificar y analizar problemas y diseñar, desarrollar, implementar, verificar y documentar soluciones software en ámbitos de aplicación de la Inteligencia Artificial en Ciencia e Ingeniería de Datos.

  Durante el desarrollo del proyecto se adoptó un enfoque de prueba y error. En las etapas iniciales de adaptación del modelo fue necesario identificar qué partes del modelo presentaban problemas de compatibilidad con las nuevas entradas. Este mismo procedimiento se empleó para adaptar la librería WeatherBench-X al formato de archivos utilizado. Asimismo, durante la fase de entrenamiento se realizaron múltiples ejecuciones para determinar cuál era la configuración de ruido más adecuada para el estudio. Todo el proceso y el flujo de trabajo han sido detallados a lo largo de este documento.

  Además, se creó una *issue* de GitHub para consultar a los autores de SeaCast sobre posibles problemas en la descarga de datos y se corrigió otro error detectado por el cotutor relacionado con la implementación de las métricas de evaluación. Uno de los

errores detectados más relevantes afectaba al cálculo del *spread-skill ratio* en versiones anteriores de la librería de WeatherBench-X, ya que los términos de la ecuación (6.17) estaban intercambiados, lo que generaba valores muy poco interpretables. Este fallo fue identificado y verificado mediante comprobaciones manuales, aunque ya había sido detectado y corregido por los desarrolladores de WeatherBench-X en versiones recientes de la librería.

- **ED6:** Capacidad para tener un conocimiento profundo de los principios fundamentales y modelos utilizados en Ciencia de Datos, particularmente las relacionadas con el análisis, predicción y prospectiva de grandes volúmenes de datos.

  En este proyecto no solo se ha realizado un estudio en profundidad del modelo SeaCast, sino también de las técnicas y componentes empleados durante su fase de entrenamiento, así como de los modelos base sobre los que se construye. De la misma forma, se estudiaron los principios básicos correspondientes a las técnicas de aprendizaje por conjuntos y de las redes neuronales de grafo. La evaluación en fase de prueba involucró el uso de métricas probabilísticas y deterministas, cuya comprensión fue fundamental para realizar un análisis detallado de los resultados.

Apéndice C

Objetivos de desarrollo sostenible

En este apéndice se define la relación de este Trabajo de Fin de Grado con los ODS propuestos por la Organización de las Naciones Unidas. Específicamente, la Tabla C.1 presenta en una escala de 0 al 3 (de menor a mayor), la vinculación de cada ODS a nivel individual con el desarrollo de este proyecto. Cabe destacar que algunos objetivos muestran una relación significativa, dada la importancia de la predicción oceanográfica para su cumplimiento.

El ODS 3 (Salud y Bienestar) es una consecuencia directa de la relación del estudio con el cambio climático. La monitorización de los océanos para su conservación influye levemente en la salud humana.

Para el ODS 6 (Agua limpia y saneamiento) la relación es algo más significativa, ya que, como se ha mencionado, la predicción oceanográfica es relevante para conocer la disponibilidad de agua y su estado. Esto se relaciona también con el ODS 15 (Vida de ecosistemas terrestres).

Respecto al ODS 9 (Industria, Innovación e Infraestructuras), la puntuación es la más alta porque se emplean técnicas recientes de aprendizaje automático para intentar encontrar alternativas a la predicción numérica clásica.

En relación al ODS 13 (Acción por el clima), la puntuación vuelve a ser la máxima porque el reconocimiento y predicción de valores extremos o variaciones significativas permite evaluar los efectos de la contaminación y acción humana en el medio natural.

Finalmente, para el ODS 14 (Vida submarina), nuevamente la puntuación es alta por razones similares al ODS 13. El trabajo con variables oceanográficas es clave para conocer la adecuación del hábitat marino para la vida.

ODS	**Nivel de relación**			
	0	**1**	**2**	**3**
1. Fin de la Pobreza	**X**			
2. Hambre cero	**X**			
3. Salud y Bienestar		**X**		
4. Educación de calidad	**X**			
5. Igualdad de género	**X**			
6. Agua limpia y saneamiento			**X**	
7. Energía asequible y no contaminante	**X**			
8. Trabajo decente y crecimiento económico	**X**			
9. Industria, Innovación e Infraestructuras				**X**
10. Reducción de las desigualdades	**X**			
11. Ciudades y comunidades sostenibles	**X**			
12. Producción y consumo sostenibles	**X**			
13. Acción por el clima				**X**
14. Vida submarina				**X**
15. Vida de ecosistemas terrestres		**X**		
16. Paz, justicia e instituciones sólidas	**X**			
17. Alianzas para lograr objetivos	**X**			

Tabla C.1: Tabla con la valoración del grado de relación del TFG con los ODS, según una escala de 0 a 3 (no procede, bajo, medio, alto).

Acrónimos

AIFS Artificial Intelligence Forecasting System. 7, 8, 29, 93

API Application Programming Interface. 23, 25, 26, 30–32

C3S Copernicus Climate Change Service. 25

CMEMS Copernicus Marine Service. 10, 17, 18, 24, 25, 28–30, 32, 53

CRISP-DM Cross-Industry Standard Process for Data Mining. iv, 18–20

CRPS Continuous Ranked Probability Score. v, 9, 20, 66–69, 82, 83, 85, 86, 88, 89, 91

ECMWF European Centre for Medium-Range Weather Forecasts. 7–9, 25, 29, 64, 65

GNN Graph Neural Network. 1, 2, 4, 6, 9–12, 16, 17, 19, 23, 38, 42, 43, 45, 46, 48, 49, 64, 72, 91, 93

GPU Graphics Processing Unit. 7, 54, 92, 100

IFS Integrated Forecasting System. 8, 64

MLP Perceptrón Multicapa. 11, 15, 46–50, 72

MSE Mean Squared Error. 41, 42

NEMO Nucleus for European Modelling of the Ocean. 10, 65

NetCDF Network Common Data Form. 30–32, 34, 54

NOAA National Oceanic and Atmospheric Administration. 26

ODS Objetivos de Desarrollo Sostenible. 1, 6, 102

ONU Organización de las Naciones Unidas. 6

RMSE Root Mean Square Error. v, vii, 8, 20, 66, 68–70, 72, 73, 78–81, 83–85, 87–89, 93

SPP Stochastically Perturbed Parametrisations. 65

SPPT Stochastically Perturbed Parametrisation Tendencies. 64, 65

SST Sea Surface Temperature. iv, 12, 18, 24, 30, 31, 35, 39, 45, 52, 81

URL Uniform Resource Locator. 32, 33

Glosario

ataques de adversario Acciones maliciosas dirigidas a alterar las características o estructura de un grafo con el objetivo de influir en las predicciones del modelo de manera engañosa. 16

batimetría Estudio de la profundidad y la topografía del fondo de cuerpos de agua, como océanos, ríos y lagos, en relación al nivel del mar. 24, 26, 28

codificador Componente de un modelo que transforma las entradas a un espacio latente de representación, capturando las características más relevantes de los datos. 9, 38, 42, 43, 45, 48, 49

dataset Conjunto de datos. 30

decodificador Componente de un modelo que convierte la representación latente a una salida interpretable o esperada. 9, 12, 38, 42, 43, 45, 48, 49

desviación estándar Medida de dispersión que indica cuánto se alejan los datos numéricos de su media. Se calcula como la raíz cuadrada de la varianza, que es el promedio de los cuadrados de las diferencias entre cada dato y la media. 30, 42, 52, 58, 59, 64, 68, 75, 82, 83, 87–89, 94, 98

economía azul Conjunto de actividades socio-económicas directa o indirectamente relacionadas con el medio marino y su uso sostenible. 1, 6

endpoint (API) Punto de acceso específico de una API relacionado a un recurso o servicio. 26

gemelo digital Modelo virtual que replica el comportamiento o estado de un sistema físico para realizar simulaciones, pruebas o tareas de mantenimiento. 10

mecanismos de atención Ayudan a los modelos de aprendizaje automático a centrarse en las partes más relevantes de la entrada para una determinada tarea. 7

perceptrón multicapa Red neuronal compuesta por varias capas densamente conectadas entre sí. Esta estructura incluye una capa inicial que recibe los datos, una o más capas intermedias que procesan la información, y una capa final que genera la salida, permitiendo así modelar relaciones complejas entre entradas y salidas. 11

pooling Operación que reduce la dimensionalidad de una representación agrupando elementos mediante funciones de agregación como máximo, promedio o suma. 11

procesador Módulo encargado de operar sobre la representación latente de los datos. 9, 38, 42, 43, 45, 48, 72

sesgo Error sistemático entre la predicción y el valor real. En aprendizaje automático, se relaciona con modelos demasiado simples que causan subajuste. v, 15, 20, 70, 79, 80

sobreajuste Fenómeno en el que un modelo se ajusta excesivamente a un conjunto de datos limitado, lo que resulta en una pobre capacidad de generalización ante nuevos datos. 13, 14, 16, 36, 71

sobremuestreo Operación que aumenta el número de píxeles de una imagen mediante técnicas de interpolación. Se aumenta tanto la resolución como el tamaño de la imagen. 28, 32

subajuste Fenómeno donde un modelo es demasiado simple, dando un rendimiento bajo tanto en los datos de entrenamiento como en nuevos. 106

submuestreo Operación que reduce el número de píxeles en una imagen, disminuyendo su tamaño y resolución espacial. 28, 33

tasa de aprendizaje Hiperparámetro de las redes neuronales que controla la velocidad a la que se modifican los pesos durante el entrenamiento. 71, 72

transformer Es un modelo de aprendizaje automático que se basa en mecanismos de atención para captar las dependencias entre la entrada y la salida. Se caracteriza por la parelización en el entrenamiento, a diferencia de modelos anteriores que eran secuenciales. 8, 64

validación cruzada Técnica que consiste en dividir el conjunto de datos en varios subconjuntos para entrenar y evaluar el modelo de forma iterativa, permitiendo obtener una estimación más fiable y generalizable de su desempeño. 21, 36

varianza Medida estadística que indica el grado de dispersión de un conjunto de datos respecto a su media, calculada como el promedio de los cuadrados de las desviaciones respecto a la media. En aprendizaje automático, refleja la sensibilidad de un modelo a las variaciones en los datos de entrenamiento, donde una varianza alta puede provocar sobreajuste. 14, 42, 57, 58, 68–70, 84

Bibliografía

[Arístegui et al., 2009] Arístegui, J., Barton, E. D., Álvarez Salgado, X. A., Santos, A. M. P., Figueiras, F. G., Kifani, S., Hernández-León, S., Mason, E., Machú, E., and Demarcq, H. (2009). Sub-regional ecosystem variability in the canary current upwelling. *Progress in Oceanography*, 83(1):33–48. Eastern Boundary Upwelling Ecosystems: Integrative and Comparative Approaches.

[Barton et al., 1998] Barton, E., Arístegui, J., Tett, P., Cantón, M., García-Braun, J., Hernández-León, S., Nykjaer, L., Almeida, C., Almunia, J., Ballesteros, S., Basterretxea, G., Escánez, J., García-Weill, L., Hernández-Guerra, A., López-Laatzen, F., Molina, R., Montero, M., Navarro-Pérez, E., Rodríguez, J., van Lenning, K., Vélez, H., and Wild, K. (1998). The transition zone of the canary current upwelling region. *Progress in Oceanography*, 41(4):455–504.

[Battaglia et al., 2016] Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., et al. (2016). Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29.

[Battaglia et al., 2018] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[Bi et al., 2023a] Bi, K., Zhang, C., Wang, Y., Jiang, L., Jin, X., Wang, Y., et al. (2023a). Accurate medium-range global weather forecasting with 3d neural networks. *Nature*, 620(7973):529–534.

[Bi et al., 2023b] Bi, K., Zhang, C., Wang, Y., Jiang, L., Jin, X., Wang, Y., et al. (2023b). Pangu-weather: A deep learning weather forecasting model [github repository].

[Brenowitz et al., 2025] Brenowitz, N. D., Cohen, Y., Pathak, J., Mahesh, A., Bonev, B., Kurth, T., Durran, D. R., Harrington, P., and Pritchard, M. S. (2025). A practical probabilistic benchmark for ai weather models. *Geophysical Research Letters*, 52(7):e2024GL113656.

[C3S, 2025a] C3S (2025a). About us - copernicus climate change service.

[C3S, 2025b] C3S (2025b). Climate datasets - copernicus climate change service.

[C3S, 2025c] C3S (2025c). Climate reanalysis overview.

[Chapman et al., 2000] Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., and Wirth, R. (2000). Crisp-dm 1.0: Step-by-step data mining guide.

[Chen et al., 2024] Chen, L., Zhong, X., Li, H., Wu, J., Lu, B., Chen, D., Xie, S.-P., Wu, L., Chao, Q., Lin, C., et al. (2024). A machine learning model that outperforms conventional global subseasonal forecast models. *Nature Communications*, 15(1):6425.

[CMEMS, 2022] CMEMS (2022). Glorys12v1 global ocean eddy-resolving reanalysis (1/12° resolution, 50 vertical levels). DOI: 10.48670/moi-00021.

[CMEMS, 2023] CMEMS (2023). European north west shelf/iberia biscay irish seas - high resolution L4 sea surface temperature reprocessed. DOI: 10.48670/moi-00153.

[CMEMS, 2023] CMEMS (2023). Observer: How copernicus marine supports the european digital twin ocean.

[CMEMS, 2024a] CMEMS (2024a). European seas mean dynamic topography. DOI: 10.48670/mds-00337.

[CMEMS, 2024b] CMEMS (2024b). Product user manual for mediterranean sea physical physical reanalysis product (cmems-med-pum-006-004).

[CMEMS, 2024c] CMEMS (2024c). Which levels are used for data processing?

[CMEMS, 2025a] CMEMS (2025a). About copernicus.

[CMEMS, 2025b] CMEMS (2025b). What are the different data types provided by copernicus marine?

[Couairon et al., 2024] Couairon, G., Singh, R., Charantonis, A., Lessig, C., and Monteleoni, C. (2024). Archesweather & archesweathergen: a deterministic and generative model for efficient ml weather forecasting. *arXiv preprint arXiv:2412.12971*.

[CW3E, 2025] CW3E (2025). Toward calibrated ensembles of neural weather model forecasts.

[de Burgh-Day and Leeuwenburg, 2023] de Burgh-Day, C. O. and Leeuwenburg, T. (2023). Machine learning for numerical weather and climate modelling: a review. *Geoscientific Model Development*, 16(22):6433–6477.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

[Dietterich, 2000] Dietterich, T. G. (2000). Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer.

[Drillet et al., 2025] Drillet, Y., Martin, M., Fujii, Y., Chassignet, E., and Ciliberti, S. (2025). Core services: an introduction to global ocean forecasting. *State of the Planet*, 5:2.

[Duan et al., 2024] Duan, R., Yan, C., Wang, J., and Jiang, C. (2024). Graph ensemble neural network. *Information Fusion*, 110:102461.

[Dutta, 2025] Dutta, N. (2025). Analytics vidhya - mixture of experts models: Explained.

[ECMWF, 2012] ECMWF (2012). The ECMWF ensemble prediction system. Technical report, European Centre for Medium-Range Weather Forecasts.

[ECMWF, 2025a] ECMWF (2025a). ECMWF's Ensemble AI Forecasts Become Operational. Technical report, European Centre for Medium-Range Weather Forecasts.

[ECMWF, 2025b] ECMWF (2025b). What are grib files and how can i read them?

[Esri, 2025a] Esri (2025a). Arcgis geospatial platform overview.

[Esri, 2025b] Esri (2025b). What is netcdf data?

[Fortin et al., 2014] Fortin, V., Abaza, M., Anctil, F., and Turcotte, R. (2014). Why should ensemble spread match the rmse of the ensemble mean? *Journal of Hydrometeorology*, 15(4):1708–1713.

[Gneiting et al., 2005] Gneiting, T., Raftery, A. E., Westveld III, A. H., and Goldman, T. (2005). Calibrated probabilistic forecasting using ensemble model output statistics and minimum crps estimation. *Monthly weather review*, 133(5):1098–1118.

[González et al., 2020] González, S., García, S., Del Ser, J., Rokach, L., and Herrera, F. (2020). A practical tutorial on bagging and boosting based ensembles for machine learning: Algorithms, software tools, performance study, practical perspectives and opportunities. *Information Fusion*, 64:205–237.

[Google Research, 2024] Google Research (2024). Weatherbench 2 – probabilistic scores. `https://sites.research.google/gr/weatherbench/probabilistic-scores/`.

[Goyal et al., 2017] Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. (2017). Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

[Gustavson, 2005] Gustavson, S. (2005). Simplex noise demystified. *Linköping University, Linköping, Sweden, Research Report*, 1(2):6.

[Hersbach et al., 2025] Hersbach, H., Bell, B., Berrisford, P., Biavati, G., Horányi, A., Sabater, M., J., Nicolas, J., Peubey, C., Radu, R., Rozum, I., Schepers, D., Simmons, A., Soci, C., Dee, D., and Thépaut, J.-N. (2025). Era5 hourly data on single levels from 1940 to present. Copernicus Climate Change Service (C3S) Climate Data Store (CDS). DOI: 10.24381/cds.adbb2d47.

[Holmberg et al., 2024] Holmberg, D., Clementi, E., and Roos, T. (2024). Regional ocean forecasting with hierarchical graph neural networks. *arXiv preprint arXiv:2410.11807*.

[Hoteit et al., 2025] Hoteit, I., Chassignet, E., and Bell, M. (2025). Improving accuracy and providing uncertainty estimations: ensemble methodologies for ocean forecasting. *State of the Planet*, 5-opsr:21.

[Kalirane, 2025] Kalirane, M. (2025). Analytics vidhya - ensemble learning methods: Bagging, boosting and stacking.

[Keisler, 2022] Keisler, R. (2022). Forecasting global weather with graph neural networks. *arXiv preprint arXiv:2202.07575*.

[Kochkov et al., 2024] Kochkov, D., Yuval, J., Langmore, I., Norgaard, P., Smith, J., Mooers, G., Klöwer, M., Lottes, J., Rasp, S., Düben, P., et al. (2024). Neural general circulation models for weather and climate. *Nature*, 632(8027):1060–1066.

[Lam et al., 2023] Lam, R., Sanchez-Gonzalez, A., Willson, M., Wirnsberger, P., Fortunato, M., Alet, F., Ravuri, S., Ewalds, T., Eaton-Rosen, Z., Hu, W., et al. (2023). Learning skillful medium-range global weather forecasting. *Science*, 382(6677):1416–1421.

[Liang et al., 2023] Liang, S., Zhao, A., Qin, M., Hu, L., Wu, S., Du, Z., and Liu, R. (2023). A graph memory neural network for sea surface temperature prediction. *Remote Sensing*, 15(14):3539.

[Loshchilov and Hutter, 2016] Loshchilov, I. and Hutter, F. (2016). Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

[Loshchilov and Hutter, 2017] Loshchilov, I. and Hutter, F. (2017). Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

[Lyon, 2014] Lyon, A. (2014). Why are normal distributions normal? *The British Journal for the Philosophy of Science*.

[Martinez et al., 2021] Martinez, I., Viles, E., and G. Olaizola, I. (2021). Data science methodologies: Current challenges and future approaches. *Big Data Research*, 24:100183.

[NOAA, 2022a] NOAA (2022a). Etopo 2022 30 arc-second global relief model. DOI: 10.25921/fd45-gt74.

[NOAA, 2022b] NOAA (2022b). Etopo 2022 user guide.

[NOAA, 2023] NOAA (2023). Where the water meets the land - a coastal digital elevation model framework.

[NOAA, 2025a] NOAA (2025a). Climate modeling.

[NOAA, 2025b] NOAA (2025b). What is upwelling?

[Novellino et al., 2024] Novellino, A., Arnaud, A., Schiller, A., and Wan, L. (2024). End user applications for ocean forecasting: present status description. *State of the Planet Discussions*, 2024:1–6.

[NumPy Developers, 2025] NumPy Developers (2025). numpy.random.normal — numpy manual.

[Ollinaho et al., 2017] Ollinaho, P., Lock, S.-J., Leutbecher, M., Bechtold, P., Beljaars, A., Bozzo, A., Forbes, R. M., Haiden, T., Hogan, R. J., and Sandu, I. (2017). Towards process-level representation of model uncertainties: stochastically perturbed parametrizations in the ecmwf ensemble. *Quarterly Journal of the Royal Meteorological Society*, 143(702):408–422.

[Omari and Figueiras-Vidal, 2015] Omari, A. and Figueiras-Vidal, A. R. (2015). Post-aggregation of classifier ensembles. *Information Fusion*, 26:96–102.

[Oskarsson et al., 2023] Oskarsson, J., Landelius, T., and Lindsten, F. (2023). Graph-based neural weather prediction for limited area modeling. *arXiv preprint arXiv:2309.17370*.

[Pathak et al., 2022] Pathak, J., Subramanian, S., Harrington, P., Raja, S., Chattopadhyay, A., Mardani, M., Kurth, T., Hall, D., Li, Z., Azizzadenesheli, K., et al. (2022). Fourcastnet: A global data-driven high-resolution weather model using adaptive fourier neural operators. *arXiv preprint arXiv:2202.11214*.

[Perlin, 1985] Perlin, K. (1985). An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296.

[Price et al., 2023] Price, I., Sanchez-Gonzalez, A., Alet, F., Andersson, T. R., El-Kadi, A., Masters, D., Ewalds, T., Stott, J., Mohamed, S., Battaglia, P., et al. (2023). Gencast: Diffusion-based ensemble forecasting for medium-range weather. *arXiv preprint arXiv:2312.15796*.

[Ramachandran et al., 2017] Ramachandran, P., Zoph, B., and Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[Rasp et al., 2020] Rasp, S., Dueben, P. D., Scher, S., Weyn, J. A., Mouatadid, S., and Thuerey, N. (2020). Weatherbench: a benchmark data set for data-driven weather forecasting. *Journal of Advances in Modeling Earth Systems*, 12(11):e2020MS002203.

[Rasp et al., 2024] Rasp, S., Hoyer, S., Merose, A., Langmore, I., Battaglia, P., Russell, T., Sanchez-Gonzalez, A., Yang, V., Carver, R., Agrawal, S., et al. (2024). Weatherbench 2: A benchmark for the next generation of data-driven global weather models. *Journal of Advances in Modeling Earth Systems*, 16(6):e2023MS004019.

[Rout, 2024] Rout, S. (2024). Interview node - ensemble learning techniques: Boosting, bagging, and stacking explained.

[Sagi and Rokach, 2018] Sagi, O. and Rokach, L. (2018). Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1249.

[Samuel, 1959] Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229.

[Sanchez-Lengeling et al., 2021] Sanchez-Lengeling, B., Reif, E., Pearce, A., and Wiltschko, A. B. (2021). A gentle introduction to graph neural networks. *Distill*, 6(9):e33.

[Shearer, 2000] Shearer, C. (2000). The CRISP-DM model: the new blueprint for data mining. *Journal of data warehousing*, 5(4):13–22.

[Veitch et al., 2025] Veitch, J., Alvarez-Fanjul, E., Capet, A., Ciliberti, S., Cirano, M., Clementi, E., Davidson, F., el Serafy, G., Franz, G., Hogan, P., Joseph, S., Liubartseva, S., Miyazawa, Y., Regan, H., and Spanoudaki, K. (2025). A description of ocean forecasting applications around the globe. *State of the Planet*, 5-opsr:6.

[Vervatis et al., 2025] Vervatis, V. D., De Mey-Frémaux, P., Karagiorgos, J., Lemieux-Dudon, B., Ayoub, N. K., and Sofianos, S. (2025). Regional ocean model uncertainties using stochastic parameterizations and a global atmospheric ensemble. *Ocean Modelling*, 194:102501.

[Vigier, 2018] Vigier, P. (2018). Perlin noise implementation in numpy [github repository].

[WeatherBenchX, 2025] WeatherBenchX (2025). Weatherbench-x documentation: Metrics api reference.

[Wei et al., 2023] Wei, W., Qiao, M., and Jadav, D. (2023). Gnn-ensemble: Towards random decision graph neural networks. In *2023 IEEE International Conference on Big Data (BigData)*, pages 956–965. IEEE.

[Wilks, 2011] Wilks, D. (2011). On the reliability of the rank histogram. *Monthly Weather Review*, 139(1):311–316.

[Wong et al., 2023] Wong, Z. H., Yue, L., and Yao, Q. (2023). Ensemble learning for graph neural networks. *arXiv preprint arXiv:2310.14166*.

[Wu et al., 2020] Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., and Yu, P. S. (2020). A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24.

[Xu et al., 2024] Xu, P., Xu, S., Shi, K., Ou, M., Zhu, H., Xu, G., Gao, D., Li, G., and Zhao, Y. (2024). Prediction of water temperature based on graph neural network in a small-scale observation via coastal acoustic tomography. *Remote Sensing*, 16(4):646.

[Yang et al., 2023] Yang, Y., Lv, H., and Chen, N. (2023). A survey on ensemble learning under the era of deep learning. *Artificial Intelligence Review*, 56(6):5545–5589.

[You et al., 2021] You, J., Gomes-Selman, J. M., Ying, R., and Leskovec, J. (2021). Identity-aware graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 10737–10745.

[Zamo and Naveau, 2018] Zamo, M. and Naveau, P. (2018). Estimation of the continuous ranked probability score with limited information and applications to ensemble weather forecasts. *Mathematical Geosciences*, 50(2):209–234.

[Zhang et al., 2025] Zhang, Q., Gao, X., Wang, H., Yiu, S. M., and Yin, H. (2025). Efficient traffic prediction through spatio-temporal distillation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 1093–1101.

[Zhou et al., 2002] Zhou, Z.-H., Wu, J., and Tang, W. (2002). Ensembling neural networks: Many could be better than all. *Artificial Intelligence*, 137(1):239–263.

ULPGC
Universidad de
Las Palmas de
Gran Canaria

ESCUELA DE INGENIERÍA
INFORMÁTICA