

**UNIÓN DE ASOCIACIONES DE INGENIEROS
TÉCNICOS INDUSTRIALES Y GRADUADOS EN
INGENIERÍA DE LA RAMA INDUSTRIAL DE ESPAÑA
(UAIIE)**

“CONVOCATORIA 2024”

**IX PREMIO NACIONAL DE INICIACIÓN A LA INVESTIGACIÓN
TECNOLÓGICA**

**Sistema de predicción aerodinámica para automóviles
basado en redes neuronales convolucionales**

AUTORES:

Guillermo Castellano Ibarrola
Álvaro Megía Caballero
Daniel Rouco Morcillo

BLOQUE TEMÁTICO:

Inteligencia Artificial

NIVEL EDUCATIVO:

2º Bachillerato
Categoría B

COORDINADOR:

Blanca López Fernández
Cristina María Jiménez Leal

MARZO 2024

Índice

1. Introducción	1
2. Objetivos	2
3. Metodología	2
4.1. Herramientas empleadas	2
4.2. Procedimientos	2
I. Recopilación de datos	3
II. Importación de librerías	4
III. Preprocesamiento de datos	5
IV. Construcción y compilación del modelo	6
V. Entrenamiento	9
5. Visualización y análisis de resultados	10
6. Conclusiones	17
7. Futuras líneas de investigación	17
8. Bibliografía	18

1. Introducción

La dinámica de fluidos, disciplina que estudia el comportamiento entre fluidos y cuerpos en movimiento, es una de las ramas de la física que más se ha desarrollado en las últimas décadas. En gran medida, este acelerado desarrollo se debe a la dinámica de fluidos computacional (CFD), la cual permite realizar análisis pormenorizados del flujo de los fluidos desde un ordenador. No obstante, esta todavía supone un alto coste tanto computacional como especialmente de tiempo. Similarmente, la mayor eficiencia derivada de la inteligencia artificial, además de haberse traducido en su reciente auge, supone y ha supuesto una revolución en la metodología de resolución de problemas. En este sentido, diversas publicaciones científicas han mostrado avances prometedores en la aplicación de sistemas basados en inteligencia artificial a la predicción del flujo de los fluidos, reduciendo drásticamente el coste computacional. De manera más significativa, los sistemas de predicción basados en inteligencia artificial suponen una reducción del tiempo requerido de un orden de magnitud 1/160 frente a las simulaciones de CFD, alcanzando una tasa de error media de tan solo 5,25% (Xiaofeng Cao, 2023). Todo lo que mejora, en última instancia, la eficiencia y agiliza la toma de decisiones.

Por otro lado, el 25 de septiembre de 2015 los 193 estados miembros de las Naciones Unidas acordaron los Objetivos de Desarrollo Sostenible (ODS), objetivos que pretenden abordar los principales desafíos globales. En ellos se hace un especial hincapié en la protección del medioambiente, de ahí, que reconociendo la excepcionalidad de la situación y el papel fundamental de una buena optimización de los recursos para reducir el impacto climático de la industria, este proyecto se refiera a la meta 9.4, «modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales» (Naciones Unidas, s.f.).

Para ello, en el presente proyecto se presenta un sistema de predicción de los campos de fluidos y coeficientes de resistencia aerodinámica de automóviles, elementos clave en su diseño aerodinámico, basado en redes neuronales convolucionales que coadyuve a una optimización en el uso de los recursos.

2. Objetivos

Este proyecto tiene como objetivo central el desarrollo de un sistema de predicción basado en redes neuronales convolucionales para optimizar y acelerar el estudio del comportamiento de los fluidos al entrar en contacto con un automóvil.

3. Metodología

La metodología procedimental de este proyecto se basa en la predicción de los campos de velocidad y presión de vehículos a partir de su función de distancia con signo (SDF). Por otro lado, se predice el coeficiente de *drag* de automóviles a partir de distintas imágenes de su perfil. Para ello, se diseñan dos algoritmos diferentes haciendo uso de redes neuronales convolucionales. Los [códigos de programación](#) desarrollados se encuentran en la *Figura 1*.



Figura 1. Código de programación

Fuente: Elaboración propia

4.1. Herramientas empleadas

- *Python*. Python, es el lenguaje de programación empleado en su versión 3.11. Además de ser de código abierto y alto nivel, es conocido por su sintaxis clara, amplio soporte de librerías, legibilidad y versatilidad.
- *PyCharm*. PyCharm, desarrollado por JetBrains, es el entorno de desarrollo integrado (IDE) utilizado en el proyecto al haber sido creado específicamente para Python. Además, ofrece una amplia gama de características y herramientas que facilitan y agilizan el proceso de programación. Entre ellas destacan el editor de código inteligente, el gestor de proyectos y la depuración.
- *GitLab*. GitLab es una plataforma de desarrollo de software integrada en PyCharm y basada en web que se centra en el control de versiones a través de Git, lo que permite recuperar y administrar cambios en el código fuente.

4.2. Procedimientos

Se siguieron los pasos descritos en la *Figura 2* con cada uno de los dos modelos.

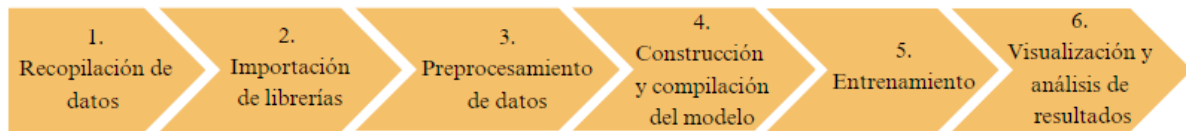


Figura 2. Procedimientos seguidos en el desarrollo de ambos modelos

Fuente: Elaboración propia

I. Recopilación de datos

Campos de velocidad y presiones

Las imágenes empleadas se recuperaron de una [base de datos](#) de libre acceso de formato '.npz' publicada en el año 2023 (Nicolas Rosset, 2023). La base de datos está compuesta por 2013 casos o automóviles representados en 2 dimensiones, compuestos a su vez por 4 clases o canales: función de distancia con signo (SDF, *Signed Distance Function*), campo de presiones y campos de velocidad en el eje X y en el eje Y. A continuación, se describe brevemente cada canal:



Figura 3. SDF



Figura 4. Campo de presiones

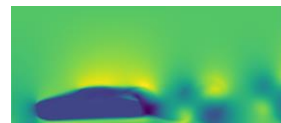


Figura 5. Campo de velocidad: eje X

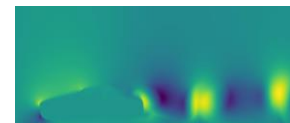


Figura 6. Campo de velocidad: eje Y

1. SDF (*Figura 3*): función matemática que asigna una distancia con un signo, en función de su posición, a cada punto en el espacio con respecto a una superficie o forma geométrica, describe la geometría del vehículo en dos dimensiones (Park, Florence, Straub, Newcombe, & Lovegrove, 2019).
2. Campo de presiones (*Figura 4*): proporciona información sobre la fuerza y distribución de las presiones en la superficie del vehículo, lo cual tiene un impacto directo en la generación de sustentación, carga aerodinámica y *drag*.
3. Campo de velocidad (*Figuras 5 y 6*): ofrece información sobre la distribución de velocidades y los patrones de flujo en la superficie del vehículo tanto en el eje X ('vel1') como en el Y ('vel2'). Por consiguiente, permite visualizar cómo el flujo de aire se comporta alrededor del automóvil, cómo se generan y disipan las corrientes de aire, y cómo varía la velocidad en diferentes puntos. Esto posibilita comprender y analizar el comportamiento aerodinámico del vehículo.

Los campos de presión y velocidad asociados a cada SDF son el resultado de una serie de simulaciones CFD realizadas a partir de las siguientes condiciones prefijadas en relación con el aire: una densidad de 1 kg/m^3 , una viscosidad de 0 y una velocidad de 54 km/h (Nicolas Rosset, 2023).

Coeficiente de drag

Por otro lado, para la predicción del coeficiente de *drag*, valor que determina la eficiencia aerodinámica de un automóvil, se recuperaron los datos de una [base](#) de libre acceso perteneciente al Massachusetts Institute of Technology publicada en agosto de 2023 (Binyang Song, 2023). Esta base contiene 9.760 vehículos en 3 dimensiones en formato '.obj' y sus coeficientes de *drag* asociados en un archivo '.xlsx'. La condición prefijada para el cálculo de los coeficientes es una velocidad del aire de 40 km/h (Binyang Song, 2023).

II. Importación de librerías

En segundo lugar, se importaron una serie de librerías, archivos importables que contienen código preescrito y funcionalidades previamente implementadas para agilizar la programación, resultando en una mayor eficacia en la ejecución del código. Las empleadas durante el proyecto en ambos modelos se exponen a continuación:

- Keras: ampliamente empleada en aprendizaje profundo, simplifica la construcción, entrenamiento y evaluación de redes neuronales o en este proyecto, redes neuronales convolucionales. Cuenta con una interfaz de alto nivel y modular, lo que permite desarrollar modelos eficientemente (Keras, s.f.).
- Matplotlib: utilizada para crear visualizaciones y gráficos con un alto grado de personalización (Matplotlib, s.f.). En este proyecto, empleada para la visualización de las imágenes y resultados de forma gráfica.
- NumPy: especializada en el análisis de grandes volúmenes de datos, es empleada para realizar cálculos numéricos y distintas operaciones matemáticas con mayor eficiencia (NumPy, s.f.). En el caso de este proyecto se emplea para el tratamiento de las imágenes en forma matricial en el primer modelo.
- Os: Empleada en la creación y manipulación de archivos (Python, s.f.).
- Pillow: centrada en el procesamiento de imágenes, utilizada para cargar, manipular, guardar y procesar imágenes en diferentes formatos.
- SciencePlots: permite adaptar el estilo de los gráficos creados por Matplotlib a los utilizados en artículos científicos. En el proyecto, se eligió el estilo *science*.
- MayaVi: centrada en la visualización de gráficos y representación de datos en tres dimensiones. En este proyecto, utilizada en el preprocesamiento de los archivos '.obj' pertenecientes a la base de datos del segundo modelo.

III. **Preprocesamiento de datos**

Se preprocesaron los archivos de las dos bases de datos, obteniendo las imágenes que sirven de entrada para ambos modelos respectivamente.

Campos de velocidad y presiones

Primeramente, al presentar la base de datos un formato '.npz' (conjunto de matrices), las imágenes solo se pudieron visualizar a través de la librería Matplotlib. Posteriormente, se recuperaron las 8052 imágenes con un formato '.png' y se las clasificó de acuerdo con su función (*training*, *validation* o *test*) y clase o canal respectivo mediante la librería Pillow. Para ello, se las asignó un nombre siguiendo el criterio anterior: 'Canal_Función_Númerolimagen', por ejemplo, 'Pressure_Training_1788'. A continuación, se separaron las imágenes por clases y en última instancia por su función, resultando en 12 directorios con el siguiente nombre 'canal_images_función', por ejemplo, 'vel1_images_validation'. El número de imágenes que se asignó a cada función se hizo siguiendo el consenso general en *deep learning*, destinando un 63% de las imágenes (1277 / 2013) a *training*, un 20% (398 / 2013) a *validation* y un 17 % (338 / 2013) a *test*. Por último y mediante NumPy, se transformó cada imagen a un único *numpy array* para el entrenamiento del modelo.

Coefficiente de drag

En primer lugar, se simplificó y cambió el formato del archivo '.xlsx' (de nombre 'drag_coefficients', *Figura 7*) a '.csv' para facilitar su posterior procesamiento. Este consistía de dos columnas: 'file', contenía el nombre de los archivos, y 'Cd', su valor asociado. Por otro lado, ya que los archivos de la base de datos presentaban un formato '.obj', no válido como valor de entrada del modelo, mediante MayaVi se realizaron 6 fotos de cada vehículo desde distintos ángulos ((0, 0), (180, 180), (90, 90), (270, 90), (180, 90), (0, 90)), esto es, de cada uno de los 9760 archivos para suplir la falta de dimensionalidad y profundidad de una única imagen. Posteriormente, para reducir el coste computacional y complejidad del modelo, se unificaron las 6 imágenes de cada '.obj' en una sola (*Figura 8*), formando una imagen resultante de 240x270 píxeles y asignándola el mismo nombre que el de su archivo predecesor '.obj' (el mismo que en la columna 'file'). Esto se tradujo en 9760 archivos '.png' (*Figura 8*) que fueron divididos en 3 directorios atendiendo a su función. De ellas, un 63% (6150 / 9760) se destinaron al entrenamiento 'drag_images_training', un 20% (1975 / 9760) a validación 'drag_images_validation' y un 17% (1635 / 9760) a *test* 'drag_images_test'. Posteriormente, se crearon 3 *DataFrames* ('train_generator'

(entrenamiento), 'validation_generator' (validación) y 'test_generator' (test)) a partir de las imágenes y archivo '.csv'. Mediante estos, estructuras de datos bidimensionales organizadas en filas y columnas, se asoció directamente cada coeficiente de *drag* a su imagen, ya no solo a su nombre como previamente en el '.csv'. Igualmente, redujeron el coste computacional, optimizando el uso de la memoria.

file	Cd
105a77e198fd243eab07f3d736b6f705.png	0,350497596
105a77e198fd243eab07f3d736b6f705_aug.png	0,351037733
10716a366de708b8fac96522b26f7fd.png	0,412332333
10716a366de708b8fac96522b26f7fd_aug.png	0,328315667
1079efee042629d4ce28f0f1b509eda.png	0,341476952
1079efee042629d4ce28f0f1b509eda_aug.png	0,338112685
1089cbe82dc0e72133d7c9e122eec9b6.png	0,442198752
109567d7d55b8fe515a520abec2f04dd.png	0,473927044

Figura 7. Archivo 'drag_coefficients' (.xlsx)

Fuente: Elaboración propia

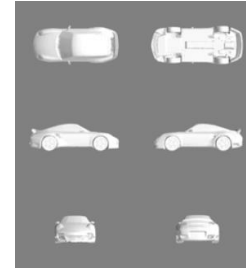


Figura 8. Entrada del segundo modelo

Fuente: Elaboración propia

IV. Construcción y compilación del modelo

Posteriormente, se diseñaron y construyeron dos modelos distintos, empleando en ambos casos la librería Keras. El primero, referido a la predicción de los campos de velocidad y presiones, y el segundo, a la predicción del coeficiente de *drag*.

Construcción del primer modelo: campos de velocidad y presiones

Para la predicción de los campos de velocidad y presiones se optó por una red neuronal convolucional con una estructura *U-Net* compuesta por un codificador y un decodificador (Figura 9). En primer lugar, el codificador, responsable de la extracción de las características de la geometría del vehículo y de la generación de patrones a partir de las mismas, recibe como *input* la función de distancia con signo. En cuanto a su estructura, consiste en varias capas ocultas, compuestas a su vez por capas convolucionales de un número de neuronas variables, que tienen como función la extracción de características, y por capas de agrupación máxima que reducen progresivamente el tamaño de las imágenes. Por otro lado, el decodificador se centra en la reconstrucción del perfil aerodinámico del vehículo a partir de las características extraídas por el codificador y en la predicción de los campos de presiones y velocidad. Consta de capas convolucionales y de capas en las que se concatenan, simétricamente, convoluciones anteriores, la primera convolución con la última, la segunda con la penúltima y así sucesivamente. Asimismo y a través de las convoluciones transpuestas, se incrementa el tamaño de las imágenes hasta los 256x600 píxeles. Finalmente, el valor de salida resultante son los campos de

presiones y velocidad. En esta ocasión, no se introdujeron capas completamente conectadas por incurrir estas en cálculos redundantes.

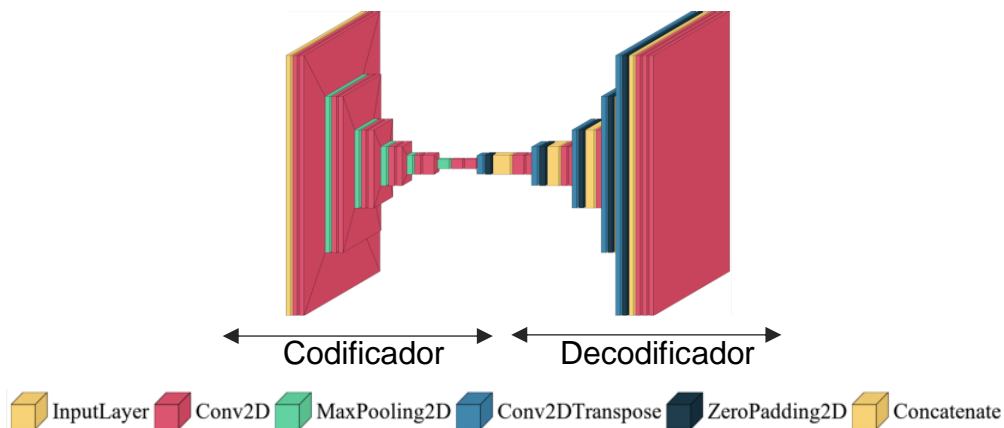


Figura 9. Arquitectura general del primer modelo

Fuente: Elaboración propia

Una vez definida la arquitectura, se eligieron las funciones de activación. A continuación, se exponen las que se tradujeron en unos mejores resultados.

- **Swish (Figura 10).** De manera algebraica se define como $f(x) = x \cdot \text{sigmoid}(\beta \cdot x) = \frac{x}{1+e^{-\beta x}}$, donde β puede ser una constante o un parámetro a optimizar. Asimismo, es continua, no lineal, y a diferencia de la mayoría de las funciones de activación no monótona. Si bien se traduce en unos mejores resultados, es computacionalmente muy costosa. Empleada en la última capa del modelo (Ramachandran, Zoph, & Le, 2017).
- **Mish (Figura 10).** Inspirada en *swish*, fue propuesta en el año 2019. Al igual que *swish* es no monótona, continua y no lineal. Más aún, es infinitamente diferenciable y computacionalmente muy costosa. Se define matemáticamente como $f(x) = x \cdot \tanh(\text{softplus}(x)) = x \cdot \tanh(\ln(1 + e^x))$. Se emplea en las capas convolucionales y de agrupación (Misra, 2020) (Shaw, 2022).

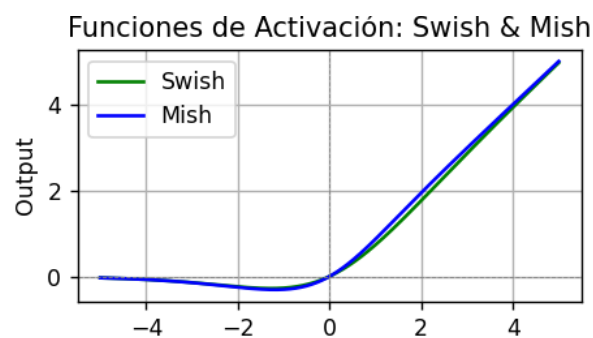


Figura 10. Funciones de activación *swish* y *mish*

Fuente: Elaboración propia

Construcción del segundo modelo: coeficiente de drag

Por otro lado, para la predicción del drag se diseñó una red neuronal convolucional (Figura 11) que toma como valor de entrada los *DataFrames* ya creados. Por lo que se refiere a su estructura, consta de una sucesión de capas convolucionales y de capas de agrupación máxima de número de neuronas variables responsables de la extracción las características aerodinámicas de los valores de entrada. Asimismo y a continuación, la capa *flatten* reduce la dimensionalidad de las salidas resultantes, ajustándola a las necesidades de la capa completamente conectada o *dense*. Esta última toma como valor de entrada la salida de la capa *flatten* y es responsable de producir el *output* definitivo. Igualmente, se intercaló una capa *dropout*, la cual inhibe un número de neuronas determinado de modo aleatorio, entre las completamente conectadas para evitar el sobreaprendizaje que se derivado de su sucesión.

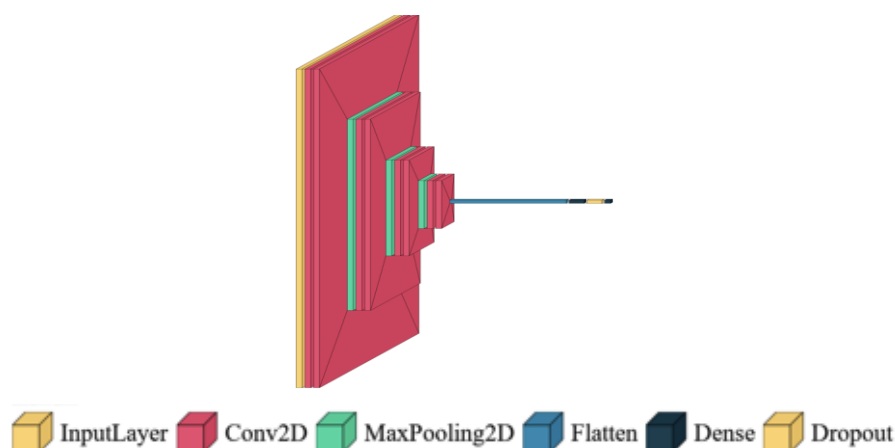


Figura 11. Arquitectura general del segundo modelo

Fuente: Elaboración propia

Como funciones de activación se emplearon las ya descritas *swish* y *mish*.

Compilación

Tras haber construido los modelos, se definieron los parámetros y funciones correspondientes a la fase de compilación. En primer lugar, se definió la tasa de aprendizaje inicial. A continuación, se eligió el error cuadrático medio (MSE) como función de pérdida a la vez que se monitorizó (únicamente en el primer modelo) la exactitud o *accuracy*, esto es, el número de predicciones correctas partido del número de predicciones totales. Por último, se escogió como optimizador *AdaMax*, una variante de *Adam* cuyo segundo momento se calcula siguiendo la norma del supremo, lo que resulta en una mayor estabilidad y mejores resultados en redes profundas.

V. Entrenamiento

Primeramente, se establecieron un conjunto de parámetros comunes a ambos modelos:

- Los datos de entrada y salida para el entrenamiento. En el primer modelo se fijó como valor de entrada el directorio 'sdf_images_training' y como salida los directorios 'vel1_images_training', 'vel2_images_training' y 'pressure_images_training'. Asimismo, se definieron los valores de entrada y salida de la validación, como entrada 'sdf_images_validation' y como salida 'vel1_images_validation', 'vel2_images_validation' y 'pressure_images_validation'. En el segundo modelo, se establecieron como valores de entrada las imágenes del *DataFrame* 'training_generator' y como salida los valores (coeficientes de *drag*) asociados. Similarmente, se fijaron las imágenes del *DataFrame* 'validation_generator' como entrada y los coeficientes de *drag* asociados como salida.
- El *batch size* o tamaño del lote, siendo este el número de ejemplos de entrenamiento que se utilizan por iteración, cuanto mayor es, más memoria se requiere. También se definió el *validation batch size*.
- Las *epochs*, épocas o iteraciones, esto es, el número de veces que el algoritmo ha procesado todo el conjunto de entrenamiento. De ahí, que su correcta elección determine en gran medida el desempeño del modelo. Se eligió un número de iteraciones totales de 100.
- *Verbose*, la cual controla la cantidad de información que se muestra durante el proceso de entrenamiento. Se estableció 'verbose=1', por lo cual se muestra una barra de evolución animada que indica el progreso de las épocas y las métricas de entrenamiento (*loss* y *accuracy*) en tiempo real.
- *Callbacks*, los objetos utilizados para personalizar el entrenamiento. Se definieron 4 *callbacks*: *BackUpAndRestore*, *CSVLogger*, *EarlyStopping* y *ReduceLROnPlateau*. *BackUpAndRestore* posibilita guardar los pesos del modelo, así como el estado del optimizador y tasa de aprendizaje durante el entrenamiento, lo que permite continuar con este en caso de una interrupción. Por otro lado, *CSVLogger* guarda la evolución de las métricas, *loss* y *accuracy*, durante el entrenamiento y validación en un archivo '.csv', permitiendo llevar un registro detallado del rendimiento del modelo en cada iteración. Más aun, *EarlyStopping* detiene automáticamente el entrenamiento del modelo cuando

la función de pérdida deja de mejorar, evitando el sobreaprendizaje o *overfitting*, ahorrando tiempo y recursos computacionales. Por último, mediante *ReduceLROnPlateau* se ajusta automáticamente la tasa de aprendizaje del modelo cuando el progreso del entrenamiento se estanca, evitando que este se quede atrapado en mínimos locales y ayudando a mejorar su convergencia. Finalmente y una vez entrenado, la arquitectura del modelo fue almacenada en formato '.json'. Igualmente, se guardaron los pesos definitivos en un archivo '.h5', posibilitando recuperar el modelo más adelante.

5. Visualización y análisis de resultados

Tras haber sido entrenados, se procedió a la visualización y análisis de los datos. Para ello, se emplearon las librerías Matplotlib y SciencePlots, mediante las cuales se extrajeron las gráficas correspondientes al error y exactitud frente a las iteraciones, así como los campos de presión y velocidad resultantes. De todos los procesos de entrenamiento se han elegido los dos más representativos de cada modelo.

Primer modelo: experimento 1

Se entrenó el modelo a partir de los hiperparámetros especificados en la *Tabla 1*. Asimismo, la arquitectura del modelo contó con 21 capas convolucionales, 5 capas de agrupación máxima, 5 capas de concatenación y 5 capas de convolución transpuesta.

Tabla 1. Parámetros del primer experimento del primer modelo

	Vel1	Vel2	Presión
Épocas (finales)	54	48	42
Tamaño del lote: entrenamiento	28	28	28
Tamaño del lote: validación	24	24	24
Tasa de aprendizaje inicial	10^{-4}	10^{-4}	10^{-4}
Tasa de aprendizaje final	10^{-8}	10^{-6}	10^{-6}
Nº de parámetros a entrenar	6.686.948	6.686.948	6.686.948

Posteriormente, se visualizó la evolución del error y exactitud del modelo durante el entrenamiento y fase de validación, frente a las iteraciones.

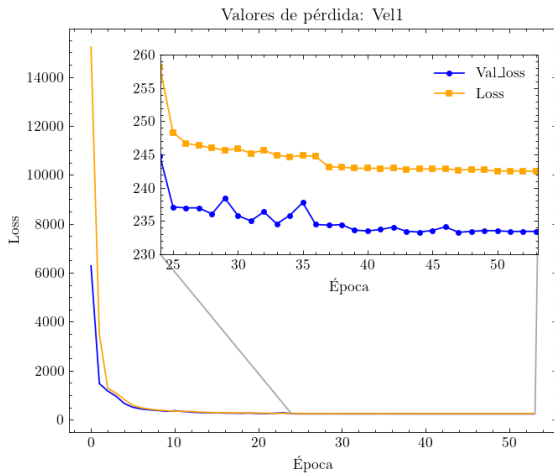


Figura 12. Valores de pérdida (Vel1) del primer modelo en su primer experimento

Fuente: Elaboración propia

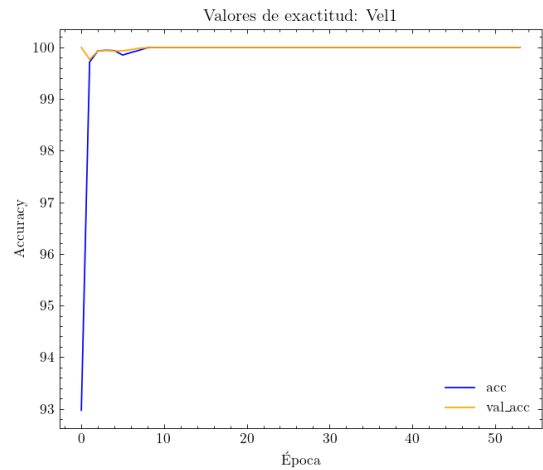


Figura 13. Valores de exactitud (Vel1) del primer modelo en su primer experimento

Fuente: Elaboración propia

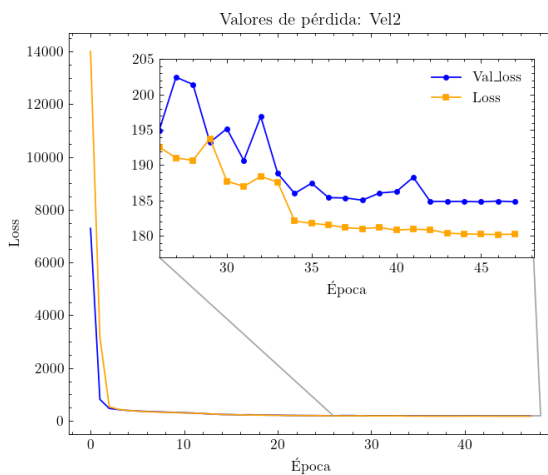


Figura 14. Valores de pérdida (Vel2) del primer modelo en su primer experimento

Fuente: Elaboración propia

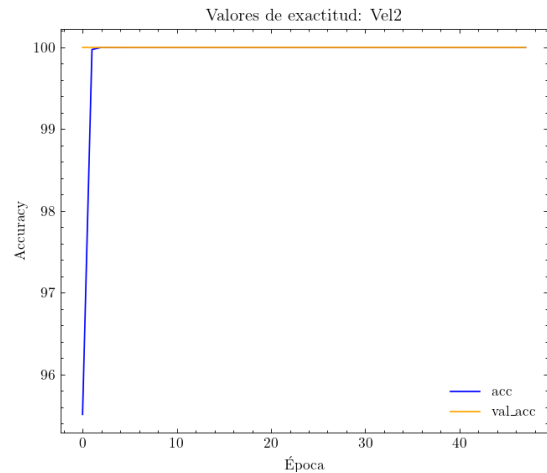


Figura 15. Valores de exactitud (Vel2) del primer modelo en su primer experimento

Fuente: Elaboración propia

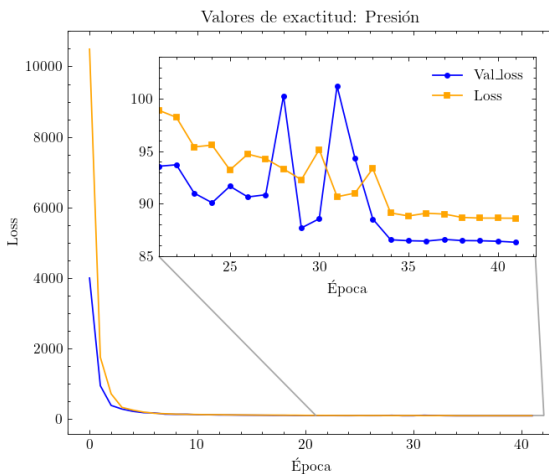


Figura 16. Valores de pérdida (presión) del primer modelo en su primer experimento

Fuente: Elaboración propia

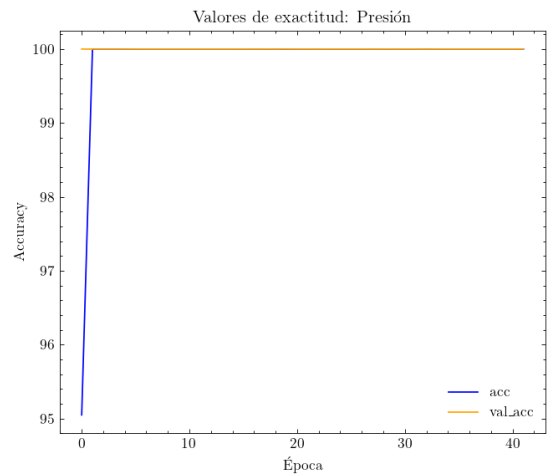


Figura 17. Valores de exactitud (presión) del primer modelo en su primer experimento

Fuente: Elaboración propia

A continuación, se realizó un análisis de los resultados obtenidos (*Figuras 12, 13, 14, 15, 16 y 17*). Primeramente, se observa como el número de épocas finales nunca alcanzó las 100 establecidas debido al *EarlyStopping*, evitando de este modo el sobreaprendizaje. Además, el error alcanzado en el conjunto de datos *test* fue en los campos de velocidad en el eje X (Vel1) de 229,70, en el eje Y (Vel2) de 175,05 y en los campos de presión de 86,05, mientras que en el caso de la exactitud siempre se alcanzó el 100%. Por ello, al ser el error bajo (véase la exactitud), se infiere un correcto funcionamiento. Más aún, en todos los casos el error en los datos de *test* es menor que durante el entrenamiento, indicando una buena capacidad de generalización y por consiguiente aplicabilidad a gran escala del modelo. En lo referente a la exactitud, se concluyó que no era una métrica relevante a monitorizar ya que rápidamente alcanzaba el 100%, debido al alto número de parámetros, aun cuando el modelo no hubiese alcanzado una tasa de error cuadrático medio igual a 0, dificultando el análisis del desempeño de los distintos hiperparámetros. Posteriormente, se visualizaron de manera gráfica los campos de presión y velocidad resultantes (*Figura 18*).

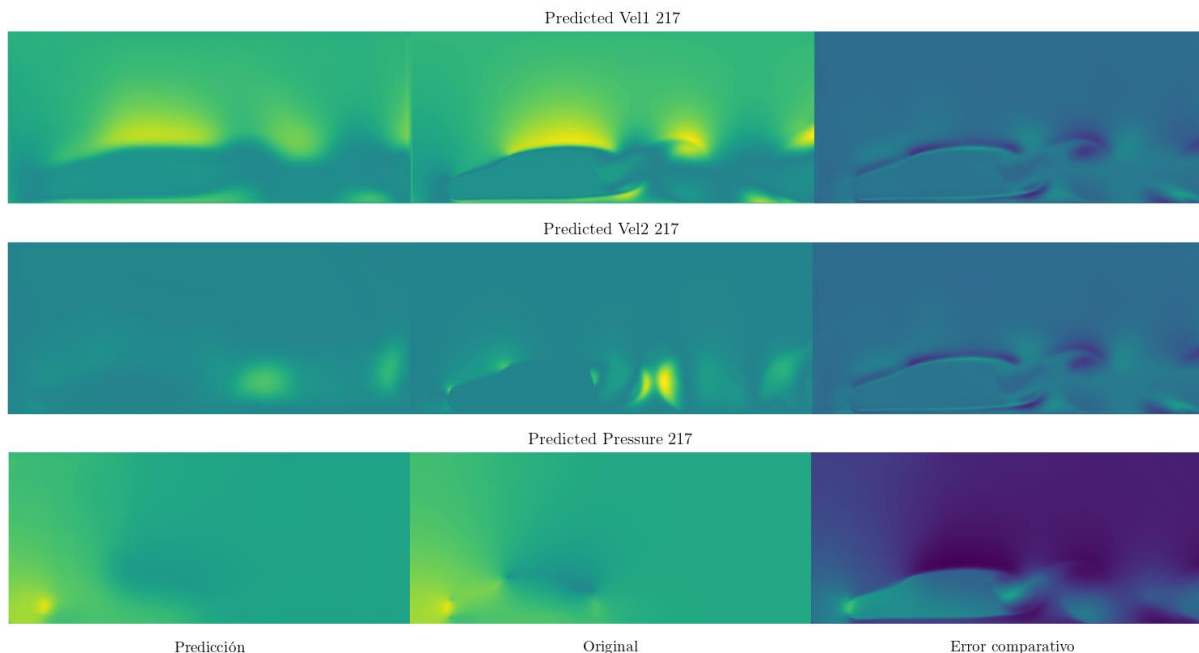


Figura 18. Resultados del primer modelo en su primer experimento

Fuente: Elaboración propia

Primer modelo: experimento 2

A la luz de los resultados anteriores, se ajustaron los hiperparámetros (*Tabla 2*) y se añadió más complejidad y profundidad al modelo mediante un mayor número de parámetros y 2 capas convolucionales más, resultando en 23 capas convolucionales, 5 capas de agrupación máxima, 5 capas de concatenación y 5 capas de convolución

transpuesta. Igualmente, se ajustaron tanto el tamaño del lote de entrenamiento como de validación al que había resultado en un menor error en el experimento anterior, 24.

Tabla 2. Parámetros del segundo experimento del primer modelo

	Vel1	Vel2	Presión
Épocas (finales)	22	38	15
Tamaño del lote: entrenamiento	24	24	24
Tamaño del lote: validación	24	24	24
Tasa de aprendizaje inicial	10^{-3}	10^{-3}	10^{-3}
Tasa de aprendizaje final	$2,5 \cdot 10^{-4}$	$3,125 \cdot 10^{-5}$	10^{-4}
Nº de parámetros a entrenar	6.760.516	6.760.516	6.760.516

Una vez más, se visualizó la evolución del error en los campos de velocidad y presión, durante el entrenamiento y validación. No obstante, ya no se monitorizó la exactitud.

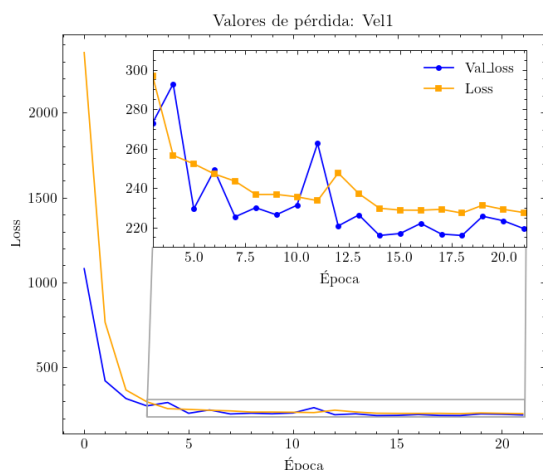


Figura 19. Valores de pérdida (Vel1) del primer modelo en su segundo experimento

Fuente: Elaboración propia

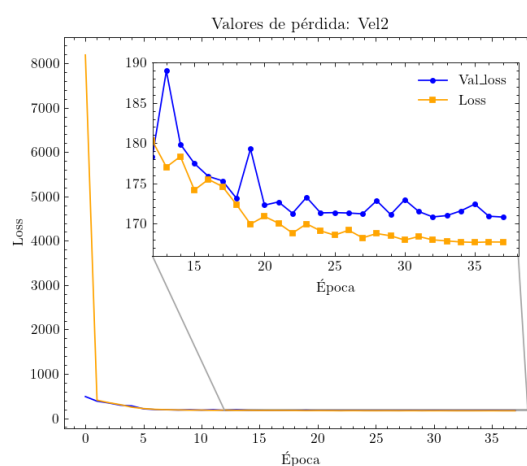


Figura 20. Valores de pérdida (Vel2) del primer modelo en su segundo experimento

Fuente: Elaboración propia

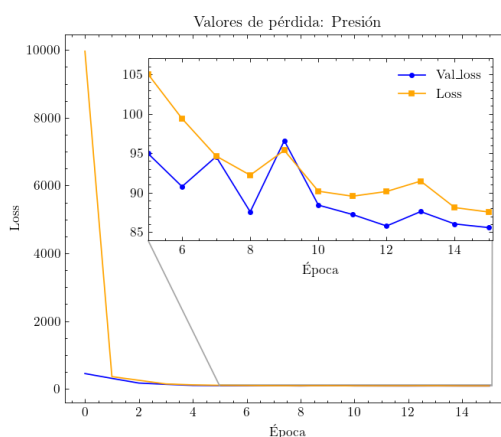


Figura 21. Valores de pérdida (presión) del primer modelo en su segundo experimento

Fuente: Elaboración propia

Una vez visualizados, se examinaron los resultados. En primer lugar y al igual que en el primer experimento, no se completaron las 100 iteraciones. Similarmente, también se observa un notable mejor desempeño del modelo en las primeras épocas (*Figuras 19, 20 y 21*), en ocasiones de más de 10.000 puntos de error cuadrático medio (Vel1), achacable a la mayor profundidad del modelo. En cuanto al MSE en los datos *test*, en los campos de velocidad en el eje X (Vel1) alcanzó 214,28, en el eje Y (Vel2) 161,90 y en los campos de presión 84,55. En resumen y en contraposición a los resultados del primer experimento, el segundo experimento fue un 6,8%, un 7,6% y un 1,8% mejor respectivamente. Igualmente, el error en los datos de validación y *test* fue de nuevo menor que los de entrenamiento. A continuación, se visualizó el desempeño del modelo gráficamente en los campos de velocidad y presión en la *Figura 22*, comprobando que es en las regiones más cercanas al vehículo, especialmente la zona trasera, donde se acumula un mayor error (azul oscuro en *Error comparativo*) por la mayor imprevisibilidad del fluido y mayor número de vórtices en esta área.

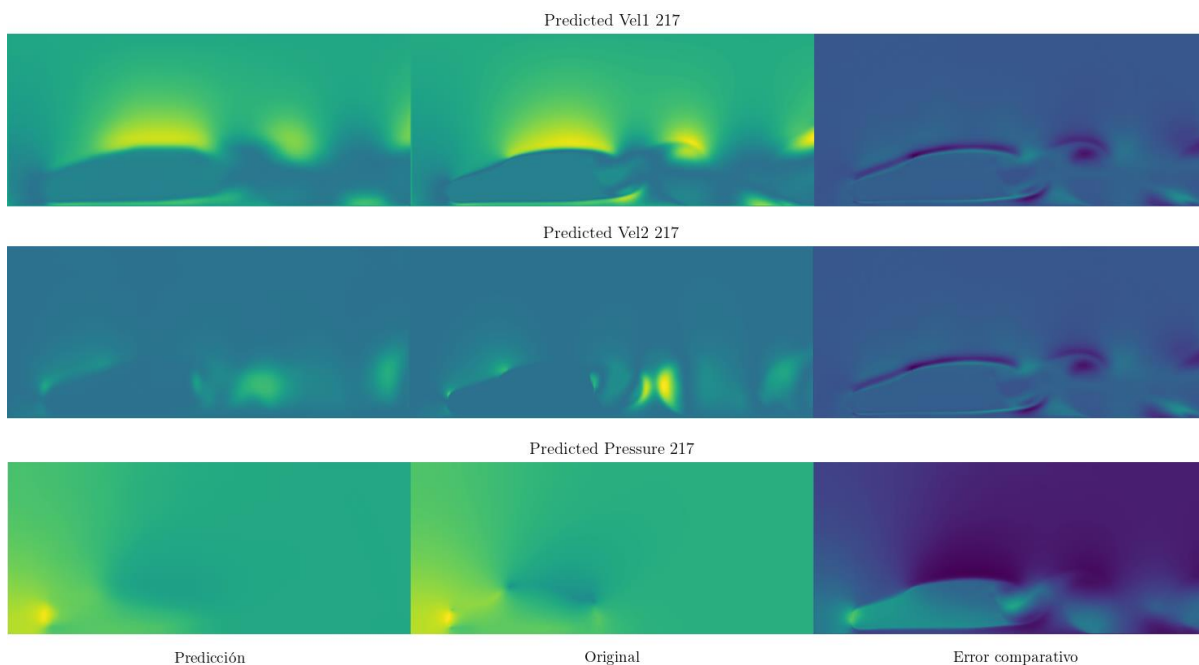


Figura 22. Resultados del primer modelo en su segundo experimento

Fuente: Elaboración propia

Segundo modelo: experimento 1

Para el entrenamiento del segundo modelo, se aumentó considerablemente el tamaño del lote (*Tabla 3*) debido al mayor número de datos y menores dimensiones de las imágenes. La arquitectura consistió en 8 capas convolucionales, 3 de agrupación máxima, 1 capa *flatten*, 1 capa completamente conectada y 1 capa *dropout*.

Tabla 3. Parámetros del primer experimento del segundo modelo

	Predicción del coeficiente de <i>drag</i>
Épocas (finales)	9
Tamaño del lote: entrenamiento	52
Tamaño del lote: validación	52
Tasa de aprendizaje inicial	10^{-4}
Tasa de aprendizaje final	10^{-6}
Nº de parámetros a entrenar	9.515.713

Una vez entrenado, se analizaron los resultados y desempeño del modelo. Al igual que con el primero, primeramente, se visualizó la evolución del error del modelo durante el entrenamiento y la fase de validación, frente a las iteraciones.

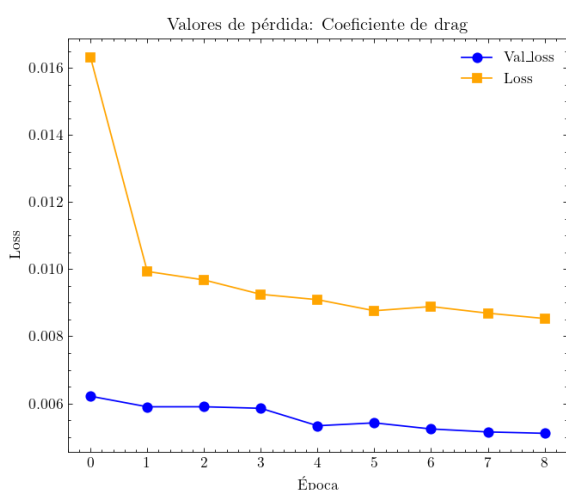


Figura 23. Valores de pérdida del segundo modelo en su primer experimento

Fuente: Elaboración propia

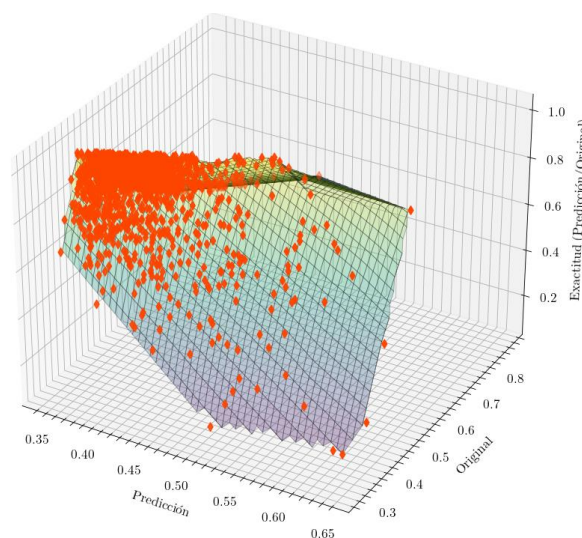


Figura 24. Resultados en el conjunto de datos *test* del segundo modelo en su primer experimento

Fuente: Elaboración propia

Posteriormente, se analizaron los resultados obtenidos. En primer lugar, y de manera similar a los experimentos del primer modelo, se observa como el entrenamiento se detiene antes de alcanzar las 100 épocas preestablecidas al no producirse mejoras significativas en el rendimiento. Igualmente, el error cuadrático medio (Figura 23) alcanzado en el conjunto de datos *test* fue de 0'005, en lo que supone una exactitud del 83,29%. Más aún, el error en los datos de validación y *test* fue siempre inferior a los de entrenamiento, denotando una buena respuesta del modelo frente a datos con los que no había sido previamente entrenado. Referente a la distribución de los resultados, en la Figura 24 se representan las predicciones realizadas por el modelo en el conjunto de datos *test* frente a su valor original y frente a su exactitud

correspondiente $\left(\frac{\text{Valor predicho}}{\text{Valor original}}\right)$. En ellas, se observa como la exactitud es cercana a 1 para aquellos valores (0,35) con un mayor número de datos y como el desempeño del modelo disminuye para los valores con un menor número de muestras similares.

Segundo modelo: experimento 2

Se entrenó el modelo de acuerdo con los hiperparámetros establecidos en la *Tabla 4*. No obstante, y a diferencia de lo realizado en el segundo experimento del primer modelo, no se modificó la estructura de la red neuronal convolucional a la luz de los buenos datos cosechados en el primer experimento, sino que únicamente se reconfiguraron los hiperparámetros, haciendo especial hincapié en el tamaño del lote.

Tabla 4. Parámetros del segundo experimento del segundo modelo

	Predicción del coeficiente de <i>drag</i>
Épocas (finales)	20
Tamaño del lote: entrenamiento	62
Tamaño del lote: validación	62
Tasa de aprendizaje inicial	10^{-4}
Tasa de aprendizaje final	10^{-8}
Nº de parámetros a entrenar	9.515.713

A continuación, se recopilaron y visualizaron los resultados tras los cambios realizados.

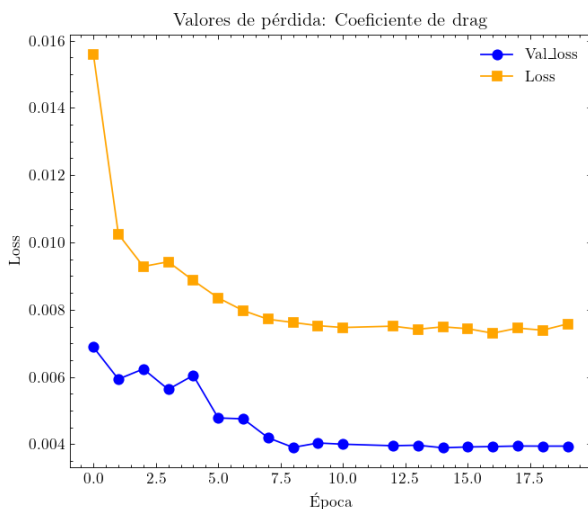


Figura 25. Valores de pérdida del segundo modelo en su segundo experimento

Fuente: Elaboración propia

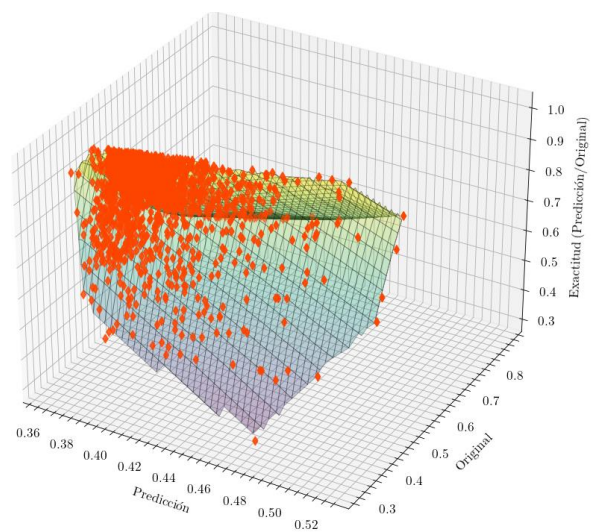


Figura 26. Resultados en el conjunto de datos *test* del segundo modelo en su segundo experimento

Fuente: Elaboración propia

Significativamente, los cambios se tradujeron en un modelo que continuó aprendiendo durante más iteraciones, reduciendo la tasa de error cuadrático medio (*Figura 25*) a 0'0039, en lo que supone una mejora del 22%, por otro lado, la exactitud mejoró a un 85,58%. Referente a la distribución de resultados (*Figura 26*) y a diferencia del primer experimento, se observa un importante aumento de la exactitud para aquellos valores menos habituales, indicando una mayor capacidad generalización del modelo ante perfiles de automóviles desconocidos.

6. Conclusiones

- Los campos de presiones y fluidos, así como el coeficiente de resistencia aerodinámica al avance, aun cuando sean no lineales, pueden ser predichos por las redes neuronales convolucionales con un alto grado de precisión.
- El alto grado de exactitud y bajo error alcanzado por ambos modelos de predicción indican que podrían ser implementados en la industria a gran escala. Más aún, la reducción en los costes económicos, de recursos y de tiempo derivados de su implementación contribuirían significativamente a la optimización y futura sostenibilidad climática y financiera de la industria.
- La capacidad de generalización de las redes neuronales convolucionales unida a mayores bases de datos permite que los modelos puedan extrapolar sus aprendizajes a diversas formas de vehículos, condiciones y entornos sin necesidad de un reentrenamiento significativo, acelerando el proceso de diseño y permitiendo a los ingenieros optimizar configuraciones más eficientemente.
- A diferencia de los métodos basados en la dinámica de fluidos computacional, las redes neuronales convolucionales operan sin el mallado de la geometría del vehículo, lo que simplifica el proceso y evita las limitaciones derivadas de la calidad y resolución de la malla.
- Durante el entrenamiento de un sistema de predicción basado en inteligencia artificial, la correcta definición de los hiperparámetros, así como una mayor cantidad de datos que entrenar se traducen en una menor tasa de error cuadrático medio y en una mayor exactitud, y con ello en mejores resultados.

7. Futuras líneas de investigación

En cuanto a futuras líneas de investigación, además de la mejora de los sistemas de predicción planteados, se propone:

- La creación de un sistema de predicción basado en inteligencia artificial que realice diseños aerodinámicos inteligentes reduciendo la resistencia aerodinámica al avance. Para ello, debería no solo predecir sino interpretar los perfiles aerodinámicos, modificándolos hasta que alcancen el diseño ideal.
- La aplicación de los sistemas de predicción planteados en industrias relacionadas, como por ejemplo la aeronáutica, adaptándolos a sus necesidades específicas derivadas.

8. Bibliografía

- Aston Zhang, Z. C. (2023). *Dive into Deep Learning*. Obtenido de Dive into deep learning: <https://bitly.ws/33Epf>
- Berzal, F. (2019). *Redes Neuronales & Deep Learning*. Independently published.
- Binyang Song, C. Y. (2023). *Surrogate Modeling of Car Drag Coefficient with Depth and Normal Renderings*. arXiv. Obtenido de <https://bitly.ws/33Esn>
- Brunton, S. L., Noack, B. R., & Koumoutsakos, P. (2020). *Machine Learning for Fluid Mechanics*. Annual Reviews. Obtenido de <https://bitly.ws/33Esh>
- Keras. (s.f.). *About Keras*. Obtenido de Keras: <https://keras.io/about/>
- Massobrio, A. (s.f.). *Applying Machine Learning in CFD to Accelerate Simulation*. Obtenido de Neural Concept: <https://bitly.ws/33EuB>
- Matplotlib. (s.f.). *Matplotlib: Visualization with Python*. Obtenido de Matplotlib: <https://matplotlib.org/>
- Misra, D. (2020). *Mish: A Self Regularized Non-Monotonic Activation Function*. arXiv. Obtenido de <https://bitly.ws/33EuX>
- Naciones Unidas. (s.f.). *Objetivo 9: Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación*. Obtenido de United Nations: <https://bitly.ws/33Eou>
- Nicolas Rosset, G. C. (2023). *Interactive design of 2D car profiles with aerodynamic feedback*. Computer Graphics Forum. Obtenido de <https://bitly.ws/33EvK>
- NumPy. (s.f.). Obtenido de NumPy: <https://numpy.org/>
- Osma, M. F. (10 de septiembre de 2019). *Efecto Coanda y drag de forma*. Obtenido de Aerodinámica F1: <https://bitly.ws/33Ew4>
- Park, J. J., Florence, P., Straub, J., Newcombe, R., & Lovegrove, S. (2019). *DeepSDF: Learning Continuous Signed Distance Functions for Shape Representation*. arXiv. Obtenido de <https://bitly.ws/33EwE>
- Python. (s.f.). *os — Miscellaneous operating system interfaces*. Obtenido de Python: <https://bitly.ws/33EoF>
- Shaw, S. (12 de octubre de 2022). *Activation Functions Compared With Experiments*. Obtenido de W&B Fully Connected: <https://bitly.ws/33Ey6>
- Xiaofeng Cao, Q. W. (2023). *Research on Fast CFD Simulation of Automobile Flow Field Based on Artificial Intelligence*. Journal of Physics: Conference Series. Obtenido de <https://bitly.ws/33Eyi>