Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE SEGOVIA

Grado en Ingeniería Informática de Servicios y Aplicaciones

Soluciones *Deep Learning* para *deblurring* y su aplicación en entornos industriales

---

Autor: David Villacorta Nicolás

Tutores: Aníbal Bregón Bregón
Paula Mielgo Martín

Fecha: 26 de junio de 2025

Soluciones *Deep Learning* para *deblurring* y su aplicación en entornos industriales

David Villacorta Nicolás

26 de junio de 2025

A Carmen y Miguel.
Sin vosotros no estaría aquí.

Resumen

El desenfoque o *blur* es un problema frecuente en las imágenes, que afecta a la calidad visual y al rendimiento de los sistemas que dependen de ellas. En este trabajo se exploran las principales técnicas de *deblurring* basadas en *Deep Learning*, analizando diferentes arquitecturas que han demostrado ser eficaces en la restauración de la calidad de las imágenes.
Además, se estudia su aplicación en entornos industriales, donde la calidad de las imágenes es un factor clave para determinados procesos. A través de este estudio, se busca evaluar el impacto de estas soluciones en la mejora de la precisión y eficiencia operativa en distintos ámbitos industriales.

Palabras claves: inteligencia artificial, *deep learning*, *blur*, *deblurring*, redes neuronales, entornos industriales.

Abstract

Blur is a common problem in images, that affects visual quality and the performance of systems that rely on them. This work explores the main *deblurring* techniques based on *Deep Learning*, analyzing different architectures that have proven to be effective in restoring image quality.
Additionally, its application in industrial environments is studied, where image quality is a key factor in certain processes. Through this study, we aim to evaluate the impact of these solutions on improving accuracy and operational efficiency in various industrial fields.

Keywords: artificial intelligence, *deep learning*, *blur*, *deblurring*, neural networks, industrial environments.

Índice general

Índice de figuras

Índice de tablas

Parte I

Descripción del proyecto

Capítulo 1

Introducción

El desenfoque o *blur* ha sido tradicionalmente un problema persistente que afecta a la calidad visual y a la utilidad de las imágenes. Este fenómeno está causado por factores como el movimiento de la cámara, la vibración, o la profundidad de campo limitada. A lo largo del tiempo, se han desarrollado diversas técnicas para reducir el desenfoque, desde métodos ópticos [37] y algoritmos basados en modelos físicos [18], hasta sofisticadas técnicas de restauración de imágenes [28]. Con el auge de la inteligencia artificial y, en particular, del *Deep Learning*, el campo de la restauración de imágenes ha experimentado un gran avance. Los modelos basados en redes neuronales profundas han demostrado ser capaces de recuperar detalles perdidos en imágenes borrosas con una precisión que supera a los métodos tradicionales [11]. El proceso de eliminación del desenfoque, conocido como *deblurring*, se ha convertido en un área de investigación clave, con aplicaciones en múltiples ámbitos, desde la mejora de imágenes en astronomía hasta la recuperación de documentos históricos o la estabilización de vídeos en dispositivos móviles.

El *Deep Learning* ha revolucionado la restauración de imágenes al permitir que las redes neuronales aprendan representaciones complejas directamente a partir de los datos, sin la necesidad de definir explícitamente modelos matemáticos del desenfoque. Como se verá más adelante, arquitecturas como las redes neuronales convolucionales (CNN), redes recurrentes (RNN, LSTM) y modelos generativos adversarios (GAN) han demostrado ser particularmente efectivas para el *deblurring*. Estas redes pueden ser entrenadas con grandes volúmenes de imágenes degradadas y sus versiones nítidas, permitiendo que el modelo aprenda a reconstruir detalles perdidos y mejorar la nitidez con un nivel de precisión sin precedentes. Además, el uso de estrategias como atenciones espaciales y conexiones residuales han supuesto mejoras significativas en la eliminación del desenfoque en imágenes con distintos tipos de degradación [4].

En este proyecto analizamos y comparamos diferentes enfoques de *deblurring* basados en *Deep Learning*, evaluando su rendimiento en distintos tipos de imágenes y niveles de desenfoque. Nuestro objetivo es aprovechar las ventajas de estas técnicas avanzadas para desarrollar soluciones más eficientes y precisas en la restauración de imágenes. A través de este estudio, buscamos identificar qué modelos y metodologías ofrecen los mejores resultados, tanto en términos de calidad visual como en eficiencia computacional para distintos escenarios relacionados con entornos industriales, en los que la eliminación del *blur* es un factor crítico.

1.1. Planteamiento del problema (*Problem Statement*)

El planteamiento del problema [35] presenta una visión general del problema planteado en el proyecto, resaltando las diferencias existentes entre la situación ideal y la realidad actual. Asimismo, se analizan las posibles repercusiones negativas de no resolver el problema. Finalmente, se plantea una propuesta de solución que servirá como fundamento para el desarrollo del proyecto.

- IDEAL: En un entorno ideal, todas las imágenes que capturamos con cámaras digitales y dispositivos ópticos serían siempre nítidas y libres de distorsiones. Factores como el movimiento de la cámara, la vibración o el desenfoque atmosférico no afectarían a la calidad de las imágenes. Esto permitiría un procesamiento con alta precisión en aplicaciones como clasificación de imágenes, reconocimiento de objetos y restauración de imágenes antiguas.
- REALIDAD: Sin embargo, en la práctica, muchas imágenes sufren de *blur* debido a diversos factores, como el movimiento de la cámara o del objeto, condiciones atmosféricas o limitaciones en la tecnología. Este desenfoque no siempre es sencillo de eliminar, y degrada la calidad de la imagen, dificultando su uso en aplicaciones críticas como la seguridad, la medicina y la visión computacional.
- CONSECUENCIAS: No resolver este problema implica que muchas imágenes capturadas en condiciones adversas sean inutilizables o requieran un procesamiento extenso para su mejora. En aplicaciones dentro de campos como la seguridad, la medicina y la visión computacional puede llevar a errores críticos. Además, la pérdida de detalles en imágenes históricas o en arte digital puede afectar su valor informativo y estético. En entornos industriales puede llevar a errores en la toma de decisiones e incluso a riesgos laborales, aumentando los costes y reduciendo la productividad.
- PROPUESTA: Para abordar este problema, en este proyecto exploramos diferentes técnicas de *deblurring* con *Deep Learning*. Para ello, evaluamos y comparamos estas técnicas para determinar cuál ofrece el mejor rendimiento en la eliminación de *blur* en distintos tipos de imágenes.

1.2. Objetivos

El proyecto tiene como objetivo general:

> **OBJ-G. Construir y evaluar la aplicación de técnicas de *deblurring* basadas en *Deep Learning* para la mejora de la calidad de imagen en entornos industriales.**

A partir de este objetivo general, se plantean los siguientes objetivos específicos:

- **OBJ-01.** Estudiar el fenómeno del desenfoque en imágenes, analizando sus causas, efectos sobre la calidad visual y clasificando los distintos tipos de *blur* más comunes.
- **OBJ-02.** Explorar las principales soluciones existentes basadas en *Deep Learning* para la restauración de imágenes, profundizando en las arquitecturas utilizadas y evaluando sus ventajas, limitaciones y adecuación al problema del desenfoque.
- **OBJ-03.** Implementar técnicas de *deblurring* basadas en *Deep Learning* aplicadas a imágenes industriales, con el objetivo de mejorar la nitidez de forma eficaz y adaptada a las condiciones propias de entornos reales.
- **OBJ-04.** Evaluar y comparar el rendimiento de los modelos, determinando su efectividad en la reducción del desenfoque y extrayendo conclusiones sobre su aplicabilidad en los entornos industriales.

1.3. Condicionantes

Los principales aspectos que condicionan el desarrollo del proyecto son las reglas de negocio, las restricciones y las suposiciones.

1.3.1. Reglas de negocio

Las reglas de negocio se refieren a los principios y normativas que rigen la metodología y el enfoque del proyecto. Estas reglas garantizan que el desarrollo se lleve a cabo de manera rigurosa, ética y alineada con los objetivos planteados. Las reglas de negocio de este proyecto vienen recogidas en la Tabla 1.1.

ID	Regla de Negocio
RN-01	El desarrollo del proyecto debe seguir una metodología clara y estructurada, asegurando que todas las fases estén correctamente documentadas.
RN-02	La solución propuesta debe estar fundamentada en principios científicos y basarse en literatura y trabajos previos sobre la materia.
RN-03	Todas las fuentes de datos y herramientas utilizadas en el desarrollo del proyecto deben estar debidamente citadas y cumplir con las normativas éticas y legales.

Tabla 1.1: Reglas de negocio

1.3.2. Restricciones

Las restricciones son limitaciones impuestas al proyecto, ya sean de tipo técnico, económico, temporal o de recursos, que pueden afectar el alcance del proyecto y su desarrollo. Las restricciones de este proyecto vienen recogidas en la Tabla 1.2. La primera restricción hace referencia al alcance del proyecto, la segunda a los costes, y la tercera al tiempo.

ID	Restricción
R-01	El proyecto debe completarse respetando la carga de trabajo establecida, que es de 12 ECTS.
R-02	La experimentación debe realizarse con herramientas y entornos accesibles para un estudiante, evitando dependencias de *hardware* o *software* propietario de alto costo.
R-03	Los modelos de Deep Learning utilizados deben ser seleccionados en función de su viabilidad, priorizando arquitecturas que puedan ser entrenadas y evaluadas dentro del tiempo disponible.

Tabla 1.2: Restricciones

1.3.3. Suposiciones

Las suposiciones son condiciones que se consideran verdaderas para el desarrollo del proyecto, aunque no puedan garantizarse completamente. Se establecen para definir un marco de trabajo bajo el cual el proyecto puede planificarse y ejecutarse. Las suposiciones consideradas para este proyecto vienen recogidas en la Tabla 1.3.

ID	Suposición
S-01	Se asume que los resultados obtenidos en las pruebas experimentales reflejan el comportamiento general de las técnicas estudiadas y pueden ser extrapolados a otros conjuntos de imágenes similares.
S-02	Se considera que las imágenes sin *blur* utilizadas como referencia en la evaluación de los modelos reflejan fielmente la calidad esperada en una imagen restaurada.

Tabla 1.3: Suposiciones del Proyecto

1.4. Alcance del proyecto

Workflow del Proyecto

En esta subsección se muestra un diagrama que muestra una visión de alto nivel sobre el flujo de trabajo del proyecto (ver Figura 1.1).

Figura 1.1: *Workflow* del Proyecto

A continuación, se describe brevemente cada fase del flujo de trabajo:

- **Introducción.** En esta fase, se explicarán los conceptos necesarios para un correcto desarrollo del proyecto. Por un lado, se introducirá el concepto de *blur*, sus causas y los distintos tipos existentes. También se definirá el concepto de *deblurring*, analizando sus tipos y distintos enfoques para la eliminación de *blur*. Y, por otro lado, se detallarán los conceptos esenciales del *Deep Learning*.

- **Exploración.** En esta fase se estudiarán las arquitecturas de Deep Learning más relevantes para el *deblurring* para su posterior implementación.

- **Implementación y Aceptación.** En esta fase se implementarán y probarán diferentes técnicas de *deblurring* utilizando modelos de *Deep Learning*. Además, se desarrollará una aplicación para el uso directo de los modelos.

- **Evaluación.** En esta fase se medirá el rendimiento de los modelos implementados en la restauración de imágenes borrosas.

- **Comparación de Modelos.** Como resultado de las fases anteriores, se obtendrá una comparación de modelos de *Deep Learning* en relación a su rendimiento eliminando el *blur* en imágenes.

Alcance

El alcance del proyecto se recoge visualmente en la Figura 1.2.

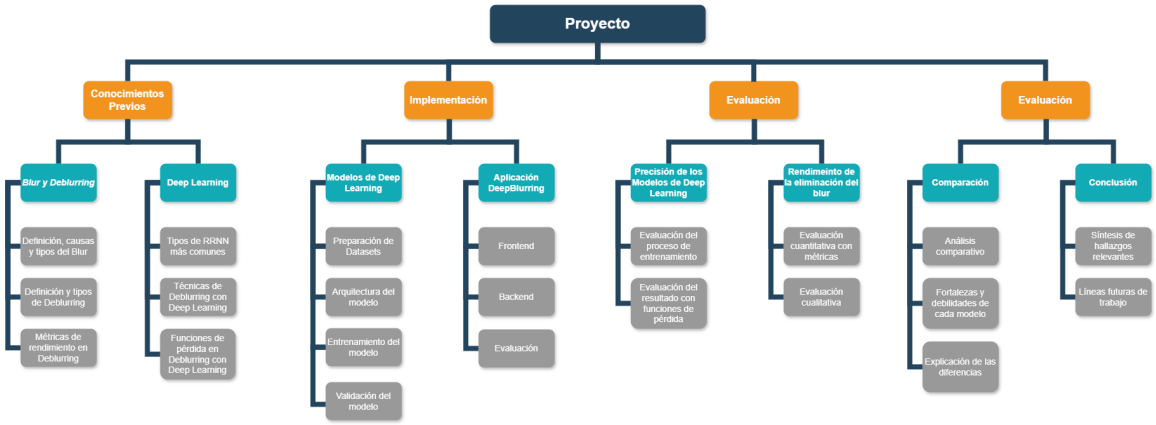


Figura 1.2: Alcance del Proyecto

Este alcance se estructura en torno a cuatro grandes áreas:

1. **Conocimientos previos.** Se abordan los fundamentos necesarios para comprender el desenfoque en imágenes y su eliminación mediante técnicas basadas en *Deep Learning*. Se incluyen aspectos como los tipos de *blur*, los enfoques tradicionales y modernos de *deblurring*, y las arquitecturas de *Deep Learning* más utilizadas.

2. **Implementación.** Se desarrolla el proceso de construcción del sistema, desde la preparación del conjunto de datos hasta el diseño, entrenamiento y validación de modelos de *Deep Learning* específicos para la tarea de *deblurring* en imágenes industriales. Además, incluye la implementación de una herramienta de visualización que permitirá utilizar los modelos de *Deep Learning* entrenados. Dicha aplicación tendrá su propio alcance delimitado, reflejado en la Figura 7.1.

3. **Evaluación.** Se analiza el rendimiento de los modelos desarrollados tanto desde la perspectiva del aprendizaje (curvas de pérdida o convergencia del modelo), como desde la calidad de los resultados obtenidos en la restauración de imágenes (mediante métricas como PSNR o SSIM).

4. **Comparación y conclusión.** Se comparan los resultados de los distintos enfoques aplicados, se identifican fortalezas y debilidades, y se sintetizan los hallazgos principales, proponiendo además líneas de mejora y trabajo futuro.

1.5. Estructura de la memoria

La estructura de la memoria es la siguiente:

- **Capítulo 1. Introducción.** Contextualiza el problema que se va a tratar en el trabajo. Además, incluye los objetivos del proyecto junto con sus condicionantes.
- **Capítulo 2. Planificación.** En este capítulo se describe la metodología empleada, Asap, junto con la planificación temporal, los presupuestos y el balance final.
- **Capítulo 3. Antecedentes.** Este capítulo describe los fundamentos teóricos y técnicos sobre los que se desarrolla el proyecto. Además, analiza diferentes propuestas ya existentes que afrontan el problema del *blur* en imágenes.
- **Capítulo 4. Diseño experimental.** Este capítulo describe la preparación necesaria para llevar a cabo la experimentación. Incluye las arquitecturas *Deep Learning* consideradas, y las métricas de éxito para evaluarlas.
- **Capítulo 5. Experimentación.** En este capítulo se detalla cómo ha sido el proceso de implementación de los modelos, así como el entrenamiento de los mismos.
- **Capítulo 6. Evaluación.** Este capítulo recoge los resultados de los modelos entrenados al enfrentarse al conjunto *test* de los diferentes *datasets* considerados.
- **Capítulo 7. Herramienta de visualización.** Este capítulo detalla todo lo relativo a la aplicación Deepblurring, una herramienta desarrollada en paralelo a la investigación que permite hacer inferencia sobre imágenes propias con los modelos considerados.
- **Capítulo 8. Conclusiones** Este capítulo final resume con perspectiva el proyecto desarrollado, además propone líneas futuras de trabajo en el campo del *deblurring* basado en *Deep Learning*.
- **Apéndice A. Glosario.** Lista una serie de términos de gran relevancia en este proyecto.
- **Apéndice B. Manual de Usuario.** Contiene las instrucciones necesarias para ejecutar la aplicación desarrollada.

Capítulo 2

Planificación

2.1. Metodología de trabajo

La metodología utilizada es ASAP (*Agile Student Academic Projects*) [29]. ASAP es una metodología reciente enmarcada dentro de las metodologías ágiles, específicamente diseñada para la organización y gestión de Trabajos Fin de Grado. El propósito fundamental de ASAP es garantizar el cumplimiento de los objetivos establecidos en el proyecto, promoviendo una dinámica de trabajo incremental basada en la comunicación regular de todos los agentes, y la generación frecuente de *feedback*. De este modo, se optimiza la calidad del producto final, asegurando que el proceso de desarrollo se ajuste a los criterios establecidos por la normativa de los Trabajos Fin de Grado.

ASAP define una serie de objetivos generales que organizan el trabajo necesario para completar un TFG de manera eficiente. Cada objetivo responde a una necesidad clave dentro del proyecto y proporciona una visión clara y estructurada del mismo. Estos objetivos se descomponen en historias de aprendizaje que detallan las tareas específicas a realizar, permitiendo una planificación más precisa. Para alcanzar un objetivo, es imprescindible completar con éxito todas sus historias de aprendizaje. Cada historia de aprendizaje incluye uno o varios criterios de aceptación, los cuales definen los resultados concretos esperados. Para que una historia se considere finalizada, es imprescindible que todos sus criterios de aceptación se cumplan y validen correctamente.

ASAP propone cinco objetivos de aprendizaje:

- **Proyecto:** Las historias relacionadas con este objetivo abarcan la definición del planteamiento del problema y de los objetivos que lo materializan, así como la identificación de las tareas necesarias para alcanzarlos, y la planificación necesaria para llevarlos a cabo. Es un proceso iterativo-incremental, lo que facilita su seguimiento y la revisión de los progresos realizados. Además, permite ajustar el ritmo de trabajo según el progreso del estudiante para no exceder la carga de trabajo prevista en la titulación.
- **Antecedentes:** Engloba las historias relacionadas con el estudio y comprensión del contexto en el que se enmarca el proyecto. Estas historias exigen familiarizarse tanto con el entorno de negocio como con el contexto científico y técnológico. Esto facilita la comprensión de la necesidad específica que motiva el desarrollo del proyecto, así como de las soluciones ya

existentes. Además, proporciona una perspectiva general sobre los conceptos fundamentales empleados en su desarrollo.

- **Desarrollo:** Este objetivo aborda el proceso de construcción del producto. Sus historias de aprendizaje se ajustan a la orientación del proyecto, ya sea de desarrollo o de investigación.
- **Aceptación:** Este objetivo de aprendizaje tiene como finalidad verificar que el producto desarrollado satisface los objetivos establecidos en el proyecto. Además, contempla una historia centrada en la discusión de los resultados, en la cual el estudiante lleva a cabo un análisis crítico del alcance y de los recursos, técnicas y procedimientos empleados, evaluando su idoneidad en relación con el cumplimiento de los objetivos del proyecto.
- **Comunicación:** Este objetivo se desglosa en dos historias de aprendizaje: la memoria técnica y el acto de defensa. La memoria técnica documenta el proceso seguido, el conocimiento adquirido y las conclusiones derivadas de los objetivos previos. Por otro lado, en el acto de defensa, el estudiante presenta y argumenta el trabajo realizado ante un tribunal evaluador. Tanto la elaboración de la memoria como la preparación de la defensa se llevan a cabo de manera iterativa a lo largo del desarrollo del proyecto, permitiendo recibir retroalimentación continua por parte del tutor o tutores, con el fin de perfeccionar el producto final y garantizar un resultado de alta calidad.

Las herramientas de Asap se basan en aquellas establecidas por Scrum [16]. De este modo, dispone de una estructura compuesta por roles, eventos y artefactos que facilitan su implementación y desarrollo.

2.1.1. Roles

La relación estudiante-tutor es clave para afrontar con éxito un TFG, pero no son las únicas personas que intervienen en su desarrollo. A continuación se muestran todos los roles existentes durante el desarrollo de este TFG:

- **Estudiante**: Es el rol principal, ya que debe realizar todas las tareas necesarias para el cumplimiento de las historias y de los objetivos de aprendizaje establecidos. Simultáneamente, debe llevar a cabo tareas de planificación como mantener actualizado el cuaderno de trabajo y el tablero del proyecto. Asimismo, debe comprometerse con la mejora continua del producto, aprovechando el feedback proporcionado por los demás roles.
- **Tutor**: Es el responsable de acompañar y guiar al estudiante a lo largo del desarrollo del proyecto, proporcionándole apoyo y *feedback*. Su papel cobra mayor importancia en las etapas iniciales del proyecto, ya que colabora con el estudiante en la definición del problema y del alcance de los objetivos, además de facilitar referencias bibliográficas para construir una sólida base teórica sobre la que desarrollar el proyecto. Asimismo, supervisa el ritmo de avance del estudiante, ajustando los objetivos en función del progreso alcanzado.
- **Comunidad**: Está formada por todas aquellas personas (estudiantes, profesores, expertos, ...) que posean la capacidad de aportar valor al proyecto durante su desarrollo. La interacción entre todos los miembros resulta fundamental para fomentar un entorno de aprendizaje donde se valoren y consideren las distintas perspectivas aportadas. De este modo, el estudiante podrá comparar diferentes puntos de vista, obteniendo así un mejor producto final.

- **Tribunal**: Representa a los miembros de la comisión evaluadora del TFG. Su rol tiene lugar en el acto de defensa del proyecto y en la posterior evaluación, de acuerdo con el nivel de satisfacción percibido tanto en el resultado entregado como en la defensa llevada a cabo por el estudiante.

2.1.2. Eventos

Los eventos constituyen un elemento fundamental para el seguimiento del TFG y garantizan una interacción constante entre el estudiante y el tutor. Además, facilitan la definición de compromisos alcanzables entre ambas partes, promoviendo así un trabajo continuo y sostenido por parte del estudiante a lo largo de todo el proyecto.

ASAP organiza el marco temporal en *sprints* de corta duración (inferior a 1 mes). Cada *sprint* establece un objetivo específico en función de las historias que se abordarán, cuyos resultados se integran al producto desarrollado previamente. La estructura por *sprints* permite minimizar la incertidumbre al organizar el trabajo en períodos cortos y construir el producto mediante incrementos sucesivos alineados con los objetivos definidos. Además, esta metodología establece un calendario regular de entregas, garantizando una retroalimentación constante al finalizar cada *sprint* y facilitando la replanificación del proyecto en función de su progreso.

- **Reunión de inicio:** Este evento marca el inicio del *sprint* y tiene como propósito definir su objetivo y planificar las tareas requeridas para alcanzarlo. El tutor selecciona las historias de aprendizaje más adecuadas para lograr los resultados esperados a corto plazo, tomando en cuenta el progreso del proyecto y la retroalimentación del *sprint* previo. Una vez definido el alcance, el estudiante determina las tareas necesarias, establece los resultados esperados de cada una y asigna fechas de finalización dentro del marco temporal del *sprint*.

- **Reunión de sincronización:** Este evento se lleva a cabo cada semana con el propósito de garantizar un seguimiento continuo del proyecto y mantener la transparencia en su progreso. Durante la reunión, el estudiante informa al tutor sobre las tareas completadas desde la última sincronización, así como sobre las actividades que tiene previsto realizar hasta la siguiente. Además, comunica cualquier obstáculo encontrado en ese período y, si es necesario, se realizan ajustes en la planificación del *sprint*.

- **Comunicación de progresos:** Este evento tiene lugar al finalizar el *sprint*, donde los estudiantes exponen el trabajo realizado hasta el momento. En este espacio, cada alumno asume un doble rol: presenta su propio TFG y participa activamente en el de sus compañeros como parte de la comunidad. Las presentaciones siguen un formato similar al de la defensa final, simulando que el proyecto está concluido. Posteriormente, se abre un debate en el que todos los asistentes pueden intervenir y aportar comentarios.

- **Retrospectiva:** Este es el evento final del *sprint*, llevado a cabo después de la comunicación de progresos. Su propósito es evaluar la calidad del proceso seguido, resaltando los aspectos positivos y proponiendo mejoras para los negativos mediante valoraciones anónimas aunque discutidas en grupo.

2.1.3. Artefactos

ASAP contempla dos artefactos para materializar los avances del proyecto:

- **Incremento:** Está formado por el conjunto de entregables que reflejan los resultados alcanzados hasta la fecha.
- **Retroalimentación:** Representa el feedback del tutor que recibe el estudiante al final de cada *sprint*, así como el de la comunidad en las Comunicaciones de progresos. Gracias a la retroalimentación, el estudiante puede comenzar el siguiente *sprint* mejorando o modificando tareas ya realizadas.

2.1.4. Entorno de trabajo

- **Espacio de trabajo compartido**: Es un entorno de comunicación que facilita la interacción privada entre los tutores y el estudiante. Además, incluye un canal general destinado al intercambio de información con la comunidad. También cuenta con un área de almacenamiento común, permitiendo compartir los avances logrados por el estudiante y la retroalimentación del tutor posteriormente.

  Este repositorio se divide en cuatro secciones:

  - **Documentación**: Contiene información relevante sobre la gestión académica del TFG.
  - **Entregas**: Espacio para almacenar el trabajo desarrollado al finalizar cada *sprint*.
  - **Referencias**: Incluye referencias bibliográficas e información útil para el desarrollo del proyecto.
  - **Workspace**: Área de trabajo individual del estudiante, que puede gestionar libremente.

  Una plataforma muy útil para este propósito, es Microsoft Teams.

- **Tablero de proyecto**: Se trata de un tablero de tipo Kanban [38], el cual consta de seis columnas: la primera muestra todas las historias del proyecto (*Backlog* del proyecto), la segunda aquellas que se van a trabajar durante el *sprint* actual (*Sprint Backlog*), la tercera incluye las tareas que aún no se han realizado (*ToDO*), la cuarta aquellas tareas que se están realizando (*Doing*), la quinta aquellas tareas que han sufrido un bloqueo (*Blocked*) y la última aquellas que han sido completadas (*Done*). Esta herramienta facilita la visualización del avance y planificación de los objetivos, puntos de historia y tareas específicas dentro de cada *sprint*. Una aplicación útil para gestionar este tipo de tablero es Trello (ver las Figuras 2.1 y 2.2).

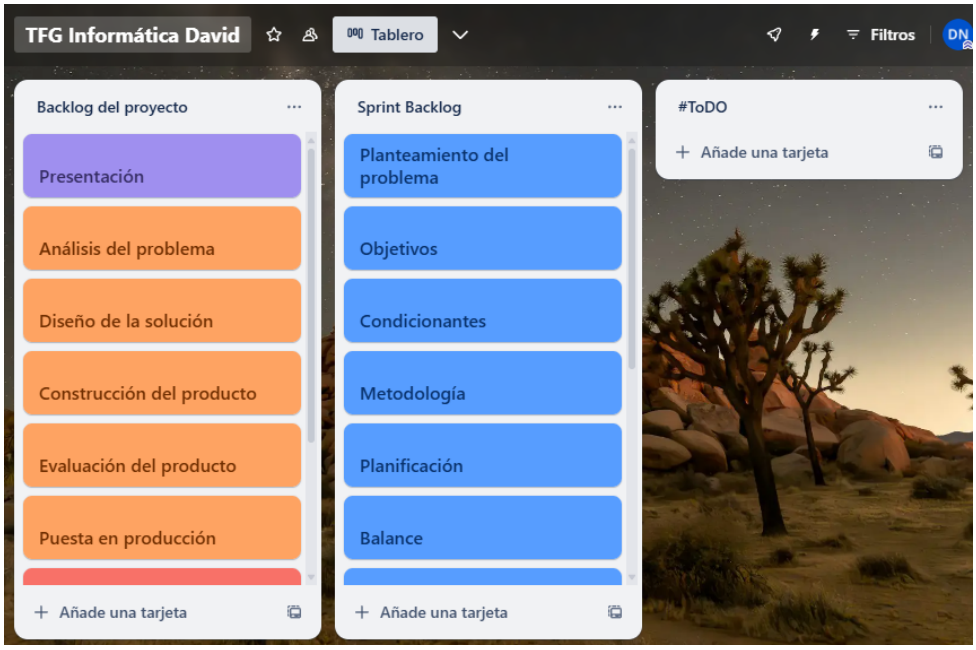


Figura 2.1: Tablero del proyecto en *Trello* (*Backlog, Sprint Backlog y ToDO*)

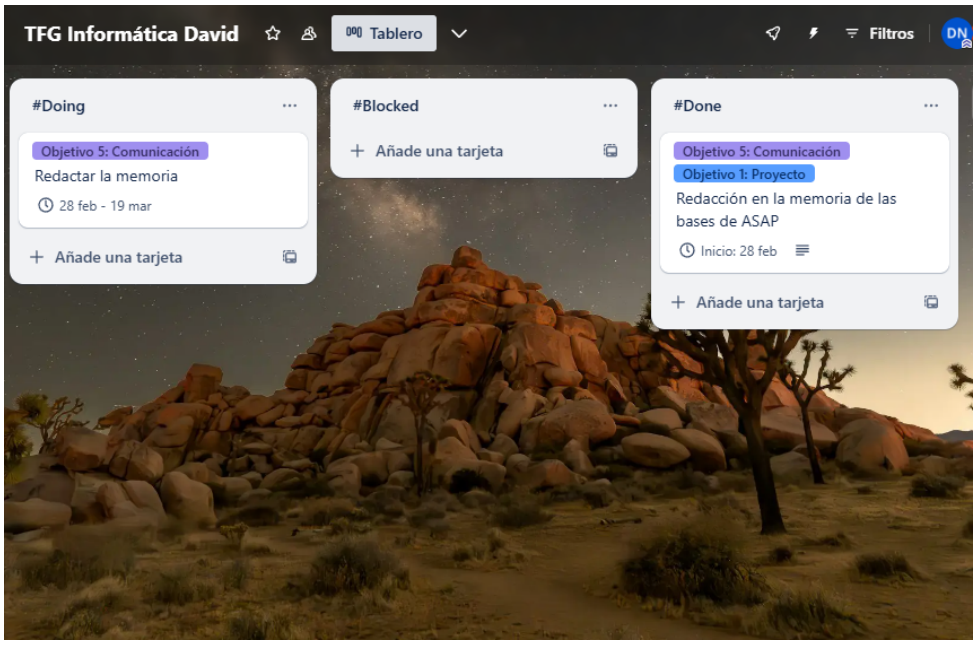


Figura 2.2: Tablero del proyecto en *Trello* (*Doing, Blocked, Done*)

- **Cuaderno de trabajo**: Es una herramienta útil para el estudiante, que le permite registrar el tiempo dedicado a cada una de las tareas planificadas, facilitando así la adopción de una rutina de trabajo. Además, permite sacar conclusiones *a posteriori* acerca de la planificación y el trabajo realizados por el estudiante. Una herramienta muy útil como cuaderno de trabajo es Excel (ver Figura 2.3).

Fecha	Historia	Tarea	Tiempo (minuto	Considaciones adicionales
27/02/2025	Planificación	Elaboración de la planificación del cronograma	60	
27/02/2025	Metodología	Lectura y comprensión de las bases de ASAP	45	
28/02/2025	Documentación técnica	Redacción de las bases de ASAP	60	
28/02/2025	Metodología	Creación del tablero del proyecto	60	Utilizando Trello
28/02/2025	Planificación	Investigación para los presupuestos	60	
01/03/2025	Planificación	Elaboración de la EDT	70	Utilizando draw.io

Figura 2.3: Cuaderno de trabajo

2.2. Planificación temporal

El proyecto se divide en cuatro *sprints*, cada uno de ellos planificado para que suponga entre 75 y 90 horas de trabajo. Esto supone entre 300 y 360 horas de trabajo en total. La planificación de los *sprints* es la siguiente:

- ***Sprint #1:*** 24 de febrero - 21 de marzo
- ***Sprint #2:*** 24 de marzo - 25 de abril
- ***Sprint #3:*** 28 de abril - 23 de mayo
- ***Sprint #4:*** 26 de mayo - 20 de junio

Sprint	Planificación	Sincro. #1	Sincro. #2	Sincro. #3	Entrega incremento	Comunicación de progresos + Retrospectiva
#1	24/02/2025	28/02/2025	07/03/2025	14/03/2025	19/03/2025	no se realiza en este sprint
#2	24/03/2025	28/03/2025	04/04/2025	11/04/2025	23/04/2025	25/04/2025
#3	28/04/2025	02/05/2025	09/05/2025	16/05/2024	21/05/2024	no se realiza en este sprint
#4	26/05/2025	30/05/2025	06/06/2025	13/06/2025	18/06/2024	20/06/2025
#5	23/06/2025	27/06/2025	04/07/2025		09/07/2024	11/07/2025

Figura 2.4: Planificación temporal

Aunque se planifica el proyecto en cuatro *sprints*, en la planificación de la Figura 2.4 aparecen cinco. El quinto desempeña el papel de *sprint* auxiliar de menor duración enfocado en la finalización de los objetivos pendientes.

Cada objetivo se compone de un conjunto de historias específicas:

- **Proyecto**
  - **Planteamiento del problema**: Esta historia de aprendizaje tiene como objetivo comprender con claridad el desafío abordado en el TFG y ser capaz de formularlo en un enunciado preciso. Dicho enunciado identificará la brecha existente entre la situación actual y el resultado esperado al finalizar el proyecto.
  - **Objetivos**: se centra en definir de manera clara y detallada los propósitos del proyecto. Estos deben servir como guía para todas las actividades y deben cumplir con los criterios SMART (específicos, medibles, alcanzables, relevantes y acotados en el tiempo) para garantizar su efectividad y evaluación adecuada.
  - **Condicionantes**: Complementando la historia anterior, esta se enfoca en identificar todos aquellos aspectos que puedan influir en el desarrollo del proyecto, ya sean de carácter técnico o de negocio. Además, se consideran suposiciones clave que podrían afectar el curso del trabajo, asegurando que se tengan en cuenta desde el inicio.
  - **Metodología**: Dado que el TFG se desarrollará bajo la metodología ASAP, esta historia busca justificar su elección como marco adecuado para estructurar y organizar el proyecto de manera eficiente y sistemática.
  - **Planificación**: Una vez definida la metodología, esta historia tiene como propósito establecer un plan de trabajo detallado, incluyendo todas las actividades necesarias para completar el proyecto. La planificación debe considerar plazos, recursos y posibles imprevistos junto con algunas contingencias.
  - **Balance**: Al final de cada *sprint*, esta historia permite analizar el uso de los recursos en relación con los resultados obtenidos, identificando desviaciones respecto a lo planificado. Esta evaluación es fundamental para optimizar la gestión del proyecto y mejorar continuamente las habilidades de planificación mediante ajustes estratégicos.
  - **Interacción**: Esta historia, que se desarrolla a lo largo de todo el *sprint*, tiene como objetivo garantizar una interacción efectiva con todos los participantes del proyecto. Implica mantener una comunicación clara, fomentar la colaboración y facilitar la retroalimentación continua para mejorar el desarrollo del trabajo.
- **Antecedentes**
  - **Entorno de negocio**: Esta historia de aprendizaje tiene como propósito adquirir un conocimiento profundo del dominio del proyecto. Para ello, es fundamental comprender los principios y dinámicas del entorno de negocio, identificando de qué manera los resultados del proyecto pueden contribuir a su evolución y qué actores o stakeholders se verán impactados por su desarrollo.
  - **Estado del arte**: Esta historia busca proporcionar una visión clara sobre cómo se han abordado problemas similares en otros proyectos. Requiere la identificación y análisis de soluciones existentes, evaluando sus fortalezas y debilidades, con el fin de detectar posibles oportunidades de mejora que puedan ser implementadas en el proyecto.
  - **Fundamentos teóricos**: Esta historia está orientada a desarrollar un conocimiento profundo sobre las teorías, principios y conceptos fundamentales que sustentan el proyecto, permitiendo comprender su contexto académico y científico.

- **Fundamentos técnicos**: Esta historia tiene como objetivo brindar una perspectiva detallada sobre las técnicas, herramientas y tecnologías comúnmente empleadas en la construcción de productos similares dentro de proyectos comparables, asegurando así una base sólida para su implementación.

- **Desarrollo**
  - **Análisis del problema**: Esta historia tiene como objetivo la comprensión del problema que se aborda en el proyecto y los diferentes tipos de requisitos que lo caracterizan.
  - **Diseño de la solución**: Esta historia se enfoca en diseñar una solución que satisfaga los objetivos del proyecto, de acuerdo con los requisitos y condicionantes identificados.
  - **Construcción del producto**: Esta historia de aprendizaje se enfoca en construir el producto que resuelve el problema planteado en el proyecto, de acuerdo con sus objetivos, condicionantes y requisitos.
  - **Evaluación del producto**: Esta historia de aprendizaje se enfoca en evaluar que el producto desarrollado cumple con los requisitos especificados y funciona correctamente en las condiciones previstas.
  - **Puesta en producción**: Esta historia de aprendizaje se enfoca en todos los aspectos relativos a la instalación y configuración del producto desarrollado.

- **Aceptación**
  - **Métricas de éxito**: Esta historia de aprendizaje persigue establecer las métricas necesarias para evaluar el éxito del proyecto, asegurando su alineamiento con los objetivos establecidos y su capacidad para evaluarlos de forma precisa y objetiva.
  - **Plan de aceptación**: Esta historia se enfoca en diseñar y ejecutar un plan de aceptación estructurado y riguroso, que asegure una evaluación precisa del proyecto.
  - **Análisis de resultados**: Esta historia se enfoca en interpretar los resultados obtenidos en el proceso de aceptación y, en base a ellos, determinar el éxito del proyecto.

- **Comunicación**
  - **Presentación**: Se centra en la elaboración de una presentación que exponga los progresos obtenidos durante el sprint, recreando el escenario de defensa del Trabajo de Fin de Grado (TFG). Debe proporcionar un contexto del proyecto, detallar los avances conseguidos y ofrecer un análisis crítico en función de la planificación y los objetivos definidos.
  - **Memoria**: Esta historia engloba las tareas orientadas a la redacción de una documentación detallada, bien estructurada y de alta calidad para el proyecto.

A continuación, se presenta la planificación y el alcance particular de cada uno de los *sprints*, utilizando una EDT (Estructura de Desglose de Trabajo) y mostrando la relación entre las actividades, su orden de ejecución y duración a través de un cronograma, específicamente un diagrama de Gantt.

2.2.1. Sprint #1

En este *sprint* se abordan las historias y tareas asociadas a los objetivos de Proyecto, Antecedentes y Comunicación. Esto se refleja en la EDT (ver Figura 2.5) y en el diagrama de Gantt (ver Figura 2.6). En términos generales, este primer *sprint* se centra en el análisis del contexto del proyecto, permitiendo la exploración de soluciones previas al problema planteado y la adquisición de conocimientos fundamentales para su desarrollo.
El identificador de cada historia comienza por "H", y su primer número corresponde al objetivo en el que se encuentra enmarcada. (Por ejemplo: La historia H7.1 es la primera historia del objetivo 7). Análogamente sucede con las tareas dentro de cada historia. (Por ejemplo: La tarea T7.1.1 es la primera tarea dentro de la historia 7.1).

2.2.2. Sprint #2

Este *sprint* se centra en abordar las historias y tareas asociadas a los objetivos de Desarrollo y Aceptación, y continúa trabajando sobre los objetivos de Proyecto, Antecedentes y Comunicación, siguiendo el trabajo realizado en el *sprint* anterior. Esto se refleja en la EDT (ver Figura 2.7) y en el diagrama de Gantt (ver Figura 2.8). En términos generales, este segundo *sprint* se centra en la implementación de las diferentes técnicas de *deblurring* basadas en *Deep Learning*, así como en su evaluación y aceptación.

2.2.3. Sprint #3

Este *sprint* se centra en abordar las historias y tareas asociadas al objetivo de Desarrollo, y continúa trabajando sobre los objetivos de Proyecto y Aceptación, siguiendo el trabajo realizado en el *sprint* anterior. Esto se refleja en la EDT (ver Figura 2.9) y en el diagrama de Gantt (ver Figura 2.10). En términos generales, este tercer *sprint* se centra en el entrenamiento completo de las diferentes técnicas de *deblurring* basadas en *Deep Learning*, así como el desarrollo de la aplicación para la mejora de imágenes.

2.2.4. Sprint #4

Este *sprint* se centra en unificar todo el trabajo desarrollado los *sprints* anteriores, por lo que trabaja todos los objetivos a excepción de Antecedentes. Esto se refleja en la EDT (ver Figura 2.11) y en el diagrama de Gantt (ver Figura 2.12). Además, pone el foco sobre la última historia de aprendizaje de este proyecto: puesta en producción, que consiste en documentar la implementación y entrenamiento de los modelos de manera que este proceso pueda ser replicable en un futuro.

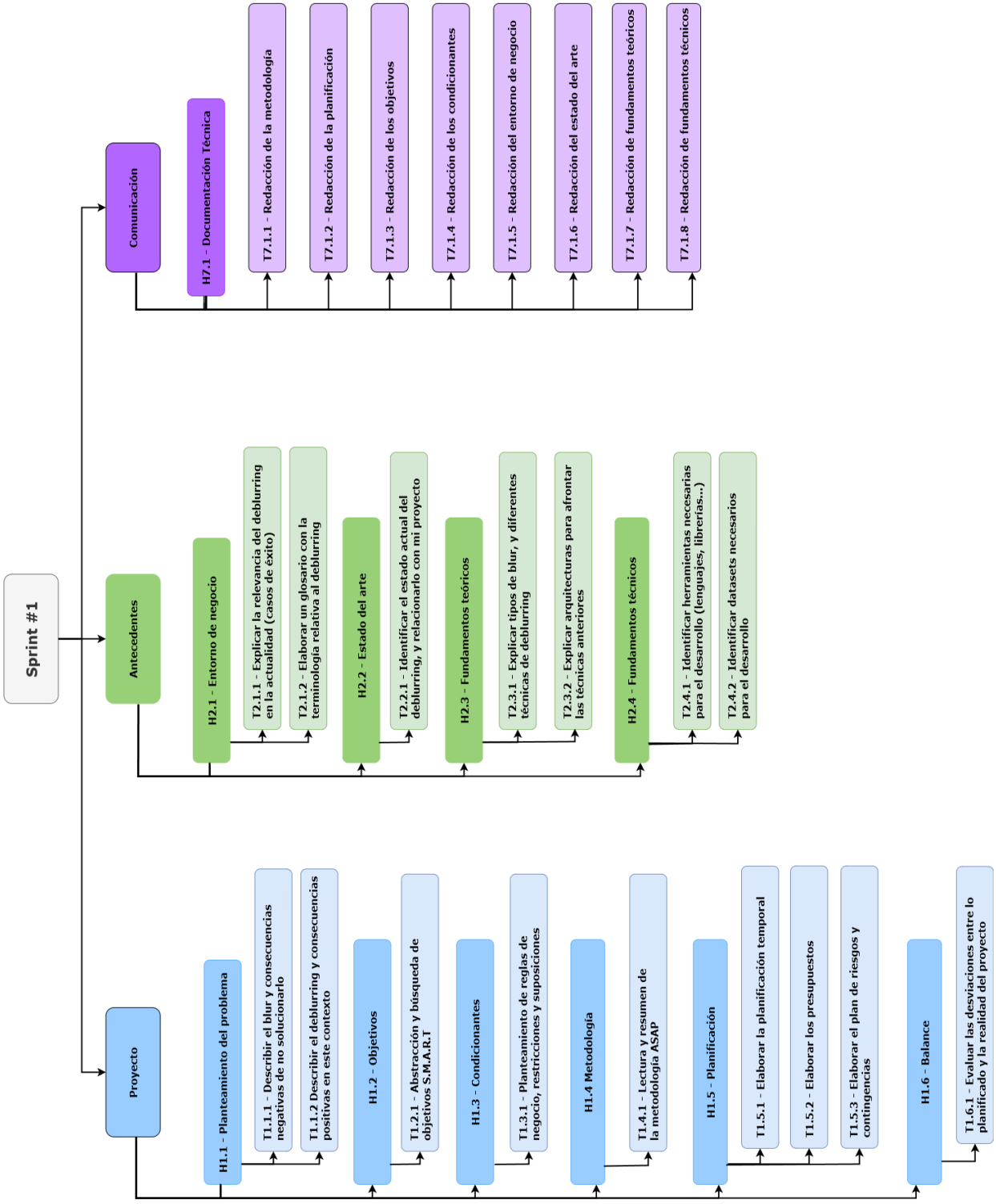


Figura 2.5: EDT del *Sprint* #1

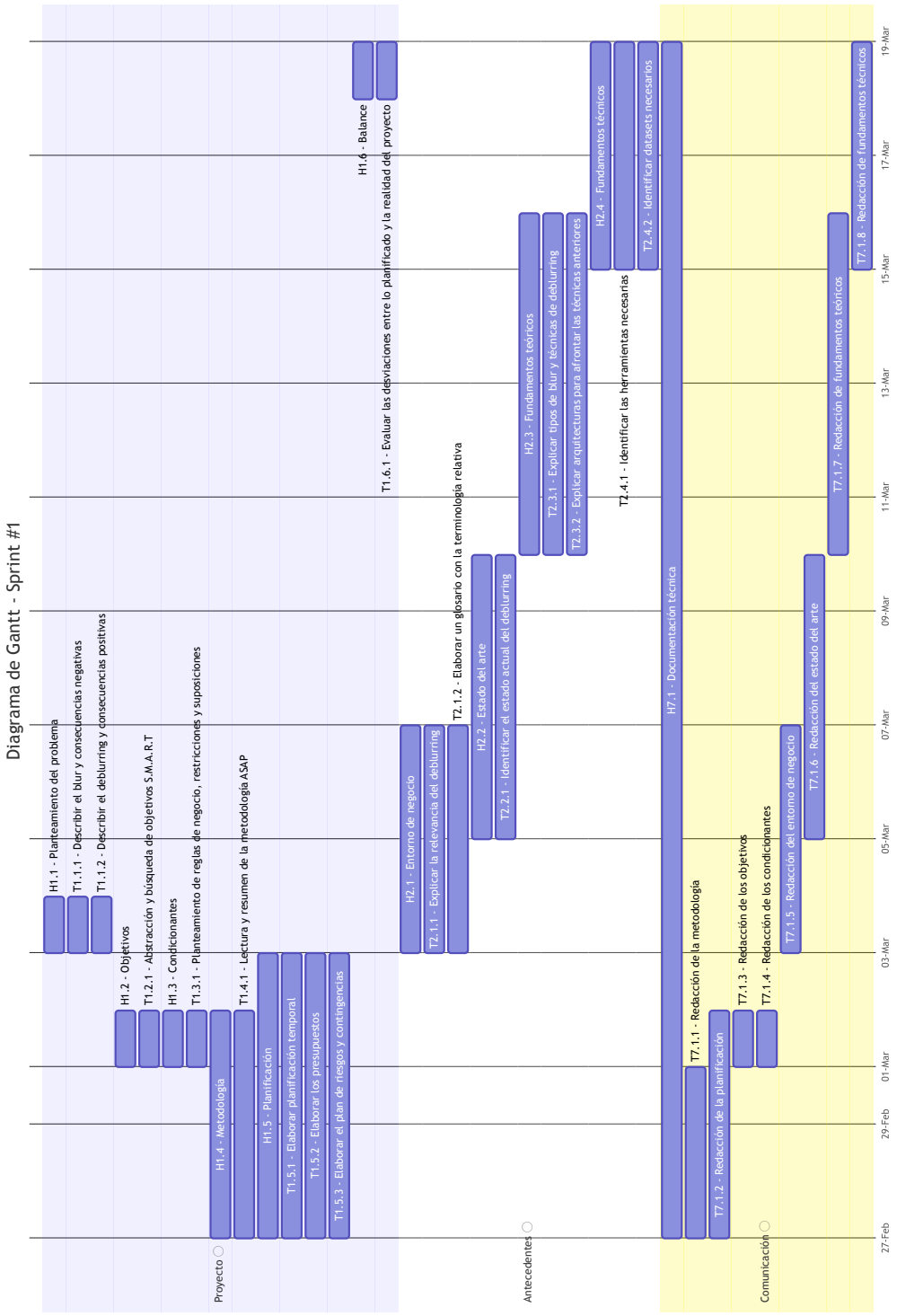


Figura 2.6: Diagrama de Gantt del *Sprint* #1

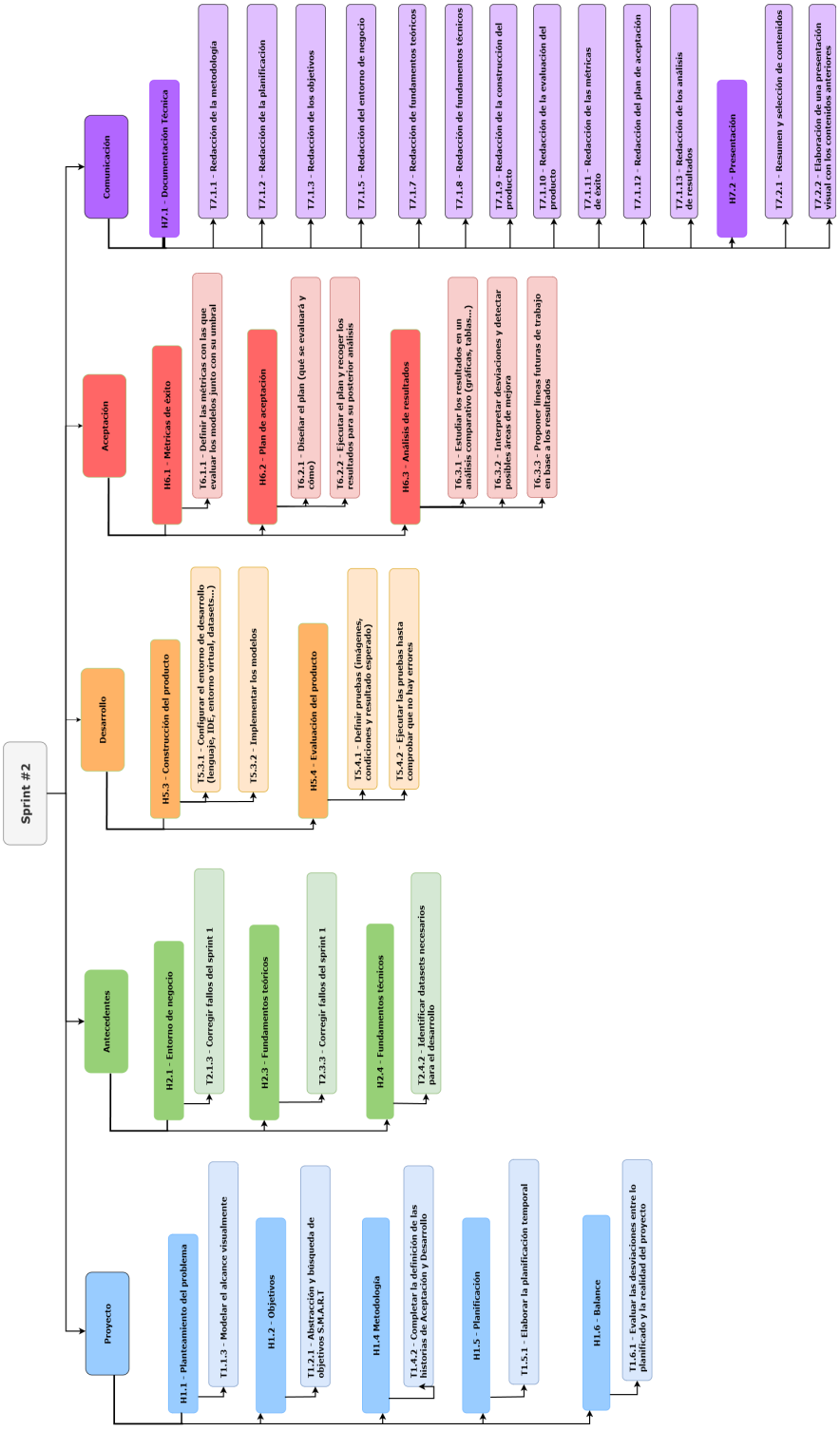


Figura 2.7: EDT del *Sprint* #2

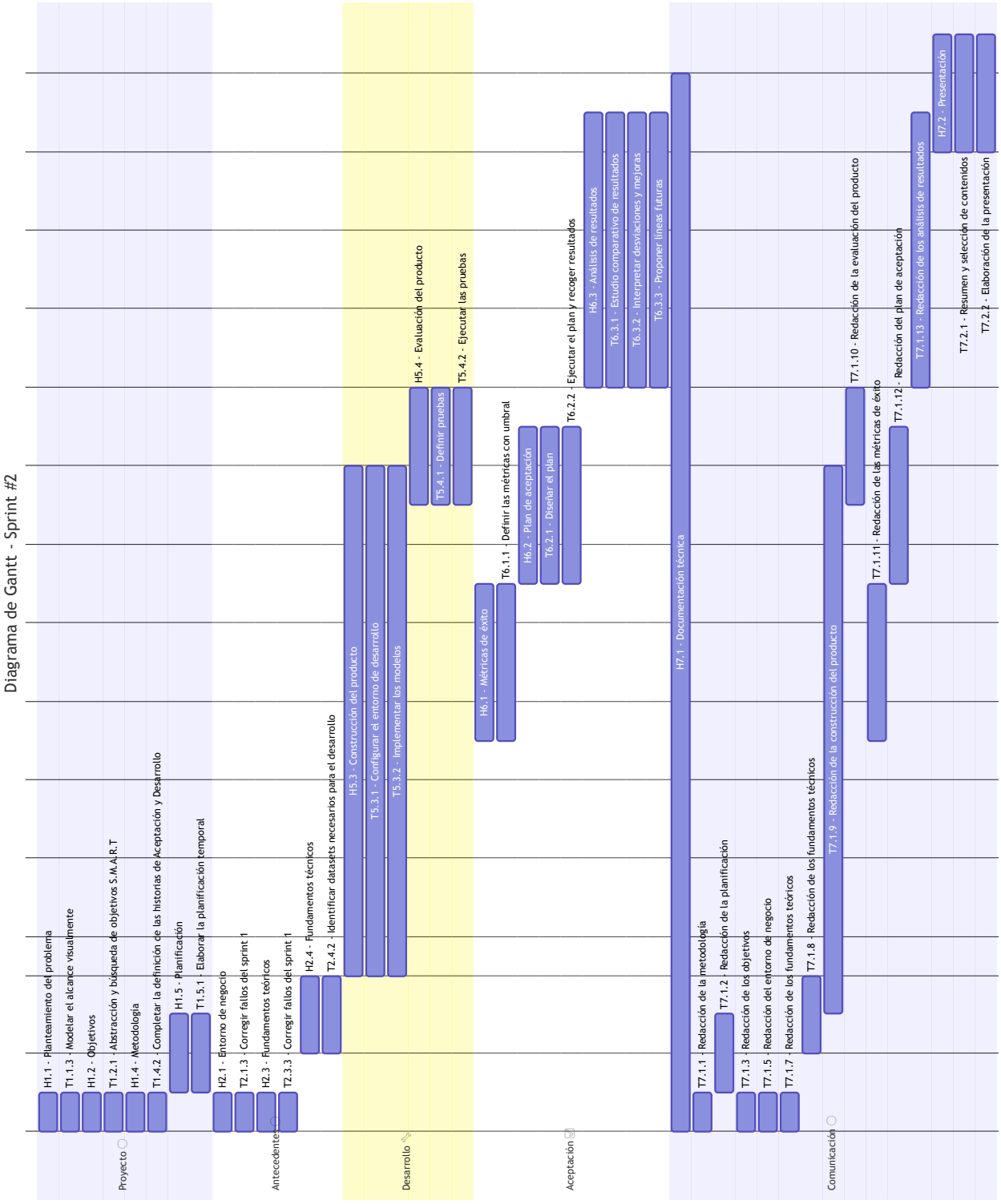


Figura 2.8: Diagrama de Gantt del *Sprint* #2

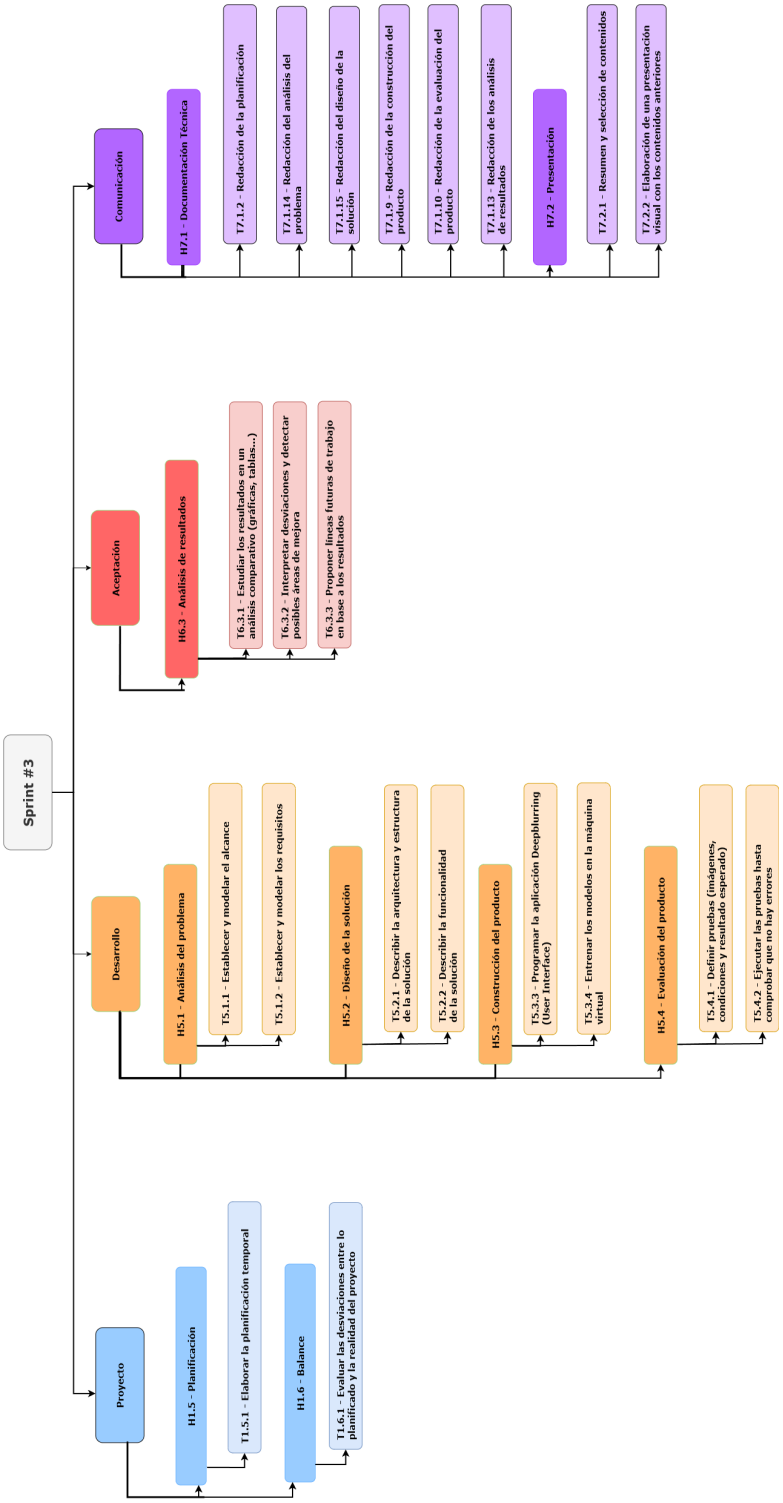


Figura 2.9: EDT del *Sprint* #3

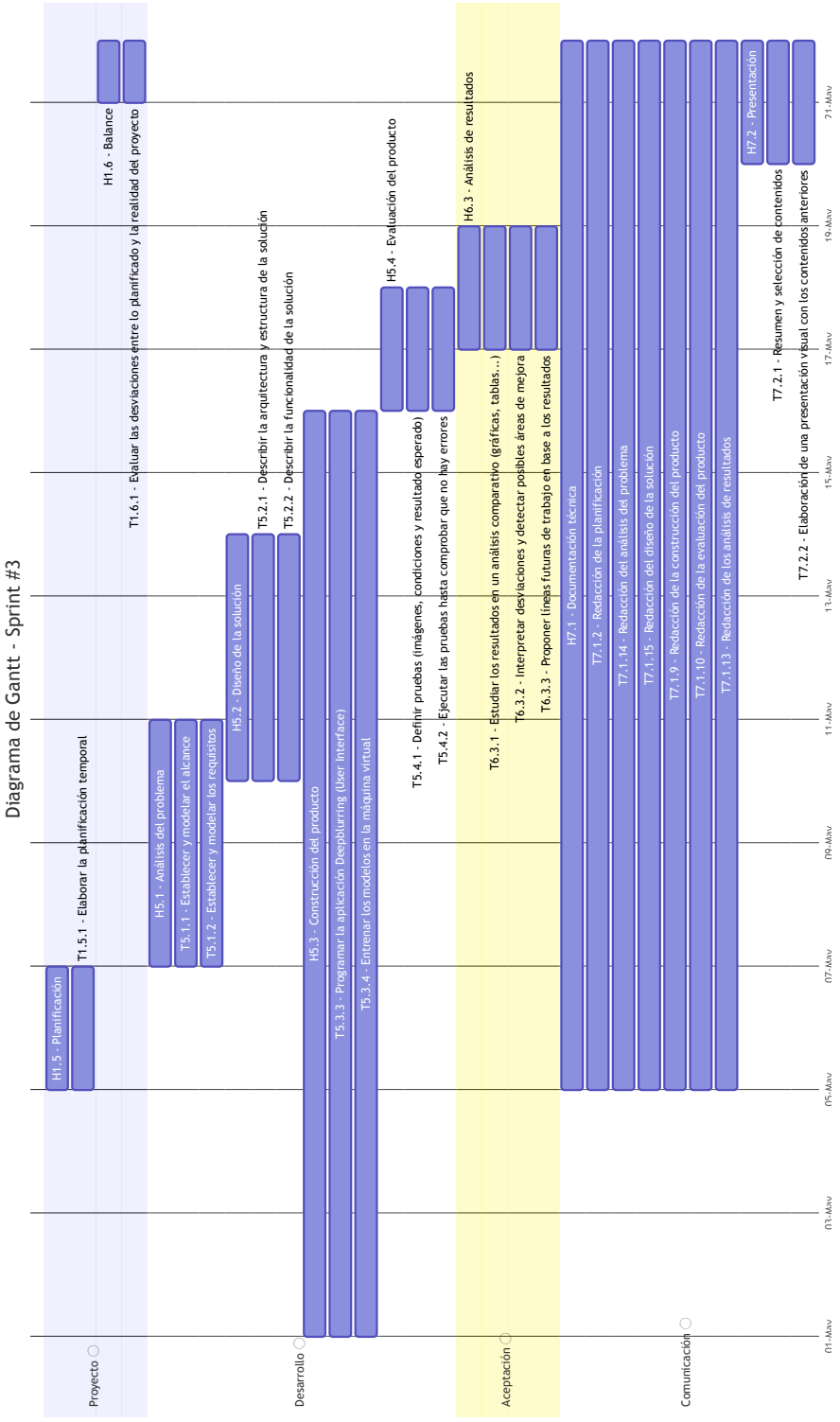


Figura 2.10: Diagrama de Gantt del *Sprint* #3

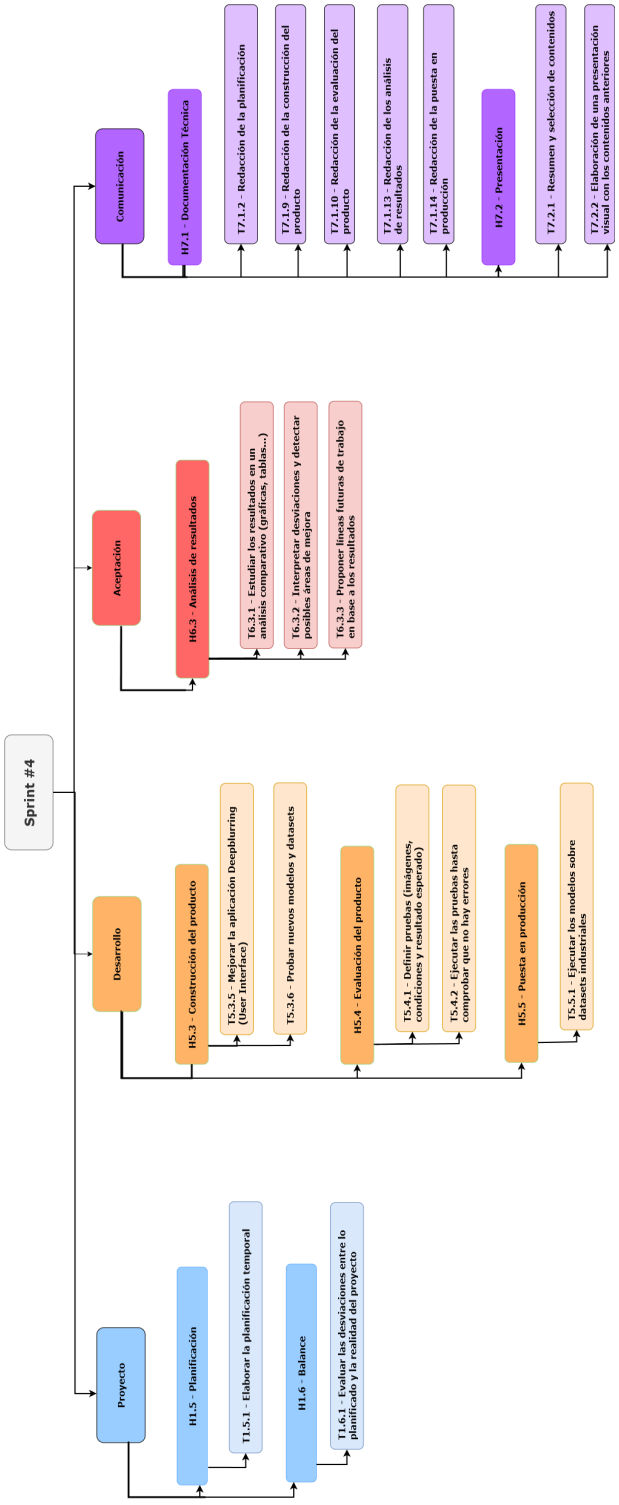


Figura 2.11: EDT del *Sprint* #4

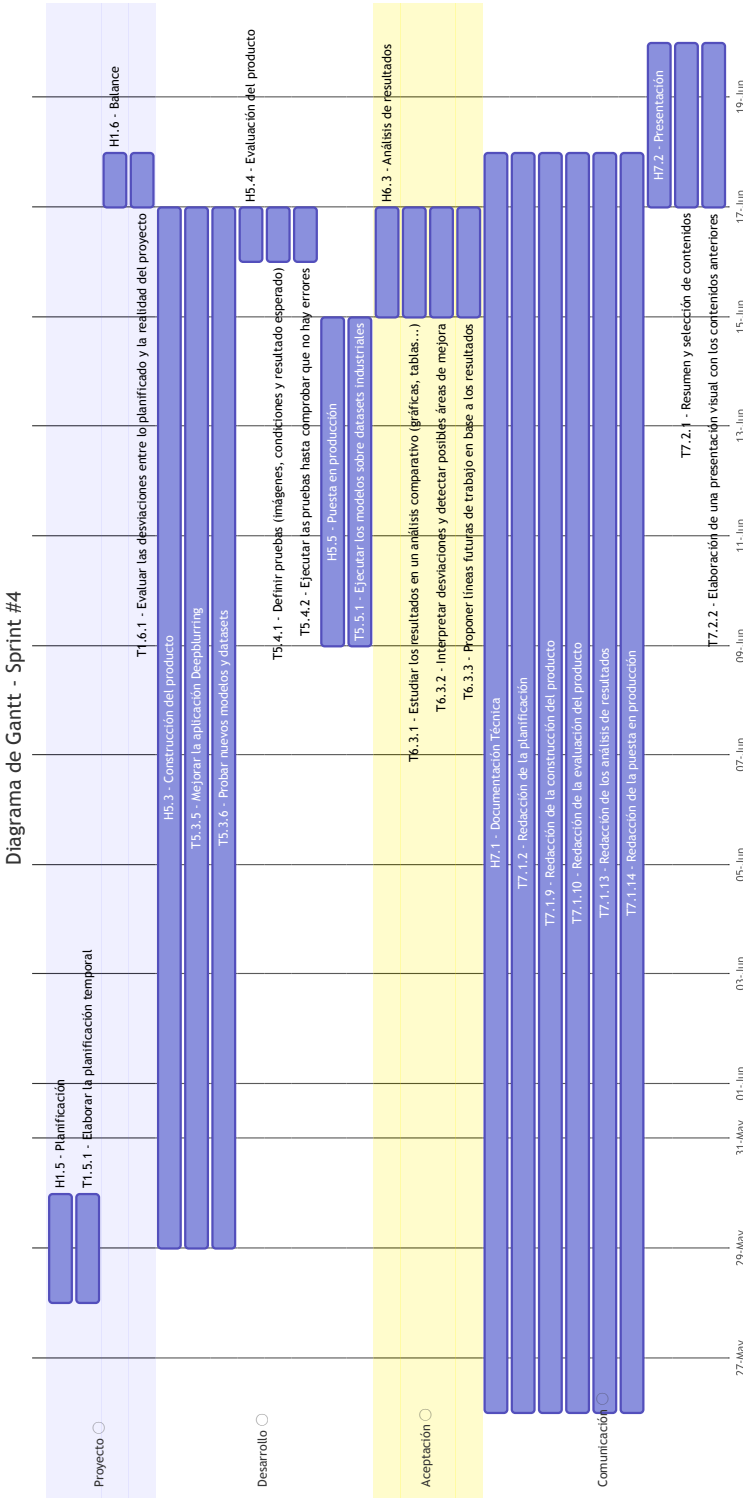


Figura 2.12: Diagrama de Gantt del *Sprint* #4

2.3. Presupuestos

En esta sección se presenta un análisis de los presupuestos necesarios para llevar a cabo el proyecto. El presupuesto es un elemento esencial en la planificación y desarrollo de cualquier proyecto, ya que facilita la estimación de los recursos necesarios para cumplir con los objetivos establecidos. En este sentido, se especifican los costos vinculados a los recursos de *hardware*, *software* y personal.

Hardware

El principal recurso *hardware* ha sido el ordenador personal Asus VivoBook con procesador AMD Ryzen 5, sistema operativo de 64 bits, 16GB de memoria RAM y 256GB de almacenamiento. Además, cuenta con una tarjeta gráfica integrada AMD Radeon Vega 8. El coste de este ordenador ha sido de 529 €, con una vida útil de 5 años. Para el uso de las herramientas de *software*, es esencial disponer de una conexión Wi-Fi rápida y estable. En este caso, se cuenta con un servicio de internet contratado, con un coste mensual de 9.95 euros.
Como se reflejará en la siguiente sección, uno de los riesgos posibles es que la GPU del ordenador personal sea limitada para el correcto desarrollo del proyecto, pues no es una GPU dedicada. Comparte memoria con el sistema y no tiene memoria propia, lo cual limita su capacidad a la hora de entrenar modelos de *Deep Learning*. Por este motivo, es razonable realizar una planificación previsora incluyendo una GPU más potente en la máquina virtual del departamento, una *Nvidia RTX A40*. El precio de esta máquina con GPU es de 11.200€, con una vida útil de 5 años. La Tabla 2.1 recoge los costes *hardware*.

Componente	Coste	% Uso mensual	Meses	Total
Ordenador personal	529 €	3,75 %	4 meses	79,35 €
Conexión Wi-Fi	9,95 €/mes	40 %	4 meses	15,92 €
Servidor con GPU	11.200,00 €	0,54 %	4 meses	241,92 €
Total				337,19 €

Tabla 2.1: Costes planificados del *hardware*

Software

Los programas que se utilizarán son de acceso libre o cuentan con licencias gratuitas. Se utilizará Overleaf para la redacción del informe, mientras que Microsoft Teams y Trello facilitarán el seguimiento, la comunicación y la organización. Adicionalmente, Google Colab proporcionará una GPU en caso de que el proyecto lo requiriese. Draw.io se empleará para confeccionar diagramas, generando representaciones visuales claras y atractivas. Para elaborar los diagramas de Gantt se hará uso de Gantt PRO. Se planea utilizar Python en Visual Studio como lenguaje principal de programación. El ordenador personal dispone de Windows 11 como sistema operativo. La Tabla 2.2 recoge los costes del *software*.

Componente	**Coste**	**Total**
Overleaf	0 €	0 €
Microsoft Teams	0 €	0 €
Trello	0 €	0 €
Draw.io	0 €	0 €
Visual Studio	0 €	0 €
Python	0 €	0 €
Google Colab	0 €	0 €
Windows 11	0 €	0 €
Total		0 €

Tabla 2.2: Costes planificados del *software*

Recursos humanos

Para el desarrollo de este proyecto, se ha asignado un equipo de trabajo con una duración total de 300 horas. En este caso, una única persona asume tres roles diferentes a lo largo del proceso, desempeñando funciones de gestor de proyecto, analista de datos, y científico de datos. A pesar de que los salarios pueden variar según la experiencia, ubicación y empresa, se ha calculado una media a partir de los datos obtenidos en [43], [44], y [45] a día 3 de marzo de 2025. A continuación, se detallan los roles asumidos por el estudiante y su retribución horaria:

1. **Gestor de Proyecto** Responsable de la organización, planificación y supervisión del desarrollo del proyecto. La retribución bruta media para este rol es de 17,63 € por hora.

2. **Analista de Datos** Encargado de la manipulación e interpretación de los datos utilizados en el estudio, evaluando la calidad de las imágenes y aplicando métricas para medir la efectividad de las técnicas de *deblurring*. Su salario medio es de 13,33 € por hora.

3. **Científico de Datos** Diseña y experimenta con modelos de *Deep Learning* para *deblurring*, seleccionando arquitecturas de redes neuronales y ajustando hiperparámetros para mejorar

el rendimiento. Engloba en su desempeño las funciones relacionadas con el desarrollo. Su retribución media por hora asciende a 18,27 €.

Adicionalmente, es importante considerar el coste asociado a la Seguridad Social, el cual asciende al 28,30 % del salario bruto, de los cuales un 23,60 % corresponde a la contribución del empleador y un 4,70 % al empleado [48]. Este coste debe ser tenido en cuenta para una estimación más realista del presupuesto total del proyecto. La Tabla 2.3 recoge el desglose de los costes necesarios en Recursos Humanos.

Puesto	Salario anual	Salario por hora	Horas Totales	Salario sin SS	SS (28.3 %)	Total
Project Manager	36.670,40 €	17,63 €/h	50h	881,50 €	249,45 €	1.130,95 €
Analista	27.726,40 €	13,33 €/h	90h	1.199,70 €	339,56 €	1.539,26 €
Científico de datos	38.001,60 €	18,27 €/h	160h	2.923,20 €	827,28 €	3.750,48 €
Total			300h	5.004,40 €	1.416,29 €	6.420,69 €

Tabla 2.3: Costes planificados de Recursos Humanos

Coste total

La Tabla 2.4 resume el coste total estimado para llevar a cabo este proyecto.

Componente	Coste
Hardware	337,19 €
Software	0 €
Recursos Humanos	6.420,69 €
Total	6.757,88 €

Tabla 2.4: Costes totales planificados

2.4. Gestión de riesgos

Llevar a cabo una gestión efectiva de los riesgos permite incrementar la probabilidad de obtener resultados favorables y de reducir los efectos adversos. Para realizar este proceso, es fundamental la identificación de riesgos, un análisis cualitativo considerando su probabilidad de ocurrencia y su impacto, seguido de un análisis cuantitativo. Además, es necesario diseñar planes de contingencia para mitigar amenazas [1], [21].

2.4.1. Identificación de riesgos

La Tabla 2.5 recoge los principales riesgos considerados en el proyecto.

Riesgos	
RI-01	El proyecto se retrasa con respecto a la planificación inicial
RI-02	Se posee insuficiente experiencia con las herramientas técnicas utilizadas
RI-03	Las técnicas de deblurring no consiguen su objetivo completamente
RI-04	La GPU utilizada no es lo suficientemente potente
RI-05	Posible dificultad en la replicabilidad de los resultados debido a la variabilidad en los modelos de *Deep Learning* y los datasets

Tabla 2.5: Lista de riesgos asociados al proyecto

2.4.2. Estimación de los riesgos

En la Tabla 2.6 y en la Tabla 2.7 se examina respectivamente la probabilidad de ocurrencia de cada riesgo y el impacto que este genera en caso de materializarse. Ambos factores se valoran en una escala de [0, 5]. La probabilidad de ocurrencia representa la posibilidad de que el riesgo se produzca si no se implementan medidas preventivas. Este valor está directamente relacionado con la probabilidad cuantificada en términos porcentuales, en un rango que va del 0 % al 100 %. Por otro lado, la pérdida se refiere a las pérdidas económicas directas derivadas de no mitigar un riesgo, mientras que el impacto en los costes mide el aumento de los costos totales en caso de que el riesgo se concrete.

Riesgo	Probabilidad	Valor [0-5]	Justificación
RI-01	25 %	2	Es razonable que eventualmente exista algún retraso respecto a la planificación debido a otras obligaciones del estudiante.
RI-02	80 %	4	En el Grado solo una asignatura imparte materia relativa a los sistemas inteligentes. Por lo que es bastante probable que no se disponga, inicialmente, de la experiencia necesaria.
RI-03	30 %	2	La eliminación completa del blur depende de numerosos factores. Podría ocurrir que esta mejora fuese parcial o menor de lo deseado.
RI-04	80 %	5	Las técnicas utilizadas suelen demandar una alta potencia de cómputo para ejecutarse en un tiempo razonablemente corto.
RI-05	40 %	2	Diferencias en los datasets pueden afectar la reproducibilidad de los modelos.

Tabla 2.6: Probabilidad de los riesgos

Riesgo	Pérdida	Impacto [0-5]	Justificación
RI-01	56,31 €por cada día de retraso	3	Un retraso en el proyecto podría implicar más tiempo de dedicación y afectar otras actividades académicas.
RI-02	56,31 €por cada día de retraso	3	La falta de experiencia puede ralentizar el avance del proyecto, pero no representa un impacto económico significativo.
RI-03	56,31 €por cada día de retraso	3	Si los resultados de deblurring no son satisfactorios, podría ser necesario explorar otras técnicas, aumentando costos y esfuerzo.
RI-04	241,92 €	5	La falta de *hardware* potente puede retrasar enormemente la ejecución de experimentos, generando sobrecarga en el cronograma.
RI-05	-	1	Los problemas de reproducibilidad tienen que tenerse en cuenta, pero al ser un proyecto de investigación su impacto financiero es menor.

Tabla 2.7: Impacto de los riesgos

2.4.3. Matriz de Probabilidad x Impacto

El producto Exposición = Probabilidad × Impacto permite calcular diferentes valores para clasificar los riesgos en función de su prioridad. En la Tabla 2.8 se recoge la exposición asociada a cada riesgo.

- **Prioridad alta**: $10 \leq$ Exposición
- **Prioridad media**: $5 \leq$ Exposición < 10
- **Prioridad baja**: Exposición < 5

Riesgo	Probabilidad (0-5)	Impacto (0-5)	Exposición
RI-01	2	3	6
RI-02	4	3	12
RI-03	2	3	6
RI-04	5	5	25
RI-05	2	1	2

Tabla 2.8: Matriz de Probabilidad x Impacto

Se puede observar que los riesgos de alta prioridad y que, por tanto, requerirían una acción correctiva son **RI-02** y **RI-04**.

2.4.4. Plan de contingencia

En la Tabla 2.9 se muestra el plan de contingencia para cada uno de los riesgos mencionados.

Riesgo	Plan de contingencia
R-01	Ajustar la planificación para que sea más realista y alineada con el ritmo de trabajo efectivo.
R-02	Dedicar más tiempo a la formación y adquisición de conocimientos esenciales para el desarrollo del proyecto.
R-03	Realizar un análisis exhaustivo de los resultados y conclusiones, independientemente de la mejora obtenida.
R-04	Utilizar una GPU externa para optimizar el entrenamiento del modelo.
R-05	Identificar y emplear un conjunto de datos alternativo que cumpla con las características necesarias.

Tabla 2.9: Plan de contingencia

2.5. Balance temporal y económico

En esta sección se realiza una comparación entre lo planificado y lo ejecutado en el marco temporal y en el marco económico. De este modo, podemos analizar la adecuación del desarrollo del proyecto a su planificación inicial.

2.5.1. Balance temporal

Para cada *sprint* presentamos un diagrama de Gantt que contiene, en color azul, las tareas planificadas. En caso de que alguna tarea se desviara de dicha planificación, se colorearán en rojo los días de la desviación.

Sprint #1

En la Figura 2.13 mostramos el diagrama de Gantt del *sprint* #1. Prácticamente todas las tareas se cumplieron en el plazo programado, a excepción de las tareas: **T1.5.2 Elaborar los presupuestos**, **T1.5.3 Elaborar el plan de riesgos y contingencias**, y **T2.3.2 Explicar arquitecturas para las técnicas de deblurring**, que se retrasaron un día cada una. En este *sprint* se destinaron 80 horas, lo cual está dentro del rango planificado (de 75 a 90 horas por *sprint*).

Sprint #2

En la Figura 2.14 mostramos el diagrama de Gantt del *sprint* #2. Prácticamente todas las tareas se cumplieron en el plazo programado, a excepción de la tarea: **T3.5.2 Implementar los modelos**, que se retrasó dos días más de lo planificado. En este *sprint* se destinaron 86 h, lo cual está dentro del rango planificado (de 75 a 90 horas por *sprint*).

Sprint #3

En este *sprint* no hubo ninguna desviación temporal, por lo que el desarrollo de las tareas coincide con su planificación (ver Figura 2.10). En este *sprint* se destinaron 82 horas, lo cual está dentro del rango planificado (de 75 a 90 horas por *sprint*).

Sprint #4

En este *sprint* tampoco hubo ninguna desviación temporal, por lo que el desarrollo de las tareas coincide con su planificación (ver Figura 2.12). En este *sprint* se destinaron 76 horas, lo cual está dentro del rango planificado (de 75 a 90 horas por *sprint*).

Balance final

Teniendo en cuenta que este Trabajo Fin de Grado tiene una carga de trabajo de 12 créditos ECTS, y que cada crédito equivale de 25 a 30 horas, el trabajo dedicado a este proyecto debería oscilar entre 300 y 360 horas de trabajo. Sumando las horas trabajadas en los cuatro *sprints*, observamos que el total es de 324 horas, lo cual entra dentro del rango esperado.

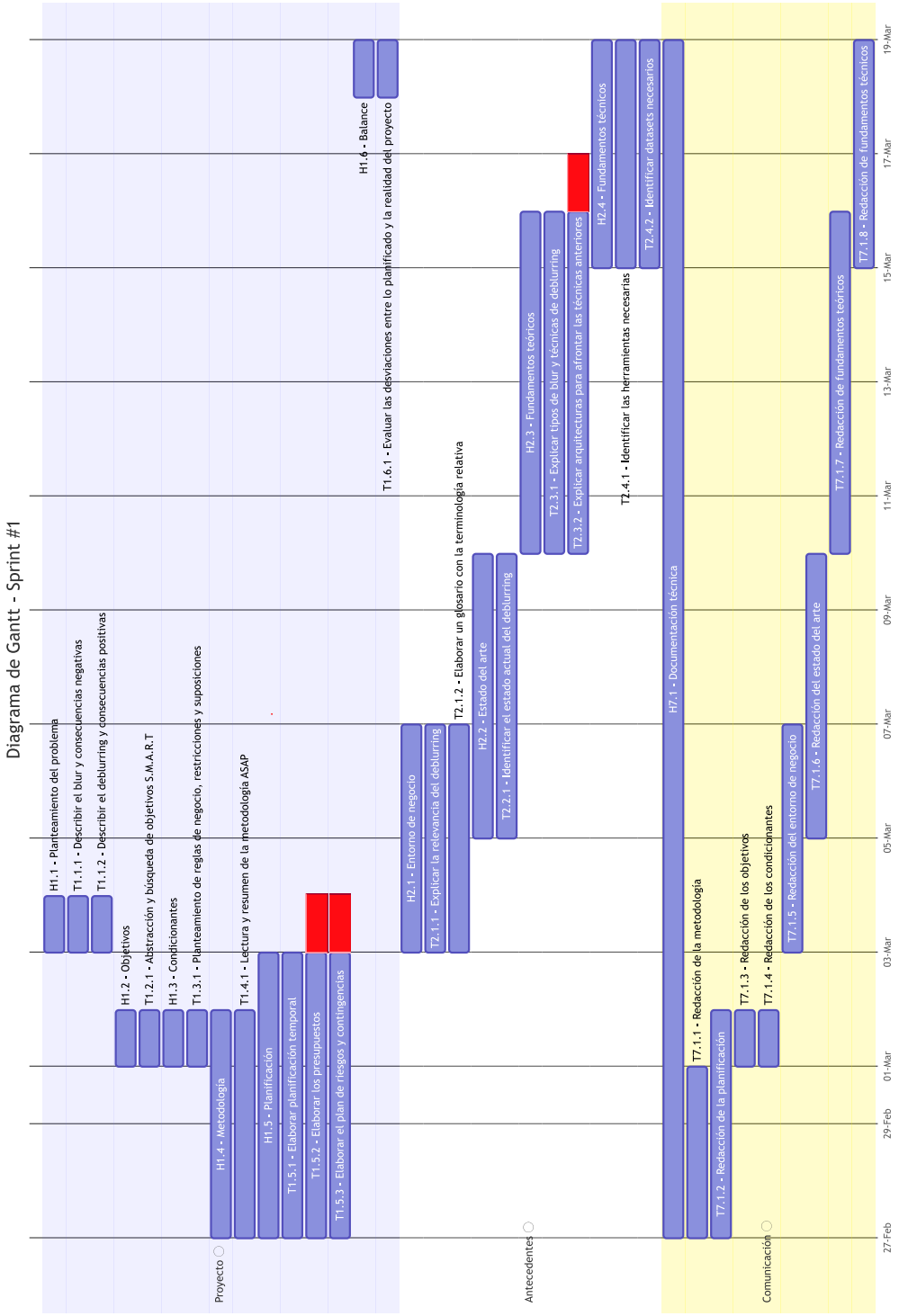


Figura 2.13: Ejecución real de las tareas del *Sprint* #1

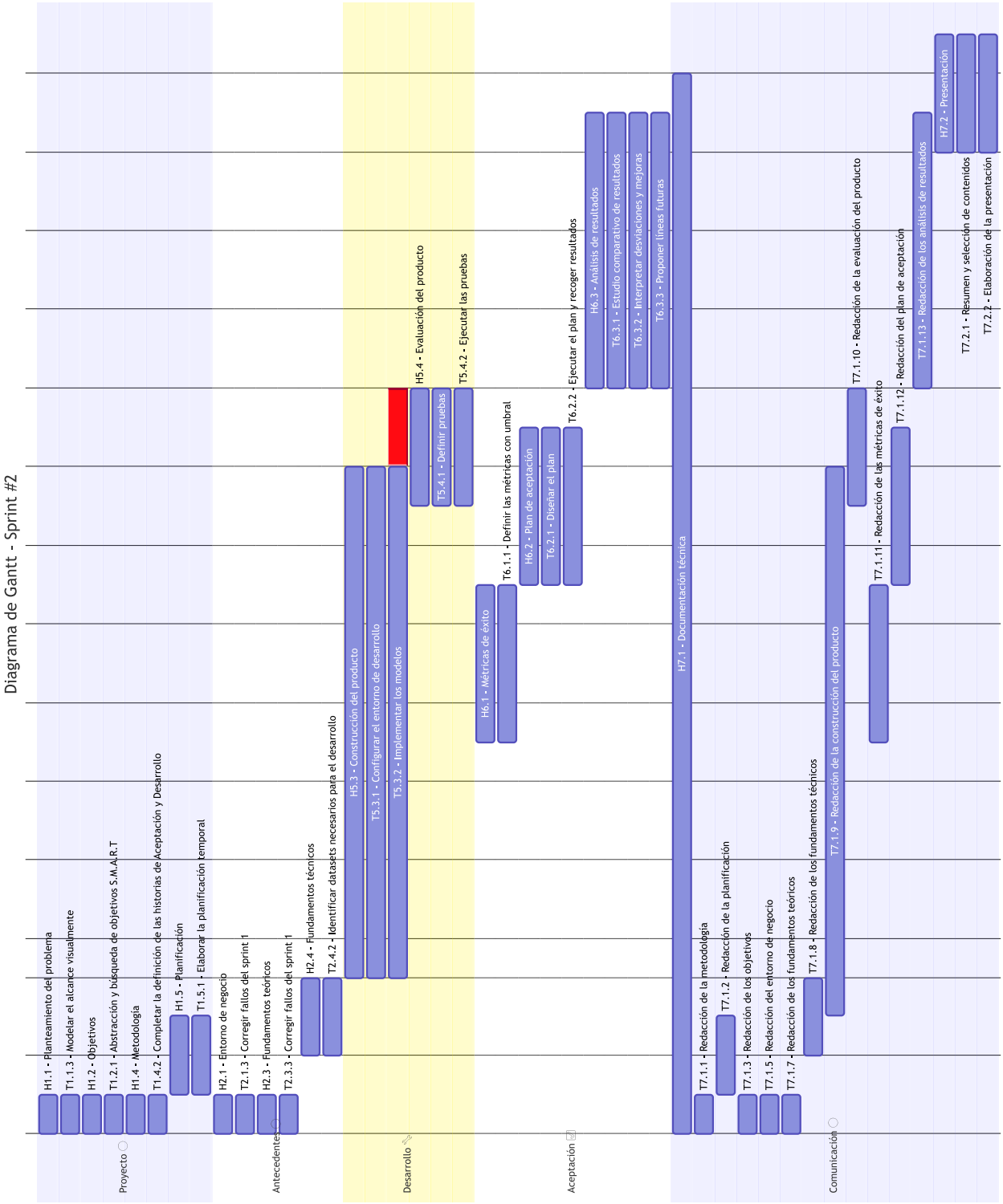


Figura 2.14: Ejecución real de las tareas del *Sprint* #2

2.5.2. Balance económico

Sprint #1

Como se ha mencionado, durante el *sprint* #1 se ha dedicado un total de 80 horas de trabajo. En la Tabla 2.10 se desglosa el número de horas trabajadas por cada rol junto con su coste asociado.

Puesto	Salario por hora	Horas Totales	Salario sin SS	SS (28.3 %)	Total
Project Manager	17,63 €/h	21h	370,23 €	104,78 €	475,01 €
Analista	13,33 €/h	49h	652,17 €	184,58 €	836,75 €
Científico de datos	18,27 €/h	10h	182,70 €	51,70 €	234,40 €
Total		80h	1.205,10 €	341,06 €	1.546,16 €

Tabla 2.10: Costes de Recursos Humanos durante el *sprint* #1

Cabe mencionar que en este *sprint* no ha sido necesaria una GPU adicional, por lo que los costes *hardware* han sido menores de lo planificado. La Tabla 2.11 resume el coste total de los recursos empleados durante el *sprint* #1.

Componente	Coste
Hardware	29,78 €
Software	0 €
Recursos Humanos	1.546,16 €
Total	1.575,94 €

Tabla 2.11: Costes totales del *sprint* #1

Sprint #2

Como se ha mencionado, durante el *sprint* #2 se ha dedicado un total de 86h de trabajo. En la Tabla 2.12 se desglosa el número de horas trabajadas por cada rol junto con su coste asociado.

Puesto	Salario por hora	Horas Totales	Salario sin SS	SS (28.3 %)	Total
Project Manager	17,63 €/h	24h	423,12 €	119,74 €	542,86 €
Analista	13,33 €/h	28h	373,24 €	105,64 €	478,88 €
Científico de datos	18,27 €/h	34h	621,18 €	175,79 €	796,97 €
Total		86h	1.417,54 €	401,17 €	1.818,71 €

Tabla 2.12: Costes de Recursos Humanos durante el *sprint* #2

Cabe mencionar que en este *sprint* sí ha sido necesaria una GPU adicional, por lo que los costes *hardware* han sido ligeramente superiores a lo planificado. La Tabla 2.13 resume el coste

total de los recursos empleados durante el *sprint* #2.

Componente	Coste
Hardware	90,26 €
Software	0 €
Recursos Humanos	1.818,71 €
Total	1.908,97 €

Tabla 2.13: Costes totales del *sprint* #2

Sprint #3

Como se ha mencionado, durante el *sprint* #3 se ha dedicado un total de 82 horas de trabajo. En la Tabla 2.14 se desglosa el número de horas trabajadas por cada rol junto con su coste asociado.

Puesto	Salario por hora	Horas Totales	Salario sin SS	SS (28.3 %)	Total
Project Manager	17,63 €/h	25h	440,75 €	124,74 €	565,49 €
Analista	13,33 €/h	21h	279,93 €	79,21 €	359,14 €
Científico de datos	18,27 €/h	36h	657,72 €	186,13 €	843,85 €
Total		82h	1.378,40 €	390,08 €	1.768,48 €

Tabla 2.14: Costes de Recursos Humanos durante el *sprint* #3

Cabe mencionar que en este *sprint* también ha sido necesaria una GPU adicional, por lo que los costes *hardware* han sido ligeramente superiores a lo planificado. La Tabla 2.15 resume el coste total de los recursos empleados durante el *sprint* #3.

Componente	Coste
Hardware	90,26 €
Software	0 €
Recursos Humanos	1.768,48 €
Total	1.858,74 €

Tabla 2.15: Costes totales del *sprint* #3

Sprint #4

Como se ha mencionado, durante el *sprint* #4 se ha dedicado un total de 76 horas de trabajo. En la Tabla 2.16 se desglosa el número de horas trabajadas por cada rol junto con su coste asociado.

Puesto	Salario por hora	Horas Totales	Salario sin SS	SS (28.3 %)	Total
Project Manager	17,63 €/h	13h	229,19 €	64,87 €	294,06 €
Analista	13,33 €/h	32h	426,56 €	120,70 €	547,26 €
Científico de datos	18,27 €/h	31h	566,37 €	160,27 €	726,64 €
Total		76h	1.222,12 €	345,84 €	1.567,96 €

Tabla 2.16: Costes de Recursos Humanos durante el *sprint* #4

Cabe mencionar que en este *sprint* no ha sido necesaria una GPU adicional, por lo que los costes *hardware* han sido menores a lo planificado. La Tabla 2.17 resume el coste total de los recursos empleados durante el *sprint* #4.

Componente	Coste
Hardware	29,78 €
Software	0 €
Recursos Humanos	1.567,96 €
Total	1.597,74 €

Tabla 2.17: Costes totales del *sprint* #4

Balance final

La Tabla 2.18 resume el coste total de los recursos empleados a lo largo de todo el proyecto.

Componente	Coste
Hardware	300,56 €
Software	0 €
Recursos Humanos	6.640,83 €
Total	6.941,39 €

Tabla 2.18: Costes totales del proyecto

Como se mostró en la Tabla 2.4, el coste total planificado fue de 6.757,88 €. El coste real del proyecto ha sido de 6.941,39 €, lo que representa un 2,72 % más de lo planificado. Esta diferencia surge de que en la planificación se estimaron 300 horas de trabajo, pero las horas trabajadas reales han sido 324. Esta diferencia se debe a que en la planificación se estimaron 300 horas de trabajo, pero las horas reales trabajadas ascendieron a 324. A pesar de esta ligera desviación, la planificación fue bastante acertada, especialmente considerando que se materializaron algunos riesgos, como la necesidad de una GPU adicional. Aunque este gasto ya estaba contemplado, dicho riesgo también implicó un aumento en las horas de trabajo, lo que incrementó el coste final.

Capítulo 3

Antecedentes

Este capítulo tiene como objetivo establecer el contexto en el que se enmarca este proyecto. En él, se busca comprender las bases teóricas y técnicas que sustentan el desarrollo del proyecto, identificando el estado actual del problema a tratar, las principales necesidades, oportunidades y desafíos a abordar.

3.1. Entorno de negocio

Para llevar a cabo el desarrollo del proyecto de manera efectiva, es fundamental comprender el entorno de negocio en el que se enmarca. Esta sección tiene como propósito tener una visión general del *deblurring*, así como de conocer el entorno final en el que se planea aplicar las técnicas de *deblurring*, que es la Industria 4.0.

Deblurring

El *deblurring* es el proceso de restauración de imágenes borrosas mediante la eliminación del desenfoque causado por movimiento, condiciones ambientales o limitaciones en los sensores. Como se ha mencionado anteriormente, este problema es común en diversos campos donde la calidad de la imagen es fundamental para la toma de decisiones y el análisis de datos.

Entre las principales áreas de aplicación del *deblurring* se encuentran:

- **Medicina:** En imágenes médicas como resonancias magnéticas o tomografías, el desenfoque puede afectar la precisión en el diagnóstico. Las técnicas de *deblurring* mejoran la nitidez y permiten una mejor identificación de estructuras anatómicas y patologías [20].

- **Seguridad y vigilancia:** En sistemas de videovigilancia, el desenfoque debido a movimientos rápidos o malas condiciones de iluminación puede dificultar la identificación de personas o eventos críticos. Aplicar *deblurring* mejora la calidad de las imágenes captadas por cámaras de seguridad [56].

- **Astronomía:** La captura de imágenes espaciales se ve afectada por la atmósfera terrestre y los movimientos de los telescopios. Las técnicas de *deblurring* permiten mejorar la nitidez de las imágenes y obtener datos más precisos para el análisis astronómico [3].

- **Fotografía y restauración de imágenes:** En la fotografía digital y la recuperación de documentos históricos, el *deblurring* es utilizado para mejorar la calidad visual y restaurar detalles perdidos en imágenes antiguas o deterioradas [33].

En los últimos años, las técnicas basadas en *Deep Learning* han revolucionado el campo del *deblurring*. Los métodos tradicionales como la deconvolución y los filtros de optimización han sido superados por modelos avanzados de redes neuronales, que aprenden directamente a partir de grandes volúmenes de datos degradados y nítidos. Ciertas arquitecturas como las redes neuronales convolucionales (CNN), y las redes generativas adversarias (GAN) han demostrado resultados superiores en la restauración de imágenes. Estas técnicas permiten no solo mejorar la nitidez de una imagen, sino también recuperar detalles perdidos de forma más efectiva y adaptarse a distintos tipos de desenfoque [64]. Un ejemplo de imagen a la que se le ha eliminado el *blur* es la Figura 3.1.

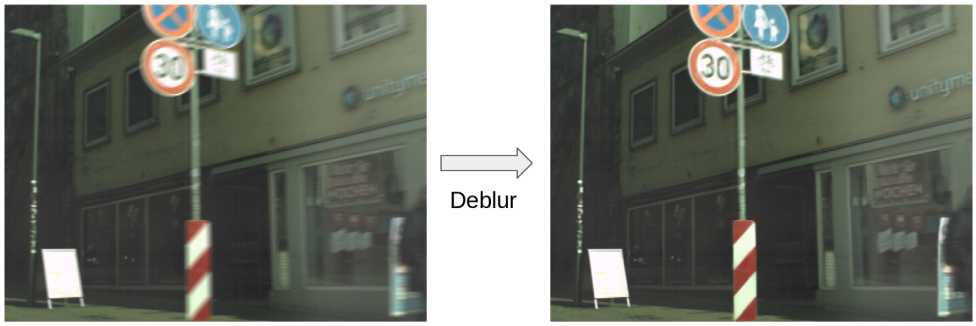


Figura 3.1: Imagen con blur vs imagen sin blur (imagen de [31])

Industria 4.0

La Industria 4.0 hace referencia a la transformación digital en el ámbito industrial, caracterizada por la integración de tecnologías avanzadas como el Internet de las Cosas (*Internet of Things*), la inteligencia artificial (IA), la robótica, la computación en la nube y el análisis de datos. Este paradigma busca optimizar la producción, mejorar la eficiencia operativa y permitir una mayor personalización de los productos mediante la interconectividad de sistemas y el procesamiento inteligente de la información [59].
En este contexto, la Industria 4.0 ha impulsado la automatización y digitalización en sectores como la manufactura, la logística y el mantenimiento predictivo. La adquisición y análisis de imágenes juegan un papel fundamental en la detección de fallos, la supervisión de equipos y la mejora de la eficiencia operativa. Uno de los pilares fundamentales de la Industria 4.0 es la visión computacional, utilizada en tareas como inspección de calidad, monitoreo de fallas, control de procesos y mantenimiento predictivo. Sin embargo, la precisión de estos sistemas depende en gran medida de la calidad de las imágenes adquiridas. Algunos factores como vibraciones en la maquinaria, condiciones de iluminación, interferencias térmicas y movimientos rápidos de los sensores o piezas pueden generar imágenes borrosas, lo que dificulta su procesamiento y análisis. Las técnicas de *deblurring* basadas en *Deep Learning* ofrecen una solución innovadora para mejorar la nitidez de estas imágenes y optimizar la toma de decisiones en entornos industriales. Entre sus principales beneficios se encuentran:

- **Mayor precisión en la inspección de calidad:** En líneas de producción, la detección de defectos en piezas mediante visión artificial requiere imágenes nítidas. La eliminación del desenfoque mejora la capacidad de los algoritmos para identificar grietas, deformaciones o impurezas en los productos.
- **Optimización del mantenimiento predictivo:** En el monitoreo de maquinaria industrial, las imágenes borrosas pueden dificultar la detección temprana de fallas mecánicas. Aplicando *deblurring*, se pueden mejorar los diagnósticos y evitar costosos tiempos de inactividad.
- **Mejor análisis en imágenes termográficas:** En industrias como la metalurgia o la generación de energía, las cámaras térmicas son utilizadas para monitorear temperaturas y detectar anomalías. Reducir el desenfoque en estas imágenes mejora la interpretación de datos críticos para la seguridad y eficiencia operativa.
- **Reducción de errores en robótica y automatización:** En sistemas de visión utilizados por robots industriales, el desenfoque puede afectar la precisión en la manipulación de objetos y en la navegación autónoma. Aplicar *deblurring* mejora la fiabilidad de estos sistemas y permite una mejor integración de la robótica en la manufactura avanzada.

Stakeholders y sus necesidades

En el contexto del proyecto, centrado en la aplicación de técnicas de *deblurring* mediante *Deep Learning* para la mejora de imágenes industriales, se identifican los siguientes *stakeholders* clave:

- **Empresas industriales.** Son los actores principales que se beneficiarán directamente de la mejora en la calidad de las imágenes. Utilizan sistemas de inspección visual para tareas como el control de calidad, el mantenimiento predictivo o la automatización de procesos. Dentro de las empresas, se pueden distinguir dos roles: ingenieros, que representan un enfoque más técnico, y responsables de calidad y producción, que representan un enfoque más orientado al negocio.
  Necesidades: Imágenes nítidas y confiables que permitan una detección precisa de fallos o anomalías.
  Expectativas: Soluciones eficientes y escalables que puedan integrarse fácilmente en entornos productivos, mejorando la precisión sin comprometer los tiempos de procesamiento.
  - **Ingenieros y técnicos de mantenimiento.** Son responsables del diagnóstico de fallos, supervisión de maquinaria y operación de sistemas basados en visión computacional.
    Necesidades: Herramientas que les permitan interpretar mejor las imágenes captadas por sensores o cámaras industriales.
    Expectativas: Modelos de *deblurring* que reduzcan el margen de error en el análisis visual y faciliten la toma de decisiones.
  - **Responsables de calidad y producción.** Toman decisiones estratégicas basadas en datos y resultados proporcionados por sistemas automatizados.
    Necesidades: Información visual clara y precisa que respalde acciones correctivas o preventivas.
    Expectativas: Mejora de la eficiencia, reducción de productos defectuosos y optimización de recursos gracias a imágenes mejoradas.

- **Desarrolladores de soluciones basadas en IA.** Profesionales encargados de diseñar e integrar modelos de *Deep Learning* en plataformas de organizaciones, en este caso, del sector industrial.
  Necesidades: Algoritmos robustos, entrenables con datasets del entorno industrial, adaptables a distintos dispositivos y tipos de desenfoque.
  Expectativas: Modelos de alto rendimiento que puedan integrarse en arquitecturas existentes y ofrecer resultados en tiempo real o cuasi real.

- **Clientes finales o usuarios del producto industrial.** Aunque no interactúan directamente con las imágenes, se benefician indirectamente de la mejora en calidad y fiabilidad de los productos fabricados.
  Necesidades y expectativas: Productos sin defectos, procesos más controlados y mayor fiabilidad en sectores como la automoción, energía o bienes de consumo.

El desarrollo de modelos de *Deep Learning* para la restauración de imágenes representa una oportunidad significativa para estos entornos industriales. No obstante, aún existen desafíos como la necesidad de entrenar modelos con grandes volúmenes de datos, la optimización del procesamiento en tiempo real y la adaptación de las técnicas de *deblurring* a diferentes tipos de cámaras y condiciones ambientales.
Para complementar el entorno de negocio, se añade un glosario con conceptos relevantes dentro de este proyecto (ver Apéndice A).

3.2. Fundamentos teóricos

Esta sección tiene como objetivo explicar los conceptos y principios que son relevantes para un adecuado entendimiento del proyecto.

3.2.1. Inteligencia Artificial

La inteligencia artificial es un campo de la informática que se enfoca en crear sistemas que puedan realizar tareas que normalmente requieren inteligencia humana, como el aprendizaje, el razonamiento y la percepción [12]. A lo largo de la historia se han considerado 4 enfoques distintos [42]:

- **Sistemas que actúan como humanos:** Se centran en desarrollar sistemas capaces de imitar el comportamiento humano en tareas como el procesamiento del lenguaje, la toma de decisiones y la interacción social. Su objetivo es crear programas que pasen la Prueba de Turing, demostrando un comportamiento indistinguible del de un ser humano.

- **Sistemas que piensan como humanos** Buscan modelar el proceso de pensamiento humano para comprender cómo razona una persona y replicar dicho razonamiento en una máquina. Esto implica el estudio de la cognición, la percepción y el aprendizaje desde una perspectiva computacional.

- **Sistemas que piensan racionalmente** Se enfocan en la lógica formal y en la aplicación de principios matemáticos para modelar el razonamiento racional. Este enfoque se basa en la teoría de la inferencia lógica y el procesamiento simbólico para resolver problemas mediante reglas estrictas y algoritmos formales.

- **Sistemas que actúan racionalmente** Su objetivo es diseñar agentes inteligentes que tomen decisiones óptimas para alcanzar un objetivo en un entorno determinado. Estos sistemas utilizan técnicas de aprendizaje automático, optimización y planificación para actuar de manera autónoma de acuerdo con principios racionales y eficientes.

3.2.2. Machine Learning

El *Machine Learning* o Aprendizaje Automático es una rama de la Inteligencia Artificial que se enfoca en el diseño y desarrollo de algoritmos que permiten a los sistemas computacionales optimizar su desempeño en la ejecución de tareas basándose en la experiencia previa. Se puede emplear en diferentes tareas, como clasificación, predicción, clustering, o reconocimiento y generación de patrones. Dentro del *Machine Learning*, es posible identificar cuatro áreas [5]:

1. **Aprendizaje supervisado:** El algoritmo entrena con conjunto de datos previamente etiquetados. Estas etiquetas proporcionan información sobre la clase a la que pertenece cada dato, lo que significa que la salida esperada del algoritmo es conocida cuando se introduce un determinado dato de entrada. Entre los algoritmos más representativos de esta categoría se encuentran las redes neuronales, el clasificador de Naive Bayes, la regresión lineal, las máquinas de Vectores Soporte (SVM), el algoritmo de los K vecinos más cercanos (KNN) y los Bosques Aleatorios. Dentro del aprendizaje supervisado, se pueden distinguir dos subcategorías principales:

   - **Algoritmos de clasificación:** Se encargan de asignar a cada elemento del conjunto de datos una categoría dentro de un conjunto predefinido de clases. Estos algoritmos aprenden a partir de un conjunto de datos de entrenamiento, identificando patrones comunes en cada grupo para posteriormente clasificar datos nuevos no vistos previamente.
   - **Algoritmos de regresión:** En este caso, la variable de salida que se busca predecir es de tipo numérico y continuo. Estos algoritmos son útiles para determinar la relación de dependencia existente, o no, entre las diferentes variables de los datos analizados.

2. **Aprendizaje no supervisado:** En este enfoque, el algoritmo recibe como entrada un conjunto de datos que no cuentan con etiquetas previas. Su objetivo principal es identificar patrones y estructuras dentro de los datos sin intervención humana. Destacan dos subcategorías:

   - **Algoritmos de clustering:** Se basan en la agrupación de elementos similares dentro del conjunto de datos en grupos denominados clústeres, estableciendo así relaciones en función de sus semejanzas o diferencias. Algunos algoritmos representativos de este tipo incluyen el método de *k-means* o k-medias, así como los modelos de mezcla gaussiana.
   - **Algoritmos de reducción de la dimensionalidad:** Estos algoritmos buscan disminuir el número de características o dimensiones del conjunto de datos sin perder información relevante. Entre las técnicas más utilizadas en este ámbito se encuentran el Análisis de Componentes Principales (PCA), y la Descomposición en Valores Singulares (SVD).

3. **Aprendizaje por refuerzo:** El algoritmo se entrena mediante la interacción con un entorno y la obtención de recompensas en función de sus acciones. El sistema conoce las posibles acciones que puede realizar y las consecuencias de cada una, lo que le permite, a través de un proceso de ensayo y error, seleccionar aquellas decisiones que maximicen su beneficio.

4. **Otros tipos de aprendizaje**

   - **Aprendizaje semisupervisado:** El algoritmo se entrena tanto con datos etiquetados como con datos sin etiquetar.
   - **Aprendizaje no balanceado:** Es un caso particular de algoritmos de clasificación. Estos algoritmos utilizan una distribución de datos no balanceada entre las clases de interés.
   - **Aprendizaje multietiqueta:** Es un caso particular de algoritmos de clasificación. Cada instancia de los datos pertenece a un subconjunto de clases de interés, en lugar de a una sola.
   - **Aprendizaje por transferencia:** Utiliza patrones o modelos previamente entrenados para otro problema.

3.2.3. Redes Neuronales

Las redes neuronales son modelos matemáticos que tratan de imitar el comportamiento de la neurona biológica y la estructura del cerebro para la identificación de patrones y la toma de decisiones. El desarrollo de este concepto tuvo sus inicios con McCulloch y Pitts en 1943 [30], cuando publicaron un estudio sobre cómo el cerebro lograba detectar patrones mediante el uso de neuronas, estableciendo una conexión con la lógica booleana.
Desde una perspectiva biológica, las neuronas poseen tres partes principales: dendritas, cuerpo y axón (ver Figura 3.2).

- **Dendritas:** Son las estructuras ramificadas de la neurona que se encargan de recibir los impulsos eléctricos provenientes de otras neuronas. Su función principal es captar y transmitir estas señales al cuerpo celular para su procesamiento.

- **Cuerpo celular o soma:** Es la parte central de la neurona donde se encuentra el núcleo y otros orgánulos esenciales para su funcionamiento. Es responsable de procesar la información recibida por las dendritas y generar una respuesta adecuada.

- **Axón:** Es una estructura alargada que permite la transmisión de impulsos eléctricos desde el cuerpo celular hacia otras neuronas o células. A través del axón, las neuronas envían señales eléctricas que facilitan la comunicación en el sistema nervioso.

La sinapsis es el mecanismo mediante el cual las neuronas intercambian señales para transmitir información. La red neuronal biológica establece nuevas conexiones o ajusta las ya existentes a medida que aprende.

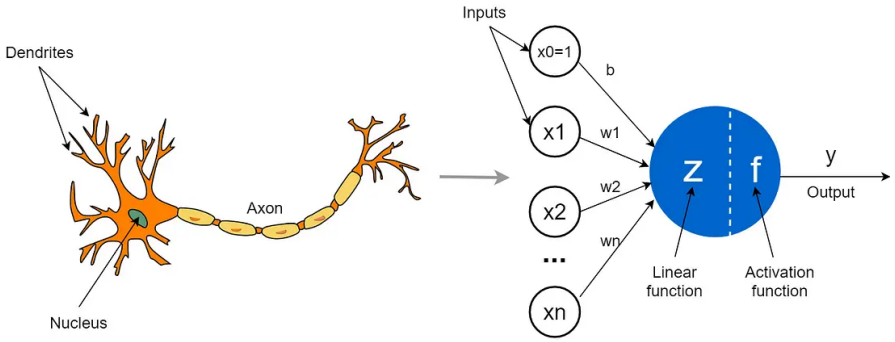


Figura 3.2: Neurona biológica y neurona artificial (imagen de [13])

Para entender el funcionamiento de las redes neuronales, es necesario conocer su unidad básica: la neurona artificial. Esta neurona parte del modelo del Perceptrón [41]. Describamos su estructura para comprobar las similitudes con la neurona biológica:

- **Un conjunto de entradas binarias** $x_1, x_2, ..., x_n$: reciben información externa, correspondiéndose con las dendritas de la neurona biológica.
- **Un conjunto de pesos** $w_1, w_2, ..., w_n$: ponderan las entradas binarias, correspondiéndose con el proceso de sinapsis.
- **Un sesgo o *bias*** b. Su función es activar la neurona, incluso cuando todas las entradas son cero, permitiendo que la red aprenda patrones más flexibles y tome mejores decisiones. De este modo, la neurona no depende solo de las entradas.
- **Función de propagación** $\sum$: función lineal que representa el producto escalar del vector de entradas binarias con el vector de pesos, añadiéndole el sumando b.

$$\Sigma = \sum_{i=1}^{n} x_i w_i + b$$

- **Función de activación escalón f**: devuelve 0 o 1 en función de si el resultado de la función de propagación es negativo o positivo, limitando la amplitud de la salida de la neurona y correspondiéndose con el axón de la neurona biológica.
- **Salida *y***: representa el valor devuelto por la función de activación **f**.

La neurona artificial es una generalización del Perceptrón, realizando ciertos cambios para relajar las condiciones de su definición. Uno de estos cambios es emplear una función de activación diferente a la función escalón. En la Figura 3.3 se muestran las funciones de activación más utilizadas.

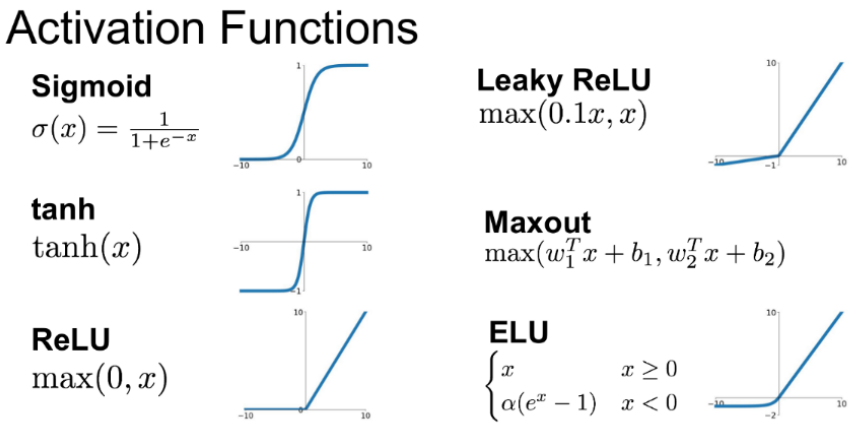


Figura 3.3: Principales funciones de activación para la neurona artificial (imagen de [22])

Una red neuronal se construye con un conjunto de neuronas artificiales que se distribuyen en diferentes capas interconectadas entre sí. Existen tres tipos de capas [6] (ver Figura 3.4):

- **Capa de entrada**: es aquella en la que se sitúan las neuronas que toman la entrada del exterior de la red.
- **Capa de salida**: es la capa de las neuronas cuya salida sale al exterior de la red.
- **Capas ocultas**: son aquellas que no son de entrada ni de salida.

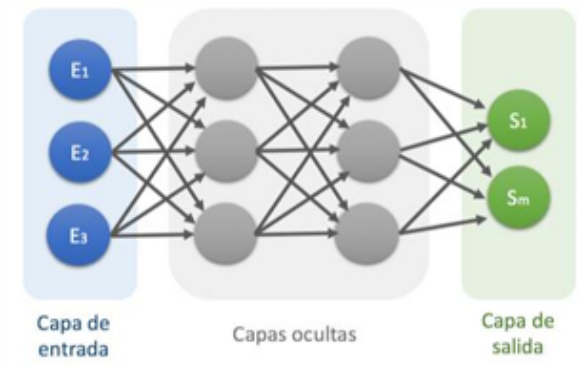


Figura 3.4: Capas de una red neuronal (imagen de [6])

Tras lo anterior, podemos definir las redes neuronales artificiales como modelos inspirados en la estructura y funcionamiento del cerebro humano. Su propósito principal es transformar un conjunto de entradas en una salida determinada, representando matemáticamente esta relación como una función. Para ello, cada conexión entre neuronas está ponderada por un conjunto de parámetros llamados pesos y un sesgo, los cuales se ajustan mediante un proceso llamado entrenamiento. El entrenamiento de la red es fundamental para su correcto funcionamiento. En el caso del aprendizaje supervisado, este proceso se basa en la utilización de datos etiquetados para optimizar los parámetros de la red. Inicialmente, los pesos y el sesgo de cada neurona se

establecen aleatoriamente. Luego, la red procesa los datos y genera una salida, la cual se compara con los valores reales mediante una función de pérdida.

Para reducir el error, es decir, reducir el valor de la función de pérdida, se utilizan algoritmos de optimización. Uno de los más comunes es el descenso del gradiente [6]. Este método ajusta iterativamente los pesos y sesgos para minimizar la función de pérdida, permitiendo que la red mejore su capacidad de generalización. Para el cálculo de los gradientes se suele utilizar la retropropagación, que se distribuye el error a través de las capas de la red, refinando los parámetros mediante múltiples iteraciones, conocidas como épocas.

Es importante mencionar que el objetivo del entrenamiento no es memorizar los datos de entrada, sino poder generalizar el conocimiento adquirido para clasificar correctamente nuevas entradas. Si la red se ajusta en exceso a los datos de entrenamiento, puede producirse sobreajuste (*overfitting*), lo que limita su capacidad de generalización. Por ello, se establecen criterios de parada para el entrenamiento, asegurando un equilibrio entre precisión y generalización.

3.2.4. Deep Learning

El *Deep Learning* [14], o Aprendizaje Profundo, es un subcampo del *Machine Learning* que se basa en el uso de redes neuronales artificiales con múltiples capas para aprender patrones complejos a partir de grandes volúmenes de datos. Su principal ventaja radica en la capacidad de extraer automáticamente características relevantes de los datos sin necesidad de intervención manual, lo que lo hace particularmente útil en tareas como visión computacional, procesamiento del lenguaje natural y reconocimiento del habla.
Las redes neuronales profundas (*Deep Neural Networks*) están estructuradas en capas jerárquicas. La primera capa aprende representaciones básicas de los datos, como por ejemplo bordes en una imagen, y a medida que la información avanza por la red, se combinan estas representaciones para formar conceptos más abstractos y complejos. Este principio de jerarquización permite que los modelos de *Deep Learning* sean altamente eficaces en la identificación de patrones, aunque a costa de un alto requerimiento computacional (ver Figura 3.5).

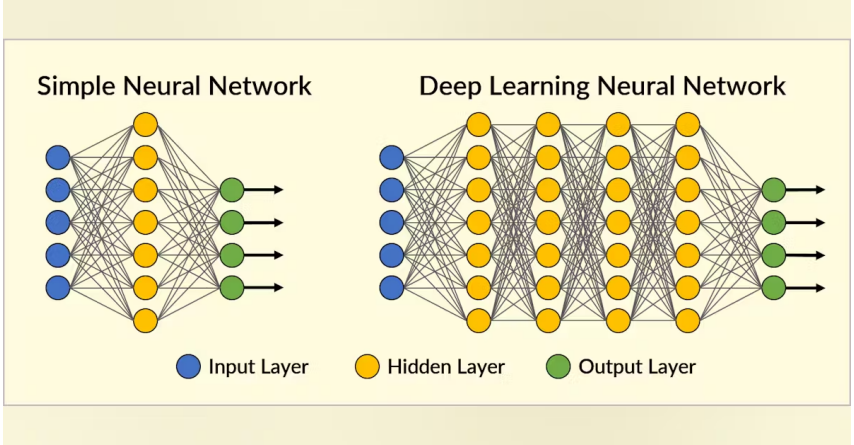


Figura 3.5: Red neuronal profunda (imagen de [2])

A continuación presentamos brevemente las redes neuronales profundas más comunes dentro del *Deep Learning*, y que se emplean a la hora de enfrentar el problema del *blur*.

Redes neuronales convolucionales

Las Redes Neuronales Convolucionales (CNN) [34] son un tipo de red neuronal artificial con capas ocultas que trabajan con volúmenes en lugar de con vectores. Se inspiran en la organización del córtex visual humano, y han demostrado ser altamente eficaces en tareas de visión por computadora, como clasificación de imágenes, detección de objetos y segmentación semántica. Su capacidad para aprender automáticamente características jerárquicas de los datos las hace ideales para problemas donde la extracción manual de características sería compleja o ineficiente.
El entrenamiento de una CNN sigue el mismo principio que otras redes neuronales profundas. Se inicia con una etapa de preprocesamiento, donde los valores de los píxeles de la imagen se suelen normalizar dividiendo por 255 para obtener valores entre 0 y 1. Luego, los datos se propagan a través de las capas de la red, donde cada capa aprende características de distinta complejidad. Para optimizar el modelo, se define una función de pérdida que mide la discrepancia entre las predicciones del modelo y los valores reales. Un optimizador ajusta los pesos de la red iterativamente minimizando la función de pérdida mediante retropropagación.

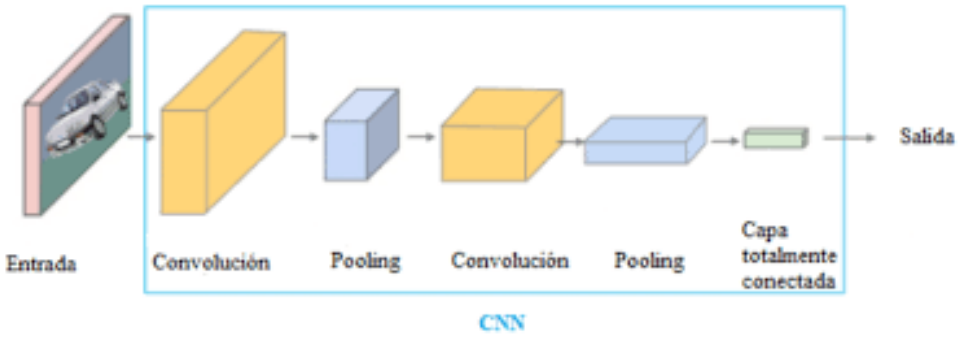


Figura 3.6: Red Neuronal Convolucional (imagen de [46])

Las CNN (ver Figura 3.6) están compuestas por varias capas que trabajan en conjunto para extraer y procesar características de los datos de entrada [7]:

- **Capa de Convolución.** Es la base de las CNN y su función principal es extraer características relevantes de la imagen mediante la aplicación de filtros o *kernels* (ver Figura 3.7). Un filtro es una pequeña matriz que se desliza sobre la imagen de entrada y realiza una operación de convolución, generando una matriz llamada mapa de características, que resalta patrones importantes como bordes, texturas o estructuras más complejas en capas más profundas. El número de filtros aplicados se denomina *depth*, el número de *pixels* saltados al desplazar el filtro se denomina *stride*. Además, se puede añadir un marco de ceros alrededor de la imagen para facilitar la aplicación de los filtros sobre los elementos de los bordes, denominado *padding*. Después de aplicar la convolución, suele utilizarse una función de activación, generalmente la ReLU (*Rectified Linear Unit*), definida como $f(x) = \text{máx}(0, x)$. Esta transformación introduce no linealidad en la red, permitiendo que las CNN modelen relaciones más complejas en los datos.

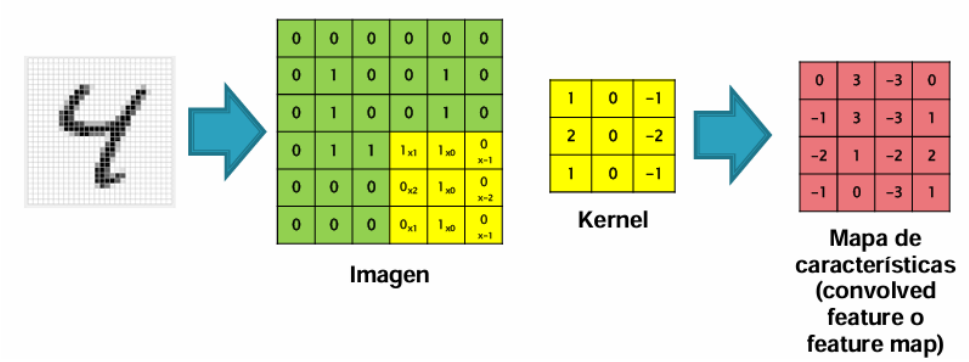


Figura 3.7: Capa de convolución (imagen de [7])

- **Capa de *Pooling*.** Se emplea para reducir la dimensionalidad de los mapas de características y preservar la información más importante (ver Figura 3.8). Los tipos más comunes de pooling son el *max pooling*, que selecciona el valor máximo dentro de una ventana de tamaño predefinido, el *sum pooling*, que selecciona la suma de los elementos dentro de la ventana, y el *average pooling*, que calcula el valor promedio de los elementos dentro de la ventana. Esta reducción de dimensiones mejora la eficiencia computacional del modelo y ayuda a minimizar el riesgo de sobreajuste.

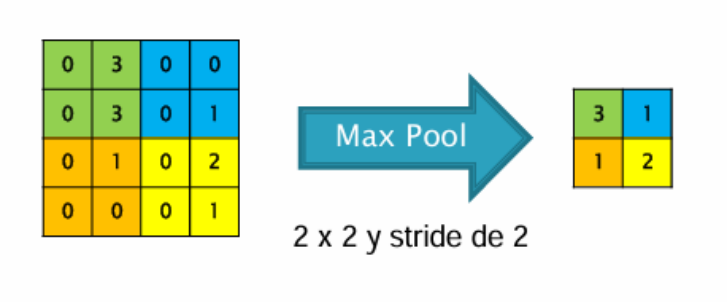


Figura 3.8: Capa de pooling (imagen de [7])

- **Capa *Fully Connected*.** Aparece en las etapas finales de la red, y tiene el objetivo de integrar las características extraídas por las capas anteriores transformando la matriz tridimensional generada en las capas anteriores en un vector unidimensional.
- **Capa *Softmax*.** Para problemas de clasificación, la última capa de la red es generalmente una capa *Softmax*, que convierte el vector de salida en una distribución de probabilidad sobre las clases posibles. Esto permite que la red asigne una probabilidad de pertenencia a cada categoría en función de los datos de entrada.

Redes Neuronales Residuales

Las *Residual Networks* (ResNet) (ver Figura 3.9) son un tipo de arquitectura de redes neuronales profundas introducidas por He et al. en 2015 [17]. Se caracterizan por su capacidad para entrenar redes extremadamente profundas sin sufrir el problema del *vanishing gradient*. El problema del *vanishing gradient* se refiere a la dificultad que ocurre durante el entrenamiento de redes neuronales profundas, donde los gradientes calculados durante la retropropagación se vuelven progresivamente más pequeños a medida que se retrocede hacia las capas iniciales. Como resultado, los pesos de estas capas apenas se actualizan, lo que impide que la red aprenda correctamente. ResNet es una arquitectura clave en tareas de visión computacional como clasificación de imágenes, segmentación y restauración de imágenes.

El concepto central de ResNet es el uso de conexiones residuales o *skip connections*, que permiten el paso directo de la información entre capas, evitando la degradación de los gradientes. Matemáticamente, en lugar de aprender una transformación directa $H(x)$ de la entrada x, una *ResNet* aprende una función residual $F(x)$ tal que

$$H(x) = F(x) + x. \tag{3.1}$$

Esta formulación permite que las redes puedan ser entrenadas con muchas más capas sin que la información se pierda o degrade en el proceso de propagación del gradiente.

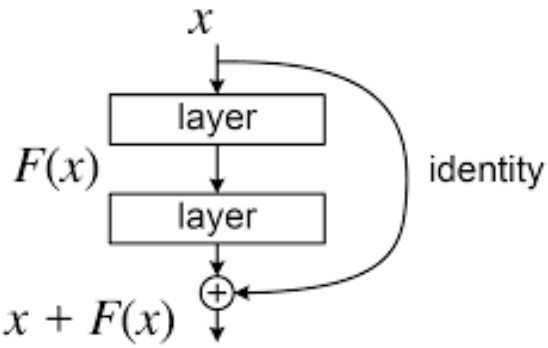


Figura 3.9: Esquema general de *ResNet* (imagen de [60])

Los bloques residuales son la base de la arquitectura de *ResNet*. Cada bloque está compuesto por capas de convolución con activaciones *ReLU*, acompañadas de una conexión de atajo (*shortcut connection*) que permite que la entrada original sea sumada a la salida de la capa transformada. Estos bloques pueden dividirse en:

- **Bloques básicos**: Utilizados en arquitecturas menos profundas, consisten en capas de convolución de 3×3 seguidas de una activación *ReLU*.
- **Bloques *Bottleneck***: Diseñados para redes más profundas, utilizan capas de convolución de 1×1 antes y después de una capa de 3×3, lo que reduce la dimensionalidad computacional sin perder información relevante.

Redes Neuronales *Encoder-Decoder*

Una *Encoder-Decoder Network* [8] es una arquitectura diseñada para transformar una secuencia de entrada en una secuencia de salida. El *encoder* procesa la entrada y la convierte en una representación interna (vector de contexto), que luego es utilizada por el *decoder* para generar la salida (ver Figura 3.10). Es una arquitectura fundamental en el aprendizaje profundo utilizada en una amplia variedad de tareas que requieren la transformación de una entrada en una representación compacta antes de reconstruir una salida. El modelo *Encoder-Decoder* se compone de dos elementos principales:

- **Encoder**: Captura la información de entrada y la comprime en una representación de menor dimensión. Normalmente, consiste en una serie de capas convolucionales o recurrentes que extraen características esenciales del dato de entrada.
- **Decoder**: Toma la representación comprimida y la expande nuevamente para generar una salida con la misma estructura que la entrada original. Este proceso se realiza mediante operaciones de *upsampling*, transconvoluciones o mecanismos de atención.

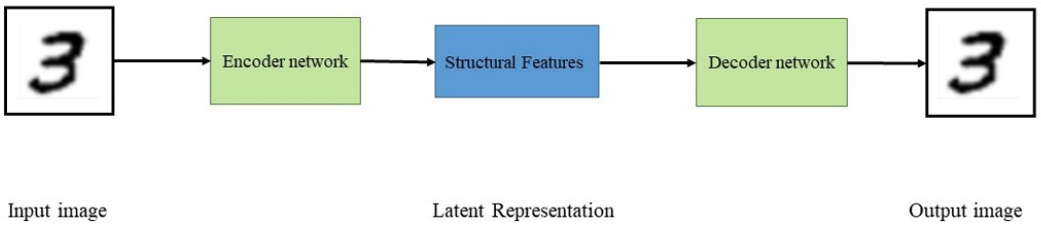


Figura 3.10: Red Neuronal *Encoder-Decoder* (imagen de [4])

Matemáticamente, si definimos X como la entrada e Y como la salida, la transformación en un modelo *Encoder-Decoder* se puede expresar como

$$Z = f_{enc}(X), \quad Y = f_{dec}(Z), \tag{3.2}$$

donde f_{enc} y f_{dec} representan las funciones de codificación y decodificación, respectivamente, y Z es la representación latente.

Redes neuronales recurrentes

Las Redes Neuronales Recurrentes (RNN) [7] son un tipo de red neuronal especializada en el procesamiento de datos secuenciales, como texto, audio y vídeo (ver Figura 3.11). A diferencia de las redes neuronales tradicionales, que procesan cada entrada de forma independiente, las RNN poseen la capacidad de retener información de estados previos para influir en la salida actual. Esto las hace ideales para tareas donde el contexto y la dependencia temporal son fundamentales, como la traducción automática, el reconocimiento de voz y la generación de texto.
A diferencia de una red neuronal convencional, donde cada capa procesa los datos de manera independiente, una RNN introduce el concepto de recurrencia. Esto significa que cada neurona no solo recibe la entrada actual, sino también información de estados anteriores. Las RNN se

entrenan mediante un algoritmo llamado *Backpropagation Through Time* (BPTT), una extensión del algoritmo de retropropagación convencional. En este enfoque, la red se desenrolla en el tiempo, lo que implica representar cada paso de la secuencia como si fuera una capa independiente, permitiendo aplicar la retropropagación sobre toda la secuencia. La función de pérdida se calcula considerando todas las salidas de la secuencia y los pesos se ajustan utilizando optimizadores.
A pesar de sus ventajas, las RNN tradicionales enfrentan problemas como recordar relaciones temporales lejanas, el *vanishing gradient*. Para mitigar este problema, se han desarrollado variantes como las LSTM (*Long Short-Term Memory*), que incorporan mecanismos de control para gestionar la memoria de manera más eficiente.

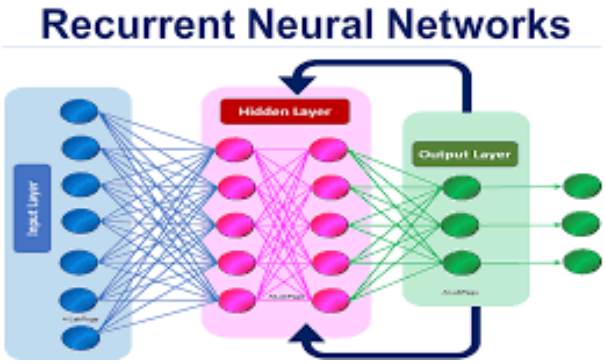


Figura 3.11: Red Neuronal Recurrente (imagen de [27])

Redes generativas adversarias

Las Redes Generativas Adversarias (GAN) [15] son un tipo de arquitecturas de aprendizaje profundo (ver Figura 3.12). Su principal característica es que se componen de dos redes neuronales que compiten entre sí en el proceso de entrenamiento. Estas redes han revolucionado el campo del aprendizaje generativo, permitiendo la creación de datos sintéticos altamente realistas en diversos dominios, como la generación de imágenes, la síntesis de voz y la creación de texto artificial. Una GAN está compuesta por dos modelos neuronales:

- **Generador:** Su función es crear datos sintéticos a partir de una distribución aleatoria. Aprende a generar muestras que se asemejan a los datos reales para engañar al discriminador.
- **Discriminador:** Actúa como un clasificador binario que distingue entre muestras reales (provenientes del conjunto de datos de entrenamiento) y falsas (creadas por el generador).

Ambos modelos se entrenan de manera simultánea y competitiva. Mientras el generador mejora su capacidad para crear datos realistas, el discriminador se vuelve más preciso al diferenciar entre datos reales y sintéticos. Este proceso se basa en la teoría de juegos, donde el objetivo es alcanzar un equilibrio de Nash en el que ambos modelos mejoren mutuamente su rendimiento.
El entrenamiento de una GAN sigue un enfoque basado en la minimización de una función de pérdida basada en la distancia entre la distribución de los datos reales y la distribución de los datos generados. Se optimizan ambas redes mediante un procedimiento iterativo que sigue estos pasos:

1. Se genera una muestra aleatoria a partir de una distribución latente y se pasa al generador.
2. El generador transforma la muestra en datos sintéticos y los envía al discriminador.
3. El discriminador evalúa los datos sintéticos junto con muestras reales del conjunto de entrenamiento, asignándoles una probabilidad de autenticidad.
4. Se calcula la función de pérdida y se actualizan los parámetros del generador y el discriminador mediante descenso del gradiente.
5. Este proceso se repite hasta que el generador es capaz de producir muestras indistinguibles de las reales para el discriminador.

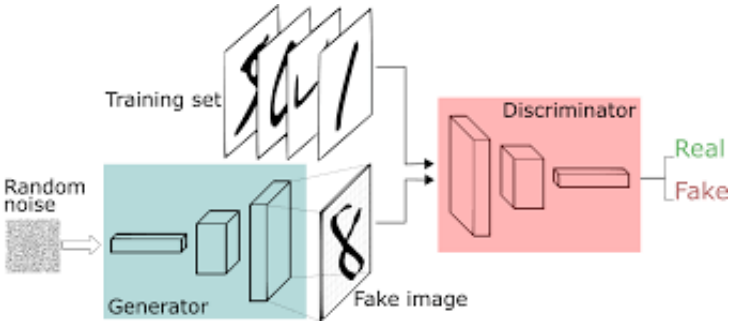


Figura 3.12: Generative Adversarial Networks (imagen de [36])

3.2.5. Blur

El *blur* o desenfoque (ver Figura 3.13) es una degradación en la calidad de una imagen causada por diversos factores, como el movimiento de la cámara, la falta de enfoque o el movimiento de los objetos dentro de la escena. Esta degradación afecta la nitidez de la imagen, dificultando la identificación de detalles y estructuras en la misma [4].

Figura 3.13: Imágenes con y sin *blur* (imagen de [4])

En la Subsección 3.2.4 se explicaron los conceptos de convolución y *kernel*. Con ello, el proceso de *blur* puede modelarse matemáticamente como la convolución de una imagen nítida con un kernel de desenfoque, además de la posible adición de ruido. En términos generales, se puede expresar como

$$I_b = I_c * K + n, \tag{3.3}$$

donde $*$ es la operación de convolución y:

- I_b es la imagen borrosa resultante,
- I_c es la imagen nítida original,
- K es el kernel de desenfoque,
- n es el ruido añadido en la captura de la imagen.

Dependiendo de si el kernel de desenfoque es conocido o no, se puede clasificar el problema en **blur no ciego** (*non-blind blur*) y **blur ciego** (*blind blur*). En el primer caso, se tiene información parcial o total del kernel utilizado, mientras que en el segundo, tanto la imagen original como el kernel de *blur* deben ser estimados.

Tipos de Blur

Existen diferentes tipos de desenfoque en imágenes, cada uno con características y efectos específicos. A continuación, se presentan los más comunes [64]:

- ***Motion Blur* (Desenfoque por Movimiento)**: Ocurre cuando la cámara o el objeto en la escena están en movimiento durante la captura de la imagen. Se modela como una convolución con un kernel que representa la trayectoria del movimiento, lo que genera estelas o rastros en la imagen.
- ***Out-of-Focus Blur* (Desenfoque por Falta de Enfoque)**: Se produce cuando la lente de la cámara no está correctamente enfocada en la escena. Se modela mediante una función de dispersión del punto (PSF, *Point Spread Function*) y puede corregirse mediante técnicas de restauración de imagen.
- ***Gaussian Blur* (Desenfoque Gaussiano)**: Se genera al aplicar un filtro gaussiano a la imagen, lo que suaviza los detalles y elimina ruido de alta frecuencia. Es común en preprocesamiento de imágenes y en simulaciones de desenfoque. Matemáticamente, se define como

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{3.4}$$

donde x y y representan la distancia desde el origen en los ejes horizontal y vertical, y σ es la desviación estándar del filtro.

- ***Mixed Blur* (Desenfoque Mixto)**: Se presenta cuando en una imagen confluyen múltiples fuentes de desenfoque, como el *motion blur* y el *out-of-focus blur*. Este tipo de desenfoque es más complejo de modelar y corregir debido a la superposición de efectos degradantes.
- ***Channel-Dependent Blur* (Desenfoque Dependiente del Canal)**: Se produce debido a aberraciones cromáticas en la óptica de la cámara, provocando que diferentes longitudes de onda (colores) sufran distintos niveles de desenfoque, generando bordes de colores en la imagen.

3.2.6. Deblurring

Una vez introducido qué es el *blur*, pasemos a estudiar el *deblurring*. El *deblurring* es una técnica utilizada en visión computacional para recuperar una imagen nítida a partir de una imagen degradada por desenfoque. Como vimos en el entorno de negocio, este proceso es de gran relevancia en diversas aplicaciones, desde la mejora de fotografías hasta la reconstrucción de imágenes médicas y el análisis de vídeo en vigilancia. Análogamente al caso del *blur*, existen dos tipos principales de *deblurring*. El primero, **deblurring no ciego** (*non-blind deblurring)*, asume que el kernel de desenfoque es conocido o puede ser estimado con precisión. El segundo, **deblurring ciego** (*blind deblurring)*. En este caso, tanto la imagen nítida original como el kernel de desenfoque son desconocidos y deben estimarse simultáneamente.

Mecanismos de *Deblurring* con Redes Neuronales

En el campo del *deblurring* basado en técnicas de *Deep Learning* se han desarrollado diferentes mecanismos para mejorar la reconstrucción de imágenes nítidas [4]:

- **Skip Connections.** Como vimos en las Redes Neuronales Residuales, las *skip connections* son un mecanismo utilizado en redes neuronales profundas que permite la propagación directa de información entre capas distantes dentro de una red (ver Figura 3.14). Estas conexiones son esenciales para mejorar la estabilidad del entrenamiento y la propagación del gradiente en modelos profundos, evitando el problema del *vanishing gradient*. En el contexto del *deblurring*, las *skip connections* han demostrado ser altamente efectivas en la captura de características borrosas en diferentes niveles de la red.

  Matemáticamente, una *skip connection* en una red convolucional se representa como

  $$H(x) = F(x) + x, \tag{3.5}$$

  donde:

  - x es la entrada de la capa,
  - $F(x)$ es la transformación no lineal aprendida por la red.
    En el caso del *non-blind deblurring*, esta transformación se encarga de refinar la imagen previamente deconvolucionada utilizando el conocimiento explícito del *blur kernel*, recuperando detalles. En cambio, en el *blind deblurring*, $F(x)$ representa un mapeo más complejo que intenta aprender simultáneamente la eliminación del desenfoque y la reconstrucción de la imagen nítida sin información previa del kernel,
  - $H(x)$ es la salida final de la capa con la *skip connection*.

Las principales ventajas del uso de *skip connections* en redes de *deblurring* incluyen preservación de detalles finos en la imagen restaurada, aceleración de la convergencia del modelo durante el entrenamiento, mejora en la propagación del gradiente y estructura más robusta para manejar diferentes niveles de desenfoque en una imagen. Dado su impacto en la restauración de imágenes, las *skip connections* se consideran un componente clave en arquitecturas avanzadas de *deblurring*, diferenciándose de las conexiones residuales utilizadas en modelos como *ResNet*.

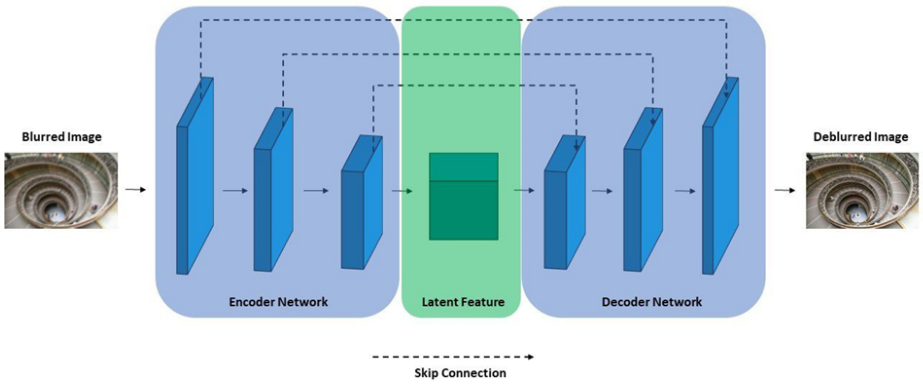


Figura 3.14: *Skip Connection* (imagen de [4])

- **Multi-Scale Processing.** El *Multi-Scale Scheme* [4] es una técnica utilizada en modelos de *deblurring* basada en la restauración progresiva de una imagen en diferentes escalas, siguiendo una estructura piramidal (ver Figura 3.15). Este enfoque permite que el modelo primero recupere información en una escala más baja, donde los detalles más gruesos son más fáciles de estimar, y posteriormente refine la reconstrucción en escalas más altas para mejorar la calidad de la imagen final. En el contexto del *deblurring ciego*, esta técnica se utiliza para estimar el *blur kernel* y la imagen restaurada a la escala más pequeña, combinando los resultados con versiones más finas de la imagen en niveles superiores para mejorar la recuperación de los detalles. Desde un punto de vista computacional, el *Multi-Scale Scheme* mejora la eficiencia del aprendizaje y la estabilidad de la red, permitiendo que las redes neuronales profundas manejen estructuras de desenfoque complejas de manera más efectiva. Su integración en modelos de *deblurring* ha demostrado ser exitosa en la mejora de la calidad de las imágenes restauradas, tanto en métodos basados en optimización como en enfoques de aprendizaje profundo.

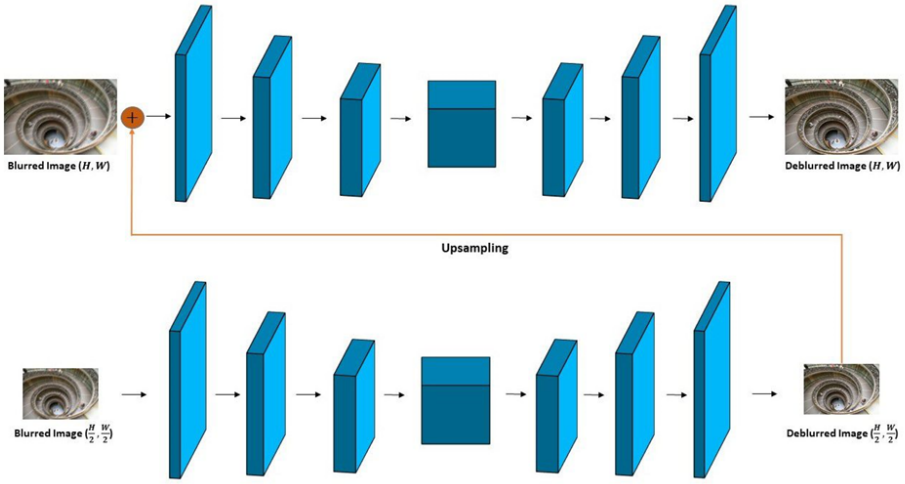


Figura 3.15: *Multi-Scale Scheme* (imagen de [4])

- **Mecanismo de Atención (*Attention Mechanism*)** El mecanismo de atención en redes neuronales profundas está inspirado en el sistema de percepción visual humano, que se enfoca en las características más relevantes de una escena en lugar de procesarla en su totalidad. Este concepto ha sido recientemente integrado en arquitecturas *CNN* para mejorar el rendimiento de las redes en tareas de *deblurring*.

  El objetivo del mecanismo de atención es asignar pesos a diferentes partes de la imagen, permitiendo que la red aprenda qué regiones son más importantes para la reconstrucción. Existen dos tipos principales de módulos de atención utilizados en el *deblurring*:

  - **Módulo de Atención Espacial (*Spatial Attention Module*)** Este módulo extrae la relación espacial entre las características de la imagen, especificando la ubicación de la información útil en todos los canales (ver Figura 3.16). Se basa en operaciones de submuestreo aplicadas a cada posición espacial para generar un mapa de atención bidimensional, al que posteriormente se le aplica una convolución para refinar la información.

    Matemáticamente, se define como

    $$M_s = \sigma(f_{conv}([AvgPool(F), MaxPool(F)])), \tag{3.6}$$

    donde:

    - M_s es el mapa de atención espacial.
    - σ representa la función sigmoide,

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

    - f_{conv} es una capa convolucional aplicada a la concatenación de los resultados de *average pooling* y *max pooling*.
    - F es el conjunto de características de la imagen.

Este enfoque ha sido utilizado en modelos como CBAM (*Convolutional Block Attention Module*) [61], lo que ha permitido mejorar la propagación de información y la recuperación de características borrosas en la imagen restaurada.

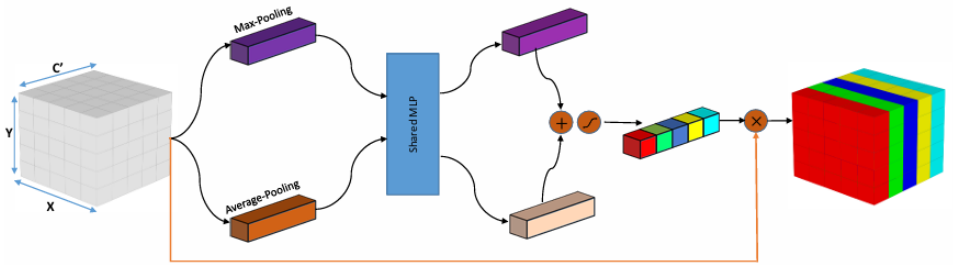


Figura 3.16: *Módulo de Atención Espacial* (imagen de [4])

- **Módulo de Atención por Canal (*Channel Attention Module*)** A diferencia del módulo de atención espacial, este mecanismo se enfoca en la información global de cada canal a través de todas las ubicaciones espaciales (ver Figura 3.17). En lugar de considerar relaciones espaciales, asigna pesos a cada canal para recalibrar la importancia de las características aprendidas.

  El módulo de atención por canal se calcula aplicando operaciones de *average pooling* y *max pooling* dentro de cada canal, seguido por un perceptrón multicapa (MLP) que genera un mapa de atención con valores en el rango $[0, 1]$.

  La ecuación del mapa de atención por canal es

$$M_c = \sigma(MLP(AvgPool(F)) + MLP(MaxPool(F))), \tag{3.7}$$

  donde:

  - M_c es el mapa de atención por canal.
  - σ representa la función sigmoide.
  - MLP es un perceptrón multicapa que aprende la importancia de cada canal.

  Este mecanismo permite a la red centrarse en características relevantes para la restauración de la imagen, mejorando la reconstrucción de texturas y detalles finos.

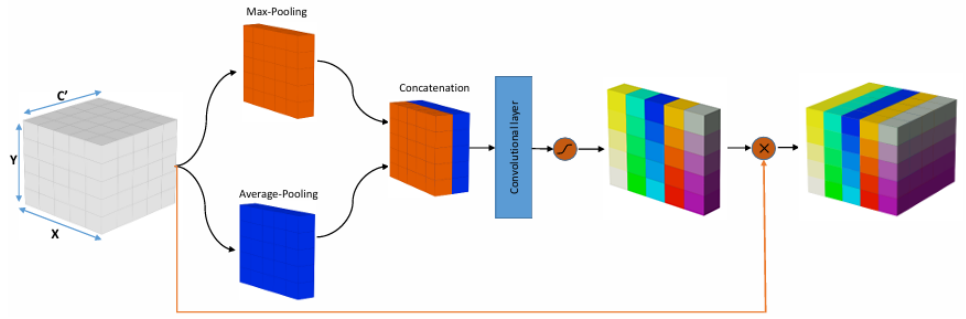


Figura 3.17: *Módulo de Atención por Canal* (imagen de [4])

Aunque muchos de estos mecanismos se han popularizado en arquitecturas de deblurring ciego (*blind deblurring*) debido a su capacidad de aprender directamente la imagen nítida a partir de una entrada borrosa sin necesidad de conocer el *blur kernel*, también pueden encontrarse en enfoques de deblurring no ciego (*non-blind deblurring*), donde se utilizan para refinar la imagen restaurada. Por ejemplo, técnicas como las *skip connections* y los módulos de atención se aplican tanto en modelos ciegos como no ciegos, mientras que el esquema de procesamiento multiescala ha sido particularmente efectivo en contextos ciegos, donde la información del kernel es desconocida [32], [51], [61].

Funciones de Pérdida en *Deblurring* con Redes Neuronales

Existen diferentes tipos de funciones de pérdida que permiten optimizar la calidad de la imagen restaurada al comparar la salida generada con la imagen original. En este apartado, se presentan algunas de las funciones de pérdida más utilizadas en redes neuronales para *deblurring* [4].

- ***Content Loss (Reconstrucción de Contenido)***. Mide la discrepancia entre la imagen restaurada y la imagen original en función de la norma L1 (*Mean Absolute Error*) [9] o la norma L2 (*Mean Square Error*) [10]. Se define como

$$L_{Cont} = \frac{1}{N} \sum_{k=1}^{K} \|I'_k - I_k\|_{norm}, \tag{3.8}$$

  donde:

  - I_k: Imagen de referencia o imagen original.
  - I'_k: Imagen restaurada generada por el modelo.
  - N: Número total de píxeles.
  - K: Número total de escalas en una estructura multi-escala.
  - $\|\cdot\|_{norm}$: Puede ser la norma L1 (*Mean Absolute Error (MAE)*) [9] o L2 (*Mean Square Error (MSE)*) [10], dependiendo del criterio de optimización.

- ***Perceptual Loss***. Compara las imágenes en función de sus representaciones en una red convolucional preentrenada, como VGG19, en lugar de realizar comparaciones píxel a píxel. Se expresa como

$$L_{Perc} = \frac{1}{C_j H_j W_j} \|\phi_j(I') - \phi_j(I)\|_2, \tag{3.9}$$

  donde:

  - $\phi_j(I)$: Mapa de características extraído de la imagen original I en la capa j de VGG19.
  - $\phi_j(I')$: Mapa de características extraído de la imagen restaurada I' en la misma capa j.
  - C_j, H_j, W_j: Número de canales, altura y ancho del mapa de características en la capa j.
  - $\|\cdot\|_2$: Norma L2 entre los mapas de características.

- ***Regularized Content Loss***. Agrega un término de regularización a la función de pérdida para imponer restricciones de dispersión en los datos de la imagen. Se define como

$$L_{RC} = \sum_{k=1}^{K} \|I'_k - I_k\|_{norm} + \lambda P(I'_k), \tag{3.10}$$

  donde:

  - λ: Peso de regularización que controla la contribución del término de penalización.
  - $P(I'_k)$: Información previa de la imagen generada (puede incluir gradientes de imagen, canales oscuros o canales brillantes).

- ***Adversarial Loss***. Utilizada en arquitecturas GAN para hacer que la imagen generada sea lo más parecida posible a la imagen real. Se expresa como

$$L_{Adv} = E_{I\sim p_{target}(I)}[\log(D(I))] + E_{B\sim p_{blurred}(B)}[\log(1 - D(G(B)))], \tag{3.11}$$

  donde:

  - $D(I)$: Discriminador que predice si la imagen I es real o generada.
  - $G(B)$: Generador que produce imágenes nítidas a partir de imágenes borrosas B.
  - $E_{X\sim Y}[f(X)]$: Esperanza matemática de una función $f(X)$ cuando la variable aleatoria X sigue la distribución Y.
  - $p_{target}(I)$ y $p_{blurred}(B)$: Distribuciones de la imagen real y la imagen borrosa, respectivamente.

- ***Gradient Loss***. Ayuda a preservar los bordes de la imagen restaurada al minimizar la diferencia entre los gradientes de la imagen original y la generada. Se define como

$$L_{Grad} = \|(\nabla I' - \nabla I)_x\|_{norm} + \|(\nabla I' - \nabla I)_y\|_{norm}, \tag{3.12}$$

  donde:

  - $\nabla I'$ y ∇I: Gradientes de la imagen generada y la imagen real.
  - x y y: Direcciones horizontal y vertical de los gradientes.

- ***Frequency Content Loss***. Evalúa la diferencia entre la imagen original y la restaurada en el dominio de la frecuencia, utilizando transformada rápida de Fourier (FFT). Su ecuación es

$$L_{FR} = \sum_{k=1}^{K} \|F(I'_k) - F(I_k)\|_{norm}, \tag{3.13}$$

  donde:

  - $F(I_k)$ y $F(I'_k)$: Transformada de Fourier de la imagen real y restaurada.

Estas funciones de pérdida se combinan frecuentemente en modelos de *deblurring* para optimizar diferentes aspectos de la restauración de imágenes, como la nitidez, la preservación de texturas y la coherencia con la imagen original.

Métricas de rendimiento

Para evaluar la calidad de las imágenes restauradas en tareas de *deblurring*, se utilizan diversas métricas de rendimiento que permiten realizar comparaciones cuantitativas. A continuación, se presentan las métricas más utilizadas.

- ***Mean Squared Error (MSE)***. Mide la diferencia entre la imagen original y la imagen restaurada a nivel de píxel. Se define como

$$MSE(I, \hat{I}) = \frac{1}{N} \sum_{i=1}^{N} (I_i - \hat{I}_i)^2, \tag{3.14}$$

  donde:

  - I: Imagen original.
  - $\hat{I}$: Imagen restaurada.
  - N: Número total de píxeles en la imagen.

  Un menor valor de MSE indica una mejor recuperación de la imagen.

Dos métricas muy utilizadas en el contexto de las tareas de *deblurring* son *Peak Signal-to-Noise Ratio* (PSNR), y *Structural Similarity Index Measure* (SSIM), que permiten cuantificar la efectividad de los métodos de *deblurring* en términos de calidad de imagen y fidelidad a la imagen original. En la Sección 4.2 se definirán, justificando su uso en este proyecto.

3.3. Estado del arte

Esta sección permite comprender cómo se ha abordado previamente el problema planteado en este proyecto. A través de la identificación y evaluación de soluciones existentes, determinamos sus principales ventajas y limitaciones, lo que facilita la detección de oportunidades de mejora.

3.3.1. Descripción de trabajos relacionados

Primero se hará una breve descripción de los trabajos seleccionados. Los dos primeros se enfocan en el *deblurring*, mientras que los tres siguientes son aplicaciones industriales de dichas técnicas:

Amrollahi Biyouki, Sajjad. & Hwangbo, Hoon. (A Comprehensive Survey on Deep Neural Image Deblurring) [4]

Este artículo presenta un análisis detallado sobre el estado actual del *deblurring* de imágenes basado en redes neuronales profundas. A lo largo del documento, los autores exploran cómo los métodos tradicionales basados en optimización han sido reemplazados o combinados con enfoques de *Deep Learning*, proporcionando una revisión extensa de las técnicas más utilizadas en este campo. En primer lugar, el artículo explica la diferencia entre los métodos de *deblurring* ciegos (*blind deblurring*) y no ciegos (*non-blind deblurring*), destacando cómo los modelos basados en redes neuronales han mejorado la restauración de imágenes. También se abordan los diferentes

tipos de redes utilizadas en el proceso de eliminación del desenfoque, desde redes neuronales convolucionales (*CNN*) hasta modelos más avanzados como redes neuronales adversarias(*GANs*) y redes multi-escala.

Uno de los puntos más interesantes del artículo es su clasificación de los métodos existentes, donde los autores no solo mencionan las arquitecturas más populares, sino que también comparan sus fortalezas y debilidades en términos de precisión y eficiencia computacional. Se presentan métricas como *PSNR (Peak Signal-to-Noise Ratio)* y *SSIM (Structural Similarity Index)* para evaluar el desempeño de los modelos y se discuten los principales conjuntos de datos utilizados en la investigación. Además, el artículo no se limita a revisar trabajos anteriores, sino que también identifica los desafíos actuales del *deblurring* con *Deep Learning*. Entre estos desafíos, se menciona la falta de datos de entrenamiento de alta calidad, la dificultad de generalización de los modelos a distintos tipos de desenfoque y la necesidad de mejorar la eficiencia computacional para que estos algoritmos puedan implementarse en dispositivos con recursos limitados.

Zhang, Kaihao. et al. (Deep Image Deblurring: A Survey) [64]

Este artículo ofrece una revisión exhaustiva sobre los avances recientes en el campo de la eliminación del desenfoque de imágenes mediante técnicas de *Deep Learning*. En esta revisión, los autores clasifican los métodos existentes, analizan métricas de evaluación, presentan los conjuntos de datos más utilizados y discuten los principales desafíos en este ámbito.

El documento inicia con una descripción de las causas del desenfoque, diferenciando entre el desenfoque por movimiento de la cámara, el desenfoque por objetos en movimiento y el desenfoque por falta de enfoque. Después, los autores presentan un análisis de los métodos de eliminación de desenfoque basados en redes neuronales convolucionales (*CNN*), distinguiendo entre métodos de deblurring ciego y no ciego, y explorando arquitecturas como autoencoders profundos (DAE), redes generativas adversarias (GAN), redes en cascada, redes multi-escala y redes de *reblurring*. En cuanto a las arquitecturas más utilizadas, el estudio destaca la eficacia de las redes en cascada y multi-escala en la eliminación del desenfoque progresivo, mientras que las *GANs* han demostrado ser útiles para generar imágenes más realistas, aunque a menudo con un menor rendimiento en métricas tradicionales como *PSNR* y *SSIM*. Asimismo, se analiza el uso de mecanismos de atención para mejorar la localización de áreas desenfocadas y refinar la restauración de detalles.

Otro aspecto importante tratado en el artículo es la evaluación del rendimiento de los modelos, incluyendo métricas como *LPIPS* y *FID*, que buscan mejorar la correlación con la percepción humana. Además, se presenta una comparación de los conjuntos de datos más utilizados en la comunidad de deblurring, como *GoPro*, *HIDE* y *RealBlur*, que han sido clave en el entrenamiento y validación de estos modelos. Finalmente, los autores discuten los principales desafíos en este campo, como la necesidad de mejorar la generalización de los modelos a escenarios con desenfoques complejos y la optimización del desempeño computacional para permitir su implementación en dispositivos con recursos limitados. También se plantea la importancia de explorar estrategias como el aprovechamiento de información de profundidad para mejorar la restauración de imágenes en escenarios tridimensionales.

Wu, Dong., Guerrero, Patricio., Sinico, Mirko. & Dewulf, Wim. (L0 Regularized Image Deblurring applied to Porosity Analysis in Industrial X-ray Computed Tomography) [63]

Este artículo aborda la aplicación de técnicas de *deblurring* en el contexto de la tomografía computarizada industrial por rayos X (*ixCT*), con el objetivo de mejorar la precisión en el análisis de porosidad en componentes manufacturados (ver Figura 3.18). Los autores destacan cómo el desenfoque, generado por factores como el tamaño del punto focal y la geometría de proyección, afecta la resolución espacial y la calidad de las imágenes reconstruidas en *ixCT*.

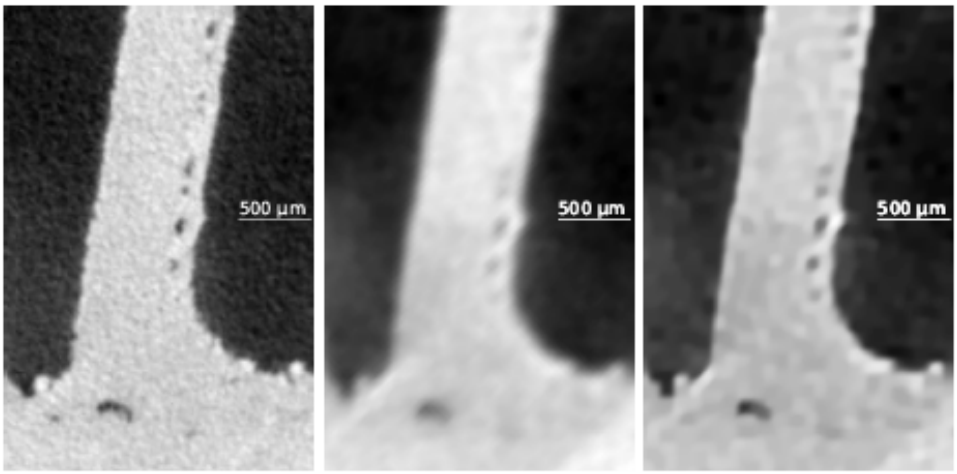


Figura 3.18: Aplicación industrial del deblurring en análisis de porosidad (imagen de [63])

El estudio presenta un enfoque basado en **regularización L0** para la eliminación del desenfoque en volúmenes reconstruidos de *ixCT*. A diferencia de métodos convencionales como la deconvolución de Wiener o la variación total, la regularización L0 aprovecha la escasez de información en intensidad y gradiente de las imágenes para mejorar la calidad visual sin introducir artefactos significativos. Uno de los aspectos más relevantes del artículo es la evaluación cuantitativa del impacto del *deblurring* en la detección de defectos internos en componentes metálicos fabricados aditivamente. Para ello, los autores comparan imágenes obtenidas con dos escáneres de diferente resolución y aplican su método de *deblurring* sobre datos de baja calidad. Los resultados muestran una mejora del 27 % en la percepción de nitidez y un 40 % en la detección de porosidades, lo que subraya la importancia del procesamiento de imágenes en aplicaciones industriales. Además, el artículo discute las implicaciones computacionales de este método, señalando que, aunque el tiempo de procesamiento es superior a los métodos tradicionales, la mejora en la precisión del análisis justifica su implementación en escenarios donde la calidad de imagen es crítica.

En conclusión, este trabajo demuestra la eficacia del *deblurring* basado en regularización L0 para la mejora de imágenes en *ixCT*, con aplicaciones directas en control de calidad y análisis estructural de materiales industriales. También abre la puerta a futuras investigaciones que combinen este enfoque con técnicas basadas en *Deep Learning* para optimizar la eficiencia y robustez del proceso.

Kim, Kyuseok., Cho, Hyosung., Je, Uikyu., Park, Chulkyu., Lim, Hyunwoo., Kim, Guna., et al. (Improvement of image characteristics in high-voltage computed tomography (CT) by applying a compressed-sensing (CS)-based image deblurring scheme) [23]

Este artículo explora la aplicación de un esquema de *deblurring* basado en **compressed sensing (CS)** para mejorar la calidad de imágenes en tomografía computarizada de alto voltaje (ver Figura 3.19). Los autores presentan un enfoque innovador para reducir el desenfoque inherente a las imágenes reconstruidas en *CT* debido al tamaño finito del punto focal, la resolución del detector y los procedimientos de reconstrucción.

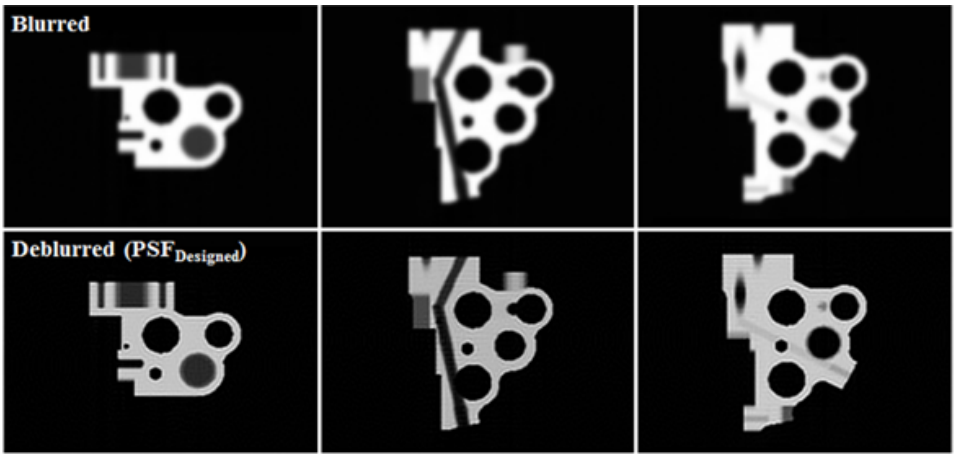


Figura 3.19: Aplicación industrial del deblurring en tomografía de alto voltaje (imagen de [23])

Uno de los aspectos más destacados del artículo es la validación experimental del método en imágenes de motores eléctricos y otros objetos industriales inspeccionados mediante *CT* de alta energía. Los resultados muestran que el esquema de *deblurring* basado en *CS* mejora significativamente la visibilidad de estructuras internas, reduciendo la pérdida de detalles en comparación con imágenes borrosas obtenidas mediante reconstrucción estándar de retroproyección filtrada (*Filtered Back Projection, FBP*). Además, el análisis cuantitativo con medidas de la función de transferencia de modulación (*MTF*) y el espectro de potencia de ruido (*NPS*) confirma la mejora en la resolución espacial y la reducción de ruido tras la aplicación del algoritmo de deconvolución.

El artículo concluye que la técnica basada en *compressed sensing* es una solución efectiva para el problema del desenfoque en tomografía computarizada industrial, con aplicaciones directas en manufactura y análisis de defectos estructurales. Sin embargo, los autores también señalan la necesidad de optimizar el tiempo de cómputo y explorar combinaciones con enfoques de *Deep Learning* para mejorar la eficiencia en entornos de producción en tiempo real.

Wang, Jiaxiang., Wan, Meng., Zhang, Pufen., Chang, Sijie., Du, Hao., Shi, Peng., Yu, Hongying., Sun, Dongbai., Wang, Jue. & Wang, Yangang. (MCIDN: Deblurring Network for Metal Corrosion Images) [55]

Este artículo aborda el problema del desenfoque en imágenes de corrosión de metales, un desafío crítico en el ámbito de la ciencia de materiales y la industria (ver Figura 3.20). Los autores presentan la red **MCIDN (Metal Corrosion Images Deblurring Network)**, una arquitectura basada en *Deep Learning* diseñada específicamente para restaurar imágenes degradadas de corrosión metálica.

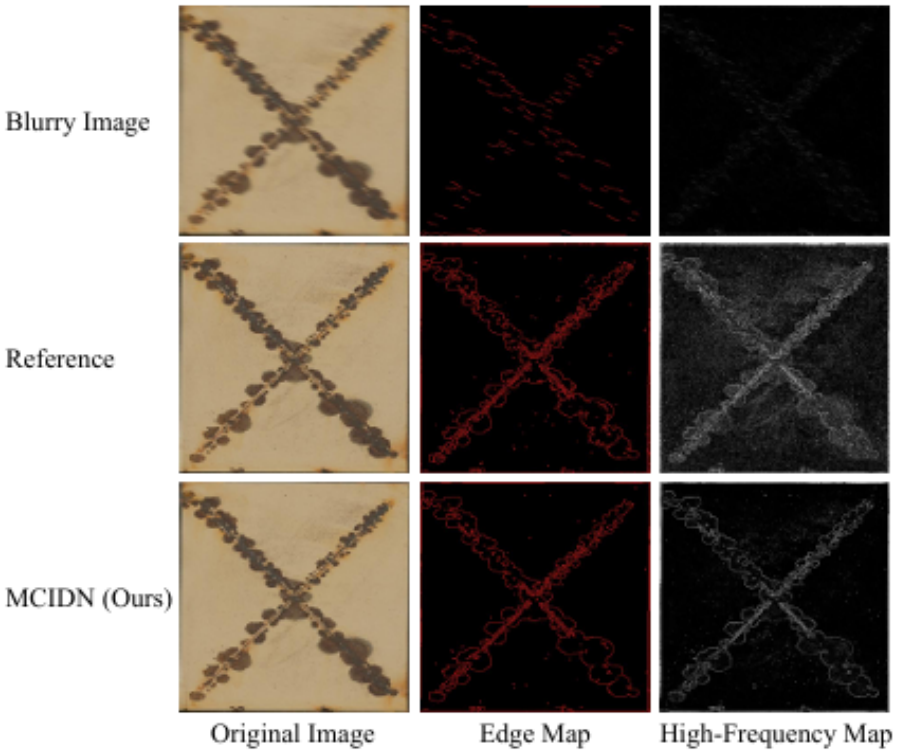


Figura 3.20: Aplicación industrial del deblurring en ciencia de materiales (imagen de [55])

El estudio destaca la importancia de obtener imágenes nítidas en la detección y análisis de corrosión, ya que el desenfoque compromete la precisión en la identificación de patrones de deterioro y en la toma de decisiones industriales. Para abordar este problema, los autores introducen una solución innovadora basada en el aprendizaje dual de dominios, combinando tanto el dominio espacial como el dominio de frecuencia en la restauración de imágenes.
Entre las principales contribuciones del artículo, se encuentran dos módulos clave:

- **Módulo de Atención de Canal Espacial (*SCAM*):** Una variante de autoatención que emplea convoluciones dinámicas para mejorar la representación de regiones borrosas sin incrementar significativamente la complejidad computacional.
- **Módulo de Atención de Canal de Frecuencia (*FCAM*):** Un mecanismo que enfatiza la recuperación de detalles mediante la manipulación selectiva de componentes de frecuencia

en las imágenes.

El artículo valida la eficacia de *MCIDN* mediante pruebas en un nuevo conjunto de datos de imágenes de corrosión borrosas y sus correspondientes versiones nítidas. Los resultados experimentales muestran que la red propuesta alcanza valores altos en las métricas *PSNR* y un *SSIM*, superando métodos previos en la restauración de detalles estructurales y texturas.
En conclusión, este trabajo presenta una solución específica para mejorar la calidad de imágenes de corrosión de metales, con aplicaciones en evaluación de infraestructuras, inspección de tuberías y mantenimiento predictivo. Además, plantea la posibilidad de integrar su método con enfoques de *Deep Learning* más avanzados para mejorar aún más la eficiencia en entornos industriales.

3.3.2. Discusión

En la Tabla 3.1 se realiza una comparación entre los artículos seleccionados y nuestra propuesta evaluando siete dimensiones relacionadas con conceptos tales como el *deblurring* o el *Deep Learning*. Cada columna de la siguiente tabla representa una dimensión específica, y cada celda indica si la propuesta de este proyecto aborda dicha dimensión. Las dimensiones consideradas son:

- **Tipo de Blur**: Indica si el artículo distingue los diferentes tipos de blur (*motion blur, defocus blur, gaussian blur, mixed blur...*).
- **Tipo de Deblurring**: Indica si el método distingue las categorías de *blind deblurring* y *non-blind deblurring*.
- **Deep Learning**: Determina si la técnica o técnicas utilizados están basadas en *Deep Learning* o en métodos físicos y matemáticos. Se marca **Sí** si usa *Deep Learning*, y **No** si emplea técnicas basadas en modelos físicos, regularización matemática, deconvolución...
- **Uso de Datos Reales**: Determina si el conjunto o conjuntos de imágenes considerados en el artículo han sido obtenidos en escenarios reales.
- **Calidad del Deblurring (Métricas de Evaluación)**: Indica si el artículo reporta valores de métricas como *PSNR, SSIM, LPIPS* o *FID* para evaluar la mejora de la calidad de las imágenes.
- **Consumo de Recursos Computacionales**: Especifica si el artículo hace mención al consumo de recursos computacionales de los diferentes modelos, comparando las diferencias entre ellos.
- **Aplicación en un Entorno Industrial**: Indica si el artículo presenta una aplicación directa en un contexto industrial.

Trabajo	Tipo de Blur	Tipo de Deblurring	Deep Learning	Uso de Datos Reales	Calidad del Deblurring	Consumo de Recursos	Aplicación en un Entorno Industrial
Amrollahi Biyouki et al.	No	Sí	Sí	Sí	Sí	No	No
Zhang et al.	Sí	Sí	Sí	Sí	Sí	No	No
Wu et al.	No	Sí	No	Sí	No	No	Sí
Kim et al.	No	No	No	Sí	No	No	Sí
Wang et al.	No	Sí	Sí	Sí	Sí	No	Sí
Nuestra propuesta	Sí	Sí	Sí	Sí	Sí	Sí	Sí

Tabla 3.1: Comparación de propuestas

El análisis del estado del arte permite así identificar diversas fortalezas y oportunidades en el ámbito del *deblurring* con *Deep Learning*, lo que respalda los objetivos de este proyecto. En primer lugar, los estudios revisados resaltan la importancia de comprender el fenómeno del desenfoque, sus causas y sus distintos tipos *(motion blur, defocus blur, gaussian blur, mixed blur)*. Este conocimiento es esencial para seleccionar la técnica de *deblurring* más adecuada y determinar qué modelos pueden adaptarse mejor según el tipo de degradación de las imágenes. Además, se ha evidenciado que el *deblurring* puede abordarse desde diferentes enfoques, diferenciando entre los métodos tradicionales basados en optimización y filtrado, y las soluciones más recientes fundamentadas en *Deep Learning*. La comparación de estas técnicas, que ha sido abordada en múltiples artículos del estado del arte, refuerza la necesidad de analizar qué métodos ofrecen mejores resultados en términos de calidad visual y eficiencia computacional.

Otro aspecto clave identificado en la literatura es el uso de diversas arquitecturas de *Deep Learning* en la restauración de imágenes, incluyendo *CNNs, GANs* y *RNNs*, cada una con sus propias ventajas y limitaciones en cuanto a precisión en la reconstrucción y velocidad de procesamiento. Los estudios revisados han demostrado que los modelos generativos y las redes convolucionales profundas pueden mejorar significativamente la recuperación de detalles en imágenes degradadas. Asimismo, algunos trabajos han aplicado estos modelos en imágenes industriales, lo que resalta la importancia de evaluar distintas técnicas de *deblurring* en este tipo de entornos. La medición del rendimiento mediante métricas de calidad como *PSNR* y *SSIM*, ampliamente utilizadas en los artículos analizados, permite establecer criterios objetivos para la evaluación de los modelos desarrollados en este estudio.

Por otro lado, la eficiencia computacional es un aspecto poco explorado en estas investigaciones, lo que representa una oportunidad para este proyecto. Dado que la implementación de modelos de *deblurring* en entornos industriales requiere un equilibrio entre precisión y rapidez, evaluar su viabilidad computacional es un punto clave para su aplicación práctica.

A partir de este análisis, este proyecto no solo se alinea con los avances recientes en la eliminación del desenfoque, sino que también busca ampliar el conocimiento existente y mejorar las soluciones actuales.

3.4. Fundamentos técnicos

Esta sección tiene como objetivo obtener una visión de las herramientas y/o tecnologías que se utilizarán en este proyecto. Además, se presentan algunos *datasets* para los modelos de *Deep Learning* implementados.

3.4.1. Lenguajes y librerías

Las principales herramientas *software* que se van a utilizar en el proyecto para la implementación de técnicas de *deblurring* mediante *Deep Lerning* vienen recogidas en la Figura 3.27.

Figura 3.21: Lenguaje *Python*

Figura 3.22: *IDE Visual Studio Code*

Figura 3.23: Librería *PyTorch*

Figura 3.24: Librería *OpenCV*

Figura 3.25: *Google Colab*

Figura 3.26: Repositorio en *GitHub*

Figura 3.27: Herramientas del proyecto

- **Python**: Lenguaje de programación de alto nivel ampliamente utilizado en inteligencia artificial y procesamiento de imágenes.
- **Visual Studio Code (VS Code)**: Un entorno de desarrollo integrado (*IDE*) ligero y potente, ideal para escribir, y ejecutar código en Python. Su compatibilidad con extensiones y herramientas como *Jupyter Notebooks* facilita el desarrollo de modelos de *Deep Learning*.
- **PyTorch**: Framework de *Deep Learning* ampliamente utilizado en la investigación y desarrollo de modelos de aprendizaje profundo. Se elige en este proyecto porque permite entrenar redes neuronales convolucionales (CNNs) y modelos avanzados de restauración de imágenes con facilidad, aprovechando la aceleración por GPU.
- **OpenCV**: Biblioteca de visión por computadora que proporciona herramientas para la manipulación y preprocesamiento de imágenes. En este proyecto, se usa para cargar imágenes, convertirlas a escala de grises si es necesario, aplicar filtros y realizar transformaciones previas antes de alimentar de imágenes a la red neuronal.
- **Google Colab**: Entorno de ejecución en la nube que permite entrenar modelos de *Deep Learning* con acceso gratuito a GPUs. Es especialmente útil para reducir los tiempos de

entrenamiento y realizar pruebas con mayor capacidad computacional, sin requerir una configuración local potente.

- **GitHub**: Plataforma de control de versiones y colaboración basada en Git. En este proyecto se utiliza para acceder a los repositorios de los autores de los diferentes modelos de *Deep Learning* considerados.

3.4.2. *Datasets* para *Deblurring*

En esta sección se presentan algunos de los datasets más utilizados en la literatura para entrenar y evaluar modelos de *Deep Learning* en la tarea de eliminación de *blur*.

Köhler et al. dataset

El conjunto de datos de Köhler et al. [25] contiene un total de 48 imágenes borrosas. Estas imágenes se generan aplicando 12 tipos diferentes de *kernels* de desenfoque no uniformes a 4 imágenes originales (ver Figura 3.28). Los *kernels* de desenfoque se obtienen mediante la grabación de trayectorias 6D del movimiento de la cámara y simulando efectos reales de desenfoque en un entorno de laboratorio.

Figura 3.28: Köhler dataset (imagen de [25])

Sun et al. dataset

Este conjunto de datos [50] aplica el mismo conjunto de 8 *kernels* de desenfoque uniformes utilizados en el trabajo de Levin et al. [28], pero a un conjunto más amplio de escenas reales. En total, se incluyen 80 imágenes originales, cada una con múltiples versiones borrosas, lo que da lugar a un total de 640 imágenes desenfocadas (ver Figura 3.29).

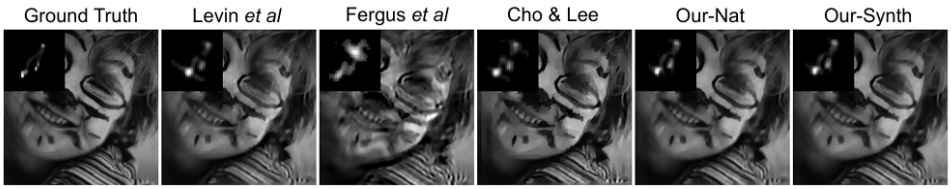


Figura 3.29: Sun et al. dataset (imagen de [50])

Lai et al. dataset

Lai et al. [26] proponen un conjunto de datos con imágenes que contienen tanto desenfoque espacialmente variable como desenfoque uniforme (ver Figura 3.30). Para generar las imágenes borrosas con desenfoque variable, se utilizan trayectorias de cámara reales obtenidas de sensores de teléfonos móviles. Además, se emplean 8 *kernels* de desenfoque (tanto uniformes como no uniformes) en 25 imágenes base, agregando ruido gaussiano del 1 % para simular ruido de la cámara.

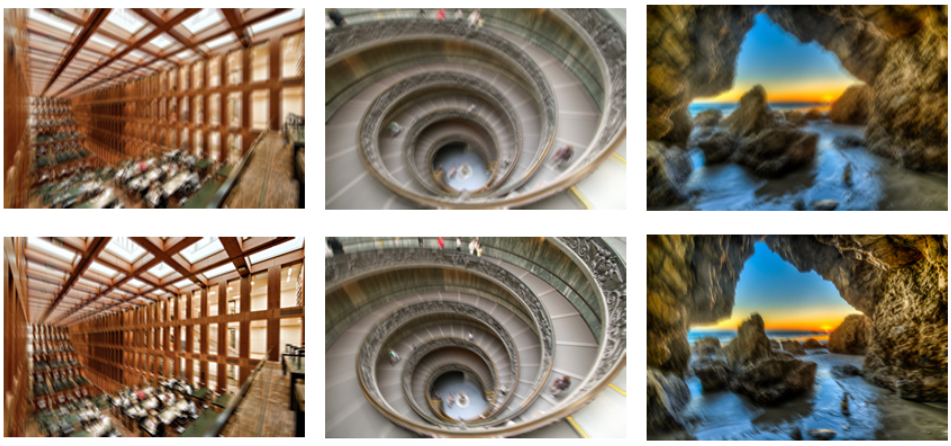


Figura 3.30: Lai et al. dataset (imagen de [26])

DeepVideoDeblurring (DVD) dataset

Este conjunto de datos [49] consta de 6,708 fotogramas borrosos tomados de 71 videos junto con sus imágenes nítidas correspondientes (ver Figura 3.31). Las imágenes borrosas se generan aplicando exposiciones largas artificiales mediante la combinación de múltiples exposiciones cortas. Además, el dataset se expande mediante transformaciones como volteo, escalado y rotación, alcanzando un total de más de 2 millones de recortes de imagen.

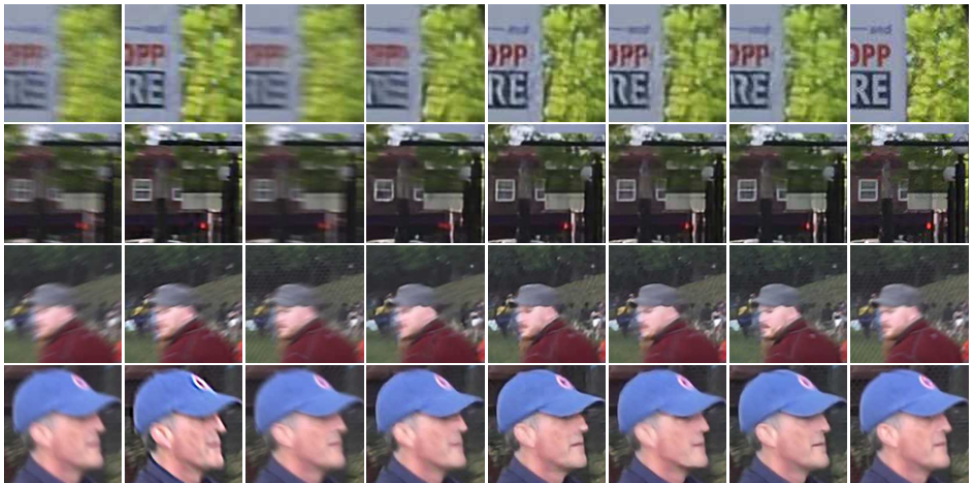


Figura 3.31: DeepVideoDeblurring dataset (imagen de [49])

GoPro dataset

El conjunto de datos de GoPro [32] es uno de los más utilizados para entrenar modelos de *deblurring*. Contiene 3,214 pares de imágenes borrosas y nítidas, capturadas con cámaras GoPro a una resolución de 1280×720 píxeles (ver Figura 3.32). Para generar las imágenes borrosas, se promedian varios fotogramas sucesivos de video, de manera que el fotograma intermedio se considera la imagen nítida de referencia.

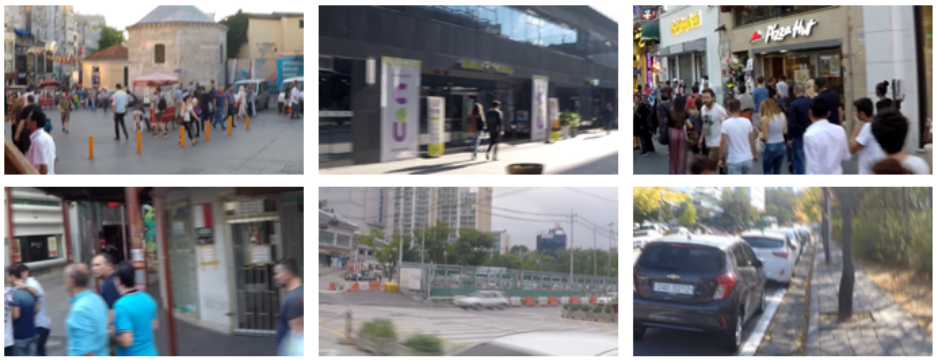


Figura 3.32: GoPro dataset (imagen de [32])

HIDE dataset

Este conjunto de datos [47] incluye imágenes borrosas obtenidas de diversas escenas con objetos en movimiento significativo en primer plano (ver Figura 3.33). Para generar las imágenes desenfocadas, se promedian 11 fotogramas secuenciales de video y se toma el fotograma medio como imagen borrosa de referencia. El dataset contiene imágenes de entornos con alta densidad de población, incluyendo escenas de primeros planos y tomas a larga distancia.

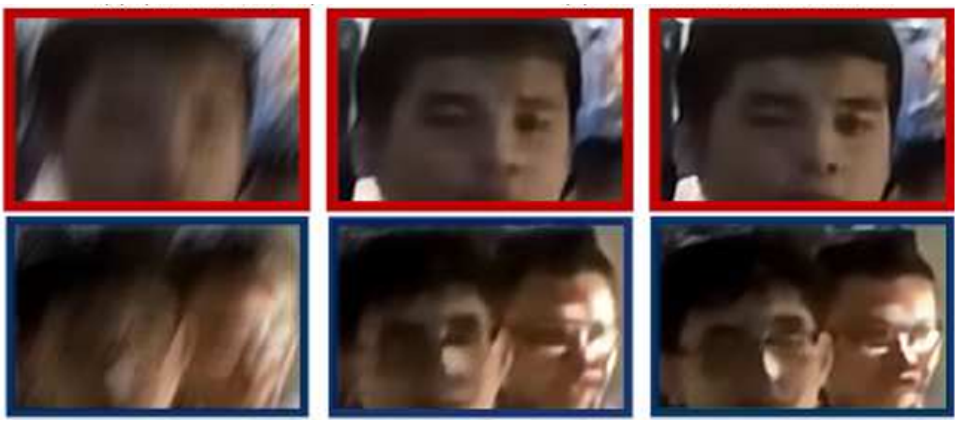


Figura 3.33: Hide dataset (imagen de [32])

RealBlur dataset

El conjunto de datos RealBlur [40] incluye un total de 4,738 pares de imágenes borrosas y nítidas (ver Figura 3.34). A diferencia de otros datasets que generan desenfoque sintético, este conjunto de datos se obtiene mediante un sistema real de adquisición de imágenes, con alineación geométrica y fotométrica para capturar imágenes borrosas de manera realista. Se ha demostrado que entrenar modelos con este dataset mejora el desempeño en la restauración de imágenes del mundo real.

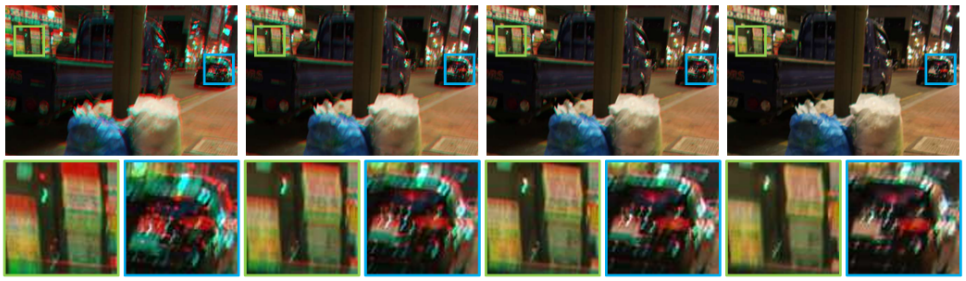


Figura 3.34: Real Blur dataset (imagen de [40])

Parte II

Desarrollo de la solución

Capítulo 4

Diseño experimental

La solución a desarrollar consiste en la implementación de diversos modelos *Deep Learning* para la tarea de *deblurring* sobre diferentes *datasets*. Una vez implementados, se evaluarán y compararán dichos modelos con el fin de identificar los más eficaces. Estos modelos serán posteriormente aplicados a conjuntos de imágenes provenientes de entornos industriales reales, donde la mejora de la calidad visual tiene un impacto significativo. En este capítulo se detalla la fase de diseño de la solución, incluyendo las arquitecturas consideradas para la experimentación, las métricas de éxito, y el plan de aceptación.

4.1. Arquitecturas consideradas

Se han seleccionado tres de las mejores arquitecturas de *Deep Learning* para la tarea de *deblurring* para su posterior entrenamiento. Para todas ellas, utilizamos como referencia el *GoPro dataset* (ver Figura 3.32).

NAFNet

Esta arquitectura, propuesta por S. Nah et al. en [32], emplea la técnica de *Multi-scale Processing* estudiada en la Subección 3.2.5. NAFNet está diseñada para abordar la tarea de restauración de imágenes de manera eficiente, reduciendo significativamente la complejidad del modelo al eliminar activaciones no lineales innecesarias. Sigue una estructura jerárquica con bloques simples y eficientes, lo que permite una reconstrucción precisa de los detalles perdidos debido al desenfoque, manteniendo un rendimiento competitivo en términos de velocidad y calidad visual.

Restormer

Restormer, propuesta por S. W. Zamir et al. en [58], es una arquitectura basada en la arquitectura Transformer (consultar [54]) optimizada para tareas de restauración de imágenes de alta resolución. A diferencia de las arquitecturas tradicionales que dependen principalmente de convoluciones, Restormer emplea mecanismos de atención de largo alcance y una estructura jerárquica que permite capturar tanto información global como local de manera eficiente. Restormer

está diseñada específicamente para tareas de *deblurring*, superresolución y eliminación de ruido, logrando resultados de alta calidad con un coste computacional moderado.

DeblurGAN

DeblurGAN, introducida por O. Kupyn et al. en [24], aborda el problema del desenfoque mediante redes generativas adversarias (*GANs*). Utiliza una arquitectura tipo *encoder-decoder* con bloques residuales, donde la red generadora intenta reconstruir imágenes nítidas a partir de imágenes borrosas, mientras que la red discriminadora evalúa la autenticidad de las imágenes generadas. La combinación de la pérdida adversarial con la pérdida de contenido basada en VGG permite a DeblurGAN producir resultados visualmente realistas, siendo una de los primeras arquitecturas en aplicar GANs de forma efectiva en la tarea de *deblurring*.

4.2. Métricas de éxito

En esta sección se definen las métricas de éxito escogidas para determinar si el modelo ha alcanzado los objetivos planteados en el inicio del proyecto. Se utilizarán las dos métricas más reconocidas en la evaluación de modelos de restauración de imágenes: el índice SSIM (*Structural Similarity Index*) [19] y el índice PSNR (*Peak Signal-to-Noise Ratio*) [57]. Como se ilustró en el Capítulo 3, ambas métricas son comúnmente empleadas en estudios de superresolución para medir la calidad de las imágenes generadas en comparación con las originales.

- ***Peak Signal-to-Noise Ratio (PSNR)***. Es una métrica que evalúa la calidad de la imagen en relación con el ruido. Se define como

$$PSNR(I, \hat{I}) = 10 \log \frac{R^2}{MSE(I, \hat{I})} \quad , \tag{4.1}$$

  donde:

  - R: Máximo valor de píxel posible en la imagen (para imágenes en formato de 8 bits, $R = 255$).
  - $MSE(I, \hat{I})$: Error cuadrático medio entre la imagen real y la restaurada.

  El PSNR mide la relación entre el valor máximo posible de una señal (en este caso, el valor máximo de píxel) y la cantidad de ruido presente en la señal restaurada. Se expresa en decibelios (dB), y un valor más alto indica mejor calidad de la imagen. En este caso, se buscará un valor de PSNR superior o igual a 30 dB en las imágenes restauradas.

- ***Structural Similarity Index Measure (SSIM)***. Mide la similitud estructural entre la imagen original y la restaurada, comparando patrones de intensidades de píxeles. Se define como

$$SSIM(I, \hat{I}) = \frac{(2\mu_I \mu_{\hat{I}} + C_1)(2\sigma_{I\hat{I}} + C_2)}{(\mu_I^2 + \mu_{\hat{I}}^2 + C_1)(\sigma_I^2 + \sigma_{\hat{I}}^2 + C_2)} \quad , \tag{4.2}$$

  donde:

  - $\mu_I, \mu_{\hat{I}}$: Medias de los valores de píxeles en las imágenes original y restaurada.

- $\sigma_I^2, \sigma_{\hat{I}}^2$: Varianzas de las imágenes.
- $\sigma_{I\hat{I}}$: Covarianza entre la imagen original y la restaurada.
- C_1, C_2: Constantes utilizadas para evitar inestabilidades en la ecuación.

El SSIM evalúa la similitud estructural entre dos imágenes, teniendo en cuenta factores como el brillo, el contraste y la estructura de la imagen. Su valor oscila entre -1 y 1, aunque es negativo en casos muy puntuales, en general su valor está entre 0 y 1. Un valor de SSIM más cercano a 1 indica una mayor similitud entre las imágenes. Esta métrica es particularmente útil para medir la calidad perceptual de las imágenes restauradas. Para el propósito de este trabajo, se espera que cada modelo alcance un valor de SSIM igual o superior a 0.93

Estas métricas, junto con los umbrales definidos, permitirán evaluar de forma objetiva si los modelos implementados cumplen con los requisitos de calidad establecidos y, por tanto, si el proyecto ha alcanzado un nivel satisfactorio de éxito técnico en la tarea de restauración de imágenes.

4.3. Plan de aceptación

En esta sección se estudia el diseño del plan de aceptación, que asegura una evaluación precisa del proyecto, junto con los recursos necesarios para llevarlo a cabo.

4.3.1. Recursos

A continuación, se describen los detalles de los recursos necesarios para llevar a cabo el plan de aceptación.

Entorno de Desarrollo

El entorno de desarrollo utilizado para la construcción y evaluación del modelo combina herramientas locales y basadas en la nube, facilitando tanto el desarrollo como la ejecución eficiente de los experimentos.

- **Entorno Local en Visual Studio Code:** *Visual Studio Code (VSCode)* es un editor de código ligero pero potente que es compatible con múltiples lenguajes de programación, incluyendo Python. Este entorno se utilizó principalmente para el desarrollo de scripts, edición de código y pruebas iniciales en CPU. Las principales herramientas y librerías utilizadas son:
  - **Python**: El lenguaje principal utilizado para implementar el modelo.
  - **PyTorch**: La biblioteca principal para la construcción y entrenamiento de redes neuronales profundas. PyTorch ofrece soporte para procesamiento en GPU y optimización avanzada de redes neuronales.
  - **NumPy y OpenCV**: Librerías auxiliares para la manipulación de imágenes y matrices, necesarias para procesar los datos y realizar las operaciones requeridas por el modelo.

  - **Otras dependencias**: Se instalaron dependencias adicionales como `matplotlib` para visualización y `lmdb` para almacenar y recuperar datos de forma eficiente.

- **Entorno de Ejecución en Google Colab:** Para acelerar los procesos de entrenamiento y aprovechar el procesamiento de las GPUs, se utilizó *Google Colab*, una plataforma basada en la nube que ofrece acceso gratuito a entornos de ejecución con GPU. El entorno configurado en Google Colab incluye:

  - **PyTorch y CUDA**: Se aprovechó la aceleración de *hardware* proporcionada por las GPUs, usando la versión de *PyTorch* compatible con la arquitectura CUDA, lo que permitió entrenar los modelos de manera mucho más eficiente en comparación con la ejecución solo en CPU.
  - **Google Drive**: Para almacenar y gestionar los datos de entrada y los modelos entrenados. Google Drive permite guardar los archivos de forma persistente y acceder a ellos durante las sesiones de trabajo en Colab, garantizando la continuidad del proceso de entrenamiento.

Hardware

Los recursos de *hardware* disponibles en este proyecto son los siguientes:

- Máquina local: Se utilizó una máquina con un AMD Ryzen 5 como procesador, sin aceleración por GPU (CPU solamente), para las fases iniciales de pruebas.
- Google Colab: En Google Colab, se aprovecharon las GPUs proporcionadas por la plataforma para realizar entrenamientos más largos y exigentes. La instancia de GPU utilizada en Colab ha sido la GPU NVIDIA T4, que permite un rendimiento considerablemente mayor durante el entrenamiento de modelos complejos.
- Máquina virtual: Se utilizó una máquina virtual del Departamento de Informática con una GPU NVIDIA RTX A40 para llevar a cabo los entrenamientos de los diferentes modelos.

4.3.2. Fases del Plan

Fase 1: Preparación del entorno de evaluación

En esta fase se garantiza que todos los recursos necesarios para la evaluación estén disponibles y correctamente configurados. Se comienza con la verificación de los entornos de desarrollo y ejecución, específicamente Visual Studio Code para pruebas en local y Google Colab con soporte de GPU para pruebas más exigentes. A continuación, se realiza la carga y organización del conjunto de datos GoPro, asegurando su disponibilidad para las pruebas iniciales. También se revisan las dependencias, configuraciones, scripts y herramientas de medición necesarias, tales como los módulos para el cálculo de PSNR y SSIM, con el fin de validar que el entorno técnico está completamente operativo antes de iniciar la evaluación formal.

Fase 2: Evaluación sobre el dataset GoPro

Una vez preparado el entorno, se ejecutan los modelos entrenados sobre el conjunto de imágenes de prueba del dataset GoPro. Esta fase tiene como objetivo medir el desempeño de los modelos en un entorno controlado y estandarizado. Se aplican los modelos seleccionados —NAFNet, Restormer y DeblurGAN— al conjunto de test, obteniendo resultados tanto visuales como cuantitativos. Estos resultados permiten una primera evaluación del rendimiento del producto en condiciones conocidas.

Fase 3: Extracción y análisis de métricas

En esta fase se analizan los resultados obtenidos durante la evaluación anterior. El análisis se realiza de forma cuantitativa mediante el cálculo de las métricas PSNR y SSIM para cada modelo, lo cual permite una comparación objetiva de su rendimiento. Además, se realiza una evaluación cualitativa basada en la observación visual de las imágenes restauradas, prestando especial atención a defectos comunes como artefactos, pérdida de textura o halos. También se verifica si los valores obtenidos superan los umbrales establecidos previamente.

Fase 4: Evaluación sobre imágenes industriales

Con los modelos validados sobre el dataset GoPro, se procede a una segunda ronda de pruebas en un entorno más realista, utilizando imágenes provenientes de entornos industriales. Esta fase tiene como finalidad estudiar la capacidad de los modelos para generalizar a situaciones no vistas durante el entrenamiento. Se vuelven a aplicar los modelos NAFNet, Restormer y DeblurGAN sobre este nuevo conjunto de imágenes, recopilando nuevamente los resultados visuales y métricos de la restauración obtenida en estas condiciones.

Fase 5: Extracción y análisis de métricas en entorno industrial

Finalmente, se realiza un análisis detallado de los resultados obtenidos en el contexto industrial. Al igual que en la Fase 3, se calculan las métricas PSNR y SSIM para evaluar cuantitativamente la calidad de las imágenes restauradas, y se efectúa una observación visual minuciosa con el fin de detectar defectos relevantes. Esta fase permite evaluar la capacidad de generalización de los modelos a condiciones de uso reales, así como su aplicabilidad en escenarios industriales, comparando su desempeño con el observado previamente en el dataset GoPro.

4.3.3. Recursos para cada Fase del Plan de Aceptación

En esta subsección se especifica, para cada una de las cinco fases del Plan de Aceptación, el conjunto de recursos que asegura la ejecución precisa de dicho plan, y coherencia respecto a las métricas definidas.

- **Fase 1 – Preparación del entorno de evaluación:** Requiere el uso de *Visual Studio Code* como entorno de desarrollo local y *Google Colab* como entorno de ejecución con GPU (NVIDIA T4), junto con la instalación de librerías como PyTorch, NumPy, OpenCV y Matplotlib.
- **Fase 2 – Evaluación sobre el dataset GoPro:** Se utiliza el conjunto de datos GoPro [32], con imágenes borrosas y sus correspondientes referencias nítidas. Los modelos se ejecutan en Colab y se procesan imágenes de test mediante scripts desarrollados en Python.
- **Fase 3 – Extracción y análisis de métricas:** Las métricas PSNR y SSIM se calculan automáticamente mediante funciones implementadas en Python. Se requiere acceso a los resultados generados por cada modelo.
- **Fase 4 – Evaluación sobre imágenes industriales:** Requiere los mismos recursos que la Fase 2 con la diferencia de que no se utiliza el *dataset GoPro*, sino imágenes provenientes de entornos industriales reales o simulados.
- **Fase 5 – Extracción y análisis de métricas en entorno industrial:** Requiere los mismos recursos que la Fase 3.

4.3.4. Resultados de cada Fase del Plan de Aceptación

En esta subsección se recopilan los resultados generados por cada una de las fases del Plan de Aceptación, destacando los principales logros, los procedimientos realizados y las conclusiones derivadas de cada una de ellas.

- **Fase 1 – Preparación del entorno de evaluación:** El resultado de esta fase es la correcta configuración de los entornos de desarrollo y ejecución, utilizando herramientas como VSCode y Google Colab. Se instalaron todas las dependencias necesarias (bibliotecas, frameworks y utilidades), asegurando la compatibilidad con los modelos a evaluar. Además, se verificó el acceso adecuado al *dataset GoPro*, y se prepararon scripts automatizados para la ejecución de las evaluaciones y la recolección de métricas. Esta fase garantiza que los experimentos puedan ser reproducibles y comparables en condiciones controladas.
- **Fase 2 – Evaluación sobre el *dataset GoPro*:** Durante esta fase, se aplicaron los modelos seleccionados al *dataset GoPro*, que actúa como referencia estándar en el campo del *deblurring*. Para cada imagen procesada por los modelos, se calcularon dos métricas clave (por ejemplo, PSNR y SSIM), con el fin de cuantificar la calidad de la reconstrucción en términos objetivos. El resultado es un conjunto de valores métricos por modelo que permiten evaluar su rendimiento.
- **Fase 3 – Extracción y análisis de métricas:** Con los resultados obtenidos en la Fase 2, se procedió a generar un ranking de modelos basado en el promedio de las métricas calculadas.

Además del análisis cuantitativo, se realizó una verificación respecto a los umbrales mínimos definidos como aceptables (por ejemplo, PSNR >28 dB y SSIM >0.9). Se documentaron observaciones relevantes sobre el comportamiento de cada modelo, destacando fortalezas y limitaciones.

- **Fase 4 – Evaluación sobre imágenes industriales:** Esta fase consistió en aplicar los mismos modelos, ya validados sobre el *dataset GoPro*, a un conjunto representativo de imágenes obtenidas en entornos industriales reales, las cuales suelen presentar mayor ruido, variabilidad y condiciones no controladas. Se utilizaron nuevamente las métricas seleccionadas para evaluar el rendimiento, permitiendo comprobar la capacidad de generalización de los modelos frente a escenarios prácticos.
- **Fase 5 – Extracción y análisis de métricas en entorno industrial:** Finalmente, se extrajeron y analizaron los valores métricos obtenidos en la Fase 4, con el objetivo de establecer un nuevo ranking de modelos según su desempeño en entornos industriales. Se verificó si los modelos mantenían un rendimiento aceptable en comparación con los umbrales definidos previamente, lo que permitió confirmar su idoneidad para su aplicación en casos reales. Se incluyó un análisis cualitativo para evaluar aspectos no reflejados por las métricas.

Capítulo 5

Experimentación

En este capítulo se aborda el proceso de implementación y entrenamiento de los modelos considerados, incluyendo las pruebas realizadas a los mismos para verificar su correcto funcionamiento, y el análisis de los resultados del entrenamiento.

5.1. Proceso de implementación de los modelos

El proceso de implementación de los modelos ha seguido una metodología iterativa y basada en pruebas controladas. Para llevar a cabo el entrenamiento, se ha optado por el uso del framework *PyTorch* del lenguaje de programación *Python*, ampliamente utilizado en aplicaciones de *Deep Learning* por su flexibilidad y rendimiento. Como se ha mencionado al comienzo del capítulo, el conjunto de datos empleado es el *GoPro dataset*, un benchmark reconocido para tareas de *deblurring* que contiene pares de imágenes borrosas y nítidas capturadas con cámaras *GoPro*.

Inicialmente, se han clonado los repositorios correspondientes a cada modelo tanto en el entorno local de desarrollo de *Visual Studio Code* como en *Google Colab*, donde se ha realizado el entrenamiento final aprovechando las GPUs gratuitas que proporciona la plataforma. Esta decisión se tomó tras constatar que los recursos computacionales del equipo local no eran suficientes para manejar el volumen de datos ni la complejidad del entrenamiento completo. El proceso comenzó con pruebas preliminares en local utilizando un subconjunto reducido del conjunto de datos. Estas pruebas permitieron verificar el correcto funcionamiento del código y de las dependencias de cada modelo, así como validar la configuración de los entornos de ejecución. Una vez comprobada la viabilidad técnica, se procedió a ejecutar los entrenamientos completos en *Google Colab*, donde fue necesario montar el *GoPro dataset* en el entorno de trabajo a través de Google Drive.

Durante el proceso de configuración, uno de los principales retos fue la gestión de dependencias, ya que algunos repositorios presentaban versiones obsoletas de librerías o incompatibilidades con versiones actuales. Estos problemas se resolvieron analizando las especificaciones originales, buscando documentación actualizada y realizando ajustes manuales en los archivos de requerimientos. Otro desafío importante fue la gestión del espacio de almacenamiento en Google Drive, debido al tamaño considerable del *GoPro dataset*. Para solucionarlo, se optó por fraccionar el conjunto de datos y cargarlo por partes durante las distintas fases del entrenamiento, asegurando en todo momento la integridad y consistencia del proceso.

5.2. Casos de prueba para los modelos

Para verificar el adecuado funcionamiento de los modelos se definen y ejecutan una serie de casos de prueba, cuyos resultados son posteriormente analizados para identificar defectos y áreas de mejora, con el fin de validar el producto bajo condiciones representativas.

Se diseñaron cuatro casos de prueba diferenciados. Los tres primeros casos se corresponden con un tipo específico de desenfoque que los modelos de *deblurring* deben ser capaces de corregir: *motion blur* (ver Figura 5.1), *gaussian blur* (ver Figura 5.2) y *out-of-focus blur* (ver Figura 5.3), todos ellos definidos en la Subsección 3.2.5. El cuarto caso, denominado "desenfoques compuestos" (ver Figura 5.4), se planteó con el objetivo de evaluar la capacidad de generalización de los modelos ante condiciones más realistas y complejas, incluyendo imágenes con diferentes tipos de *blur*.

Imagen original Imagen restaurada

Figura 5.1: Ejemplo de imagen con *motion blur* (imagen de [32])

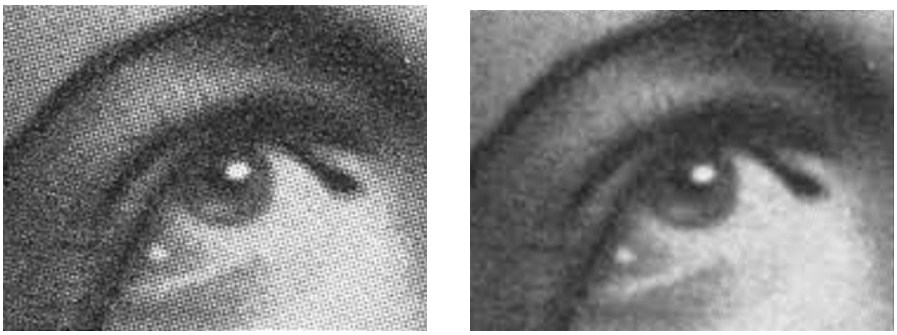


Imagen original Imagen restaurada

Figura 5.2: Ejemplo de imagen con *gaussian blur* (imagen de [67])

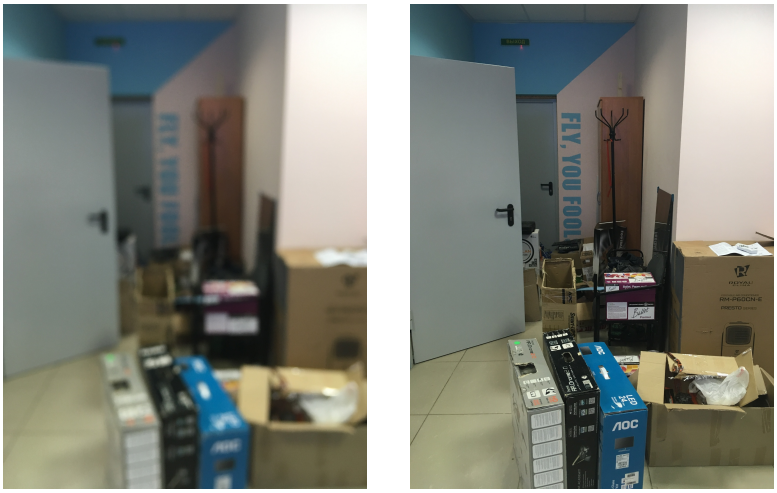


Imagen original Imagen restaurada

Figura 5.3: Ejemplo de imagen con *out-of-focus blur* (imagen de [40])

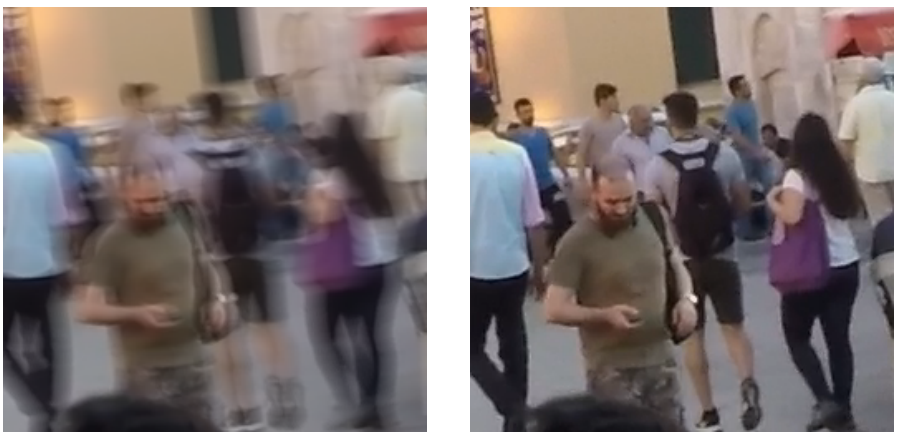


Imagen original Imagen restaurada

Figura 5.4: Ejemplo de imagen con desenfoques compuestos (imagen de [32])

Para cada uno de los casos, se definieron los siguientes elementos:

- **Entradas:** Imágenes con diferentes tipos y niveles de *blur*, adecuadas al tipo de caso analizado.
- **Resultados esperados:** Imágenes reconstruidas visualmente nítidas, que recuperen el contenido original con fidelidad, y sin introducir distorsiones o artefactos visuales.
- **Criterios de aceptación:**
  - PSNR (Peak Signal-to-Noise Ratio) superior a 25 dB.
  - SSIM (Structural Similarity Index) superior a 0,85.
  - Ausencia de artefactos visuales, es decir, que la imagen reconstruida no presente defectos perceptibles al ojo humano que afecten negativamente a la calidad visual, tales como halos, ruido artificial, contornos fantasmas o distorsiones cromáticas, introducidos durante el proceso de deblurring.

Cabe destacar que los valores mínimos establecidos en esta sección (PSNR $>$ 25dB y SSIM $>$ 0,85) tienen como objetivo verificar el correcto funcionamiento del producto, es decir, procesamiento y mejora de las imágenes, y entrega resultados técnicamente válidos. Los umbrales definidos para las métricas de calidad final, más exigentes, se detallan en la Sección 4.2, e implican que el producto no solo mejora la calidad de las imágenes, sino que lo hace a un nivel alto, competitivo y profesional.

Ejecución de las pruebas

Se aplicaron los modelos seleccionados (NAFNet, Restormer, DeblurGAN) a cada conjunto de imágenes de prueba. Las pruebas se ejecutaron en un entorno controlado, utilizando los mismos parámetros para asegurar una comparación justa. Los resultados obtenidos se almacenaron y clasificaron en función del modelo. Los resultados de NAFNet se presentan en la Tabla 5.1, los de Restormer en la Tabla 5.2, y los de DeblurGAN en la Tabla 5.3.

Caso de Prueba	PSNR (dB)	SSIM	Artefacto Visual
Caso 1 (*Motion Blur*)	26,3	0,88	No
Caso 2 (*Gaussian Blur*)	26,8	0,89	No
Caso 3 (*Out-of-Focus Blur*)	25,7	0,86	No
Caso 4 (Desenfoques Compuestos)	25,2	0,85	No

Tabla 5.1: Evaluación del modelo NAFNet por caso de prueba

Caso de Prueba	PSNR (dB)	SSIM	Artefacto Visual
Caso 1 (*Motion Blur*)	27,1	0,89	No
Caso 2 (*Gaussian Blur*)	27,6	0,90	No
Caso 3 (*Out-of-Focus Blur*)	26,4	0,87	No
Caso 4 (Desenfoques Compuestos)	26,0	0,86	No

Tabla 5.2: Evaluación del modelo Restormer por caso de prueba

Caso de Prueba	PSNR (dB)	SSIM	Artefacto Visual
Caso 1 (*Motion Blur*)	25,4	0,85	No
Caso 2 (*Gaussian Blur*)	25,8	0,86	No
Caso 3 (*Out-of-Focus Blur*)	25,1	0,85	No
Caso 4 (Desenfoques Compuestos)	24,9	0,84	No

Tabla 5.3: Evaluación del modelo DeblurGAN por caso de prueba

Análisis de las pruebas

Los resultados fueron analizados utilizando métricas objetivas (PSNR, SSIM) y evaluación visual subjetiva. El análisis de los resultados ha permitido identificar no solo el grado de cumplimiento de los criterios establecidos, sino también ciertas limitaciones comunes observadas durante la evaluación. Una de las principales deficiencias observadas ha sido la pérdida de detalle en texturas finas, especialmente en aquellas regiones donde la imagen original presentaba patrones regulares o estructuras complejas, como superficies rugosas o elementos arquitectónicos. También, se ha detectado menor eficiencia en el rendimiento de los modelos ante desenfoques compuestos. Mientras que los modelos tienden a comportarse de forma aceptable ante un tipo de desenfoque aislado, su desempeño es ligeramente inferior cuando se presentan múltiples tipos de desenfoque superpuestos, una situación común en escenarios reales no controlados. Otra limitación ha sido la presencia de artefactos visuales en los bordes de objetos con alto contraste, lo cual puede generar contornos duplicados, halos de luz o ruido localizado.

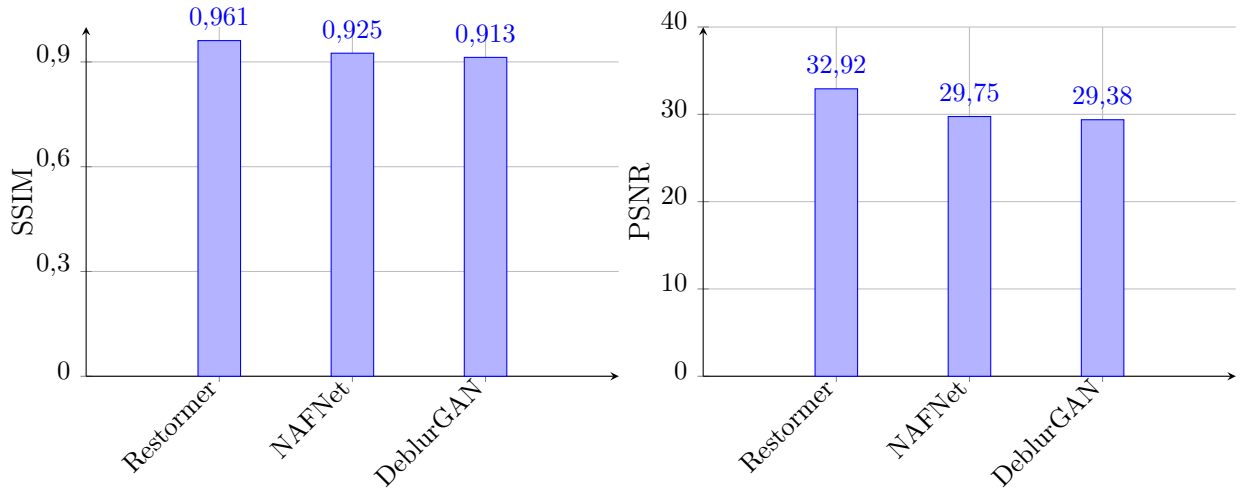
Es importante señalar que las pruebas descritas se han realizado utilizando modelos parcialmente entrenados, cuyo objetivo principal era verificar el correcto funcionamiento del proceso de entrenamiento. Esta circunstancia explica la presencia de ciertos artefactos visuales en algunos casos de prueba, así como unos valores de PSNR y SSIM que, si bien superan los umbrales mínimos establecidos, no alcanzan el rendimiento óptimo esperado tras un entrenamiento completo. A pesar de estas limitaciones, los resultados obtenidos son suficientes para confirmar que el producto está adecuadamente construido y que cumple con los requisitos funcionales establecidos en esta fase del proyecto. Tal como se indicó anteriormente, el propósito de esta evaluación no es maximizar el rendimiento de los modelos, sino asegurar su operatividad, coherencia, y adecuación a los objetivos establecidos.

5.3. Resultados de *train*

En esta sección se presentan y analizan diferentes datos obtenidos a lo largo del proceso de entrenamiento de los modelos. A diferencia de los resultados mostrados en la Sección 5.2, estos resultados han sido obtenidos tras un entrenamiento completo para cada uno de los modelos.

5.3.1. Métricas con el *dataset GoPro*

En la Figura 5.5 se muestran las métricas obtenidas por cada modelo en el conjunto de *train* del *dataset GoPro*.



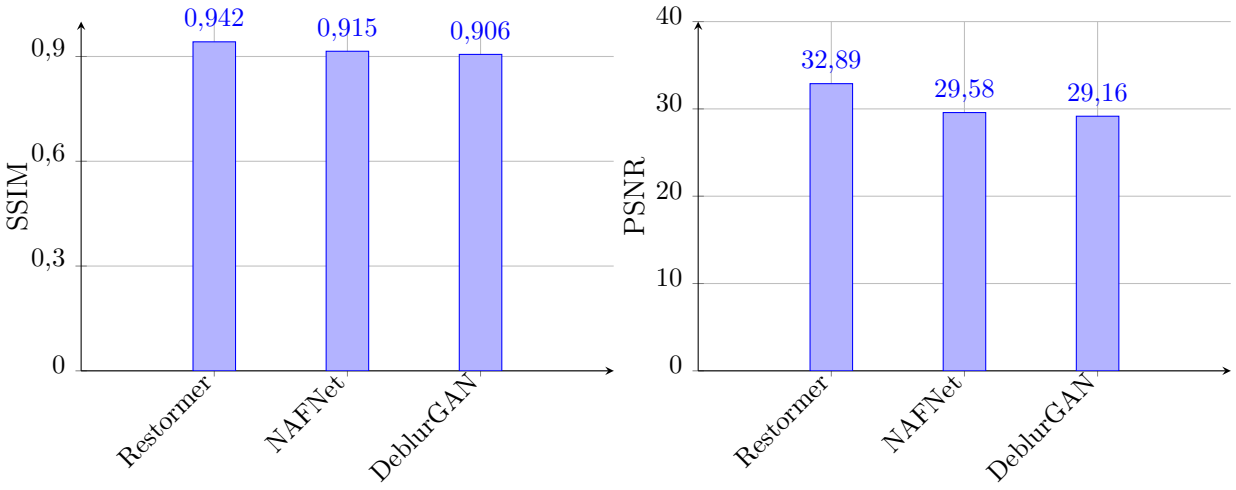
(a) SSIM (b) PSNR

Figura 5.5: Comparación de métricas entre modelos usando el *dataset GoPro* (*train*)

Al comparar los resultados obtenidos con los umbrales establecidos (PSNR $\geq$ 30 dB y SSIM $\geq$ 0.93), se observa que solo el modelo Restormer logra cumplir ambos criterios. En el caso de NAFNet y DeblurGAN, los valores de PSNR se sitúan ligeramente por debajo del umbral, mientras que sus SSIM también se mantienen por debajo de 0.93, aunque en ambos casos los resultados son bastante cercanos.

5.3.2. Métricas con un *dataset* de imágenes industriales

En la Figura 5.6 se recogen los resultados de las métricas SSIM y PSNR de cada uno de los modelos en el conjunto de *train* del *dataset* de imágenes industriales considerado. Este *dataset* contiene imágenes de piezas industriales. Dichas imágenes no se pueden mostrar en esta memoria por un acuerdo de confidencialidad firmado con la empresa proveedora de las mismas.



(a) SSIM (b) PSNR

Figura 5.6: Comparación de métricas entre modelos usando un *dataset* industrial (*train*)

5.3.3. Evolución de la Función de Pérdida

La evolución de la función de pérdida durante el entrenamiento permite analizar la convergencia de cada modelo y detectar posibles problemas como sobreajuste o aprendizaje insuficiente. En la Figura 5.7, se muestran las curvas de pérdida obtenidas para los tres modelos considerados durante el entrenamiento con el *dataset GoPro*.

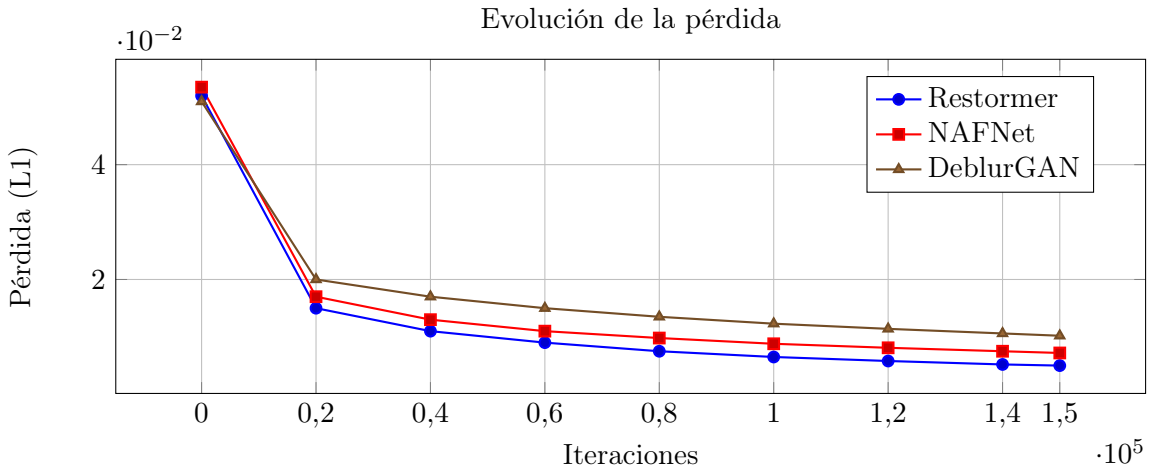


Figura 5.7: Evolución de la pérdida (L1) durante el entrenamiento.

Los resultados muestran que los tres modelos presentan una tendencia claramente decreciente en la función de pérdida, lo que indica una correcta convergencia durante el entrenamiento. Restormer alcanza los valores más bajos de pérdida, seguido de NAFNet y DeblurGAN. Esta diferencia sugiere que Restormer es capaz de ajustar sus pesos de manera más eficiente, lo cual concuerda con los resultados métricos obtenidos en las fases de evaluación.

5.3.4. Tiempo Medio por Iteración

El tiempo medio por iteración es un indicador relevante del coste computacional de cada modelo. Su análisis permite comparar la eficiencia de entrenamiento y valorar su viabilidad en entornos con recursos limitados. En la Figura 5.8, se muestran el tiempo medio por iteración para los tres modelos considerados durante el entrenamiento con el *dataset GoPro*.

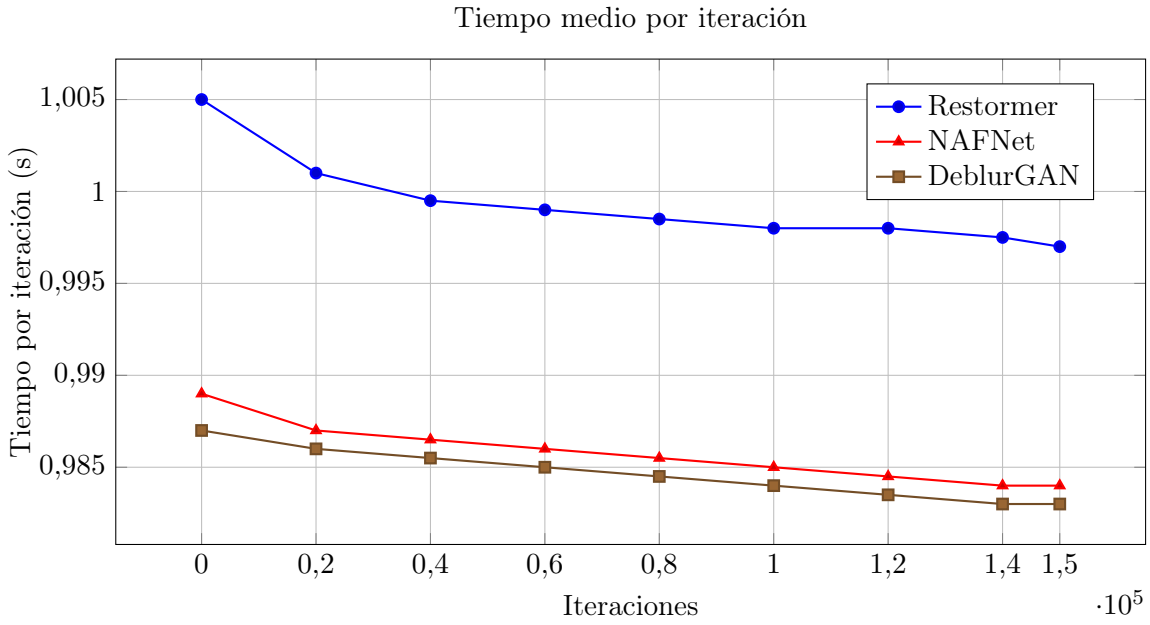


Figura 5.8: Tiempo medio por iteración para cada modelo hasta 150,000 iteraciones

En cuanto al tiempo medio por iteración, se observa que deblurGAN es el modelo más eficiente en términos computacionales, seguido de NAFNet y, finalmente, Restormer. Esta diferencia puede explicarse por la complejidad arquitectónica de Restormer, que, si bien proporciona mejores resultados métricos, requiere un mayor tiempo de cómputo por iteración.

Capítulo 6

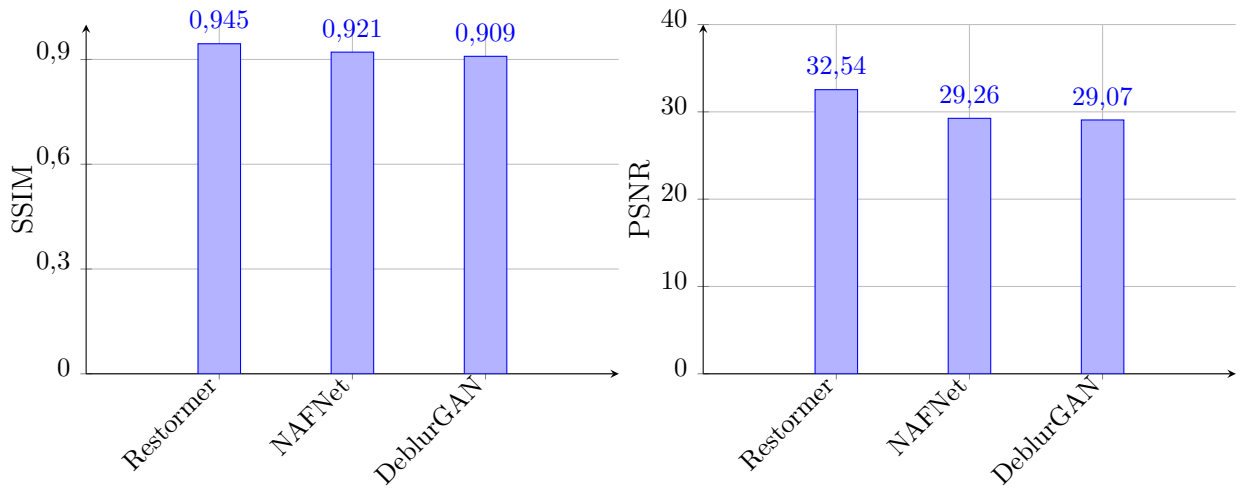
Evaluación

En este capítulo se muestra la evaluación de los modelos una vez están implementados, entrenados y validados. Por un lado mostraremos los resultados cuantitativos con las métricas de éxito consideradas, y por otro lado mostraremos los resultados cualitativos obtenidos al restaurar imágenes reales con los modelos.

6.1. Resultados de *test*

6.1.1. Métricas con el *dataset GoPro*

En la Figura 6.1 se recogen los resultados de las métricas SSIM y PSNR de cada uno de los modelos en el conjunto de *test* del *dataset GoPro*



(a) SSIM (b) PSNR

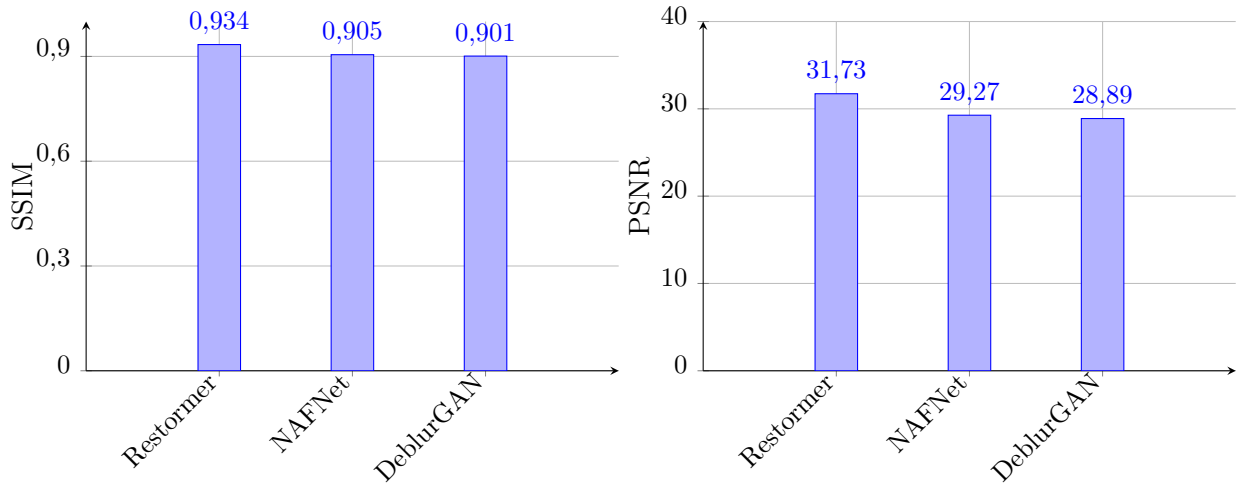
Figura 6.1: Comparación de métricas entre modelos usando el *dataset GoPro* (*test*)

Al igual que en el entrenamiento, al comparar los resultados obtenidos con los umbrales establecidos (PSNR $\geq$ 30 dB y SSIM $\geq$ 0.93), se observa que solo el modelo Restormer logra cumplir ambos criterios. En el caso de NAFNet y DeblurGAN, los valores de PSNR se sitúan

ligeramente por debajo del umbral, mientras que sus SSIM también se mantienen por debajo de 0.93, aunque en ambos casos los resultados son bastante cercanos.

6.1.2. Métricas con un *dataset* de imágenes industriales

En la Figura 6.2 se recogen los resultados de las métricas SSIM y PSNR de cada uno de los modelos en el conjunto de *test* del *dataset* de imágenes industriales considerado.



(a) SSIM (b) PSNR

Figura 6.2: Comparación de métricas entre modelos usando imágenes en entornos industriales

6.2. Resultados visuales

Una vez mostradas las métricas de éxito para cada uno de los modelos, mostremos cualitativamente la mejora de los tres modelos sobre una misma imagen. Veremos como estos resultados reafirman las métricas obtenidas anteriormente. En las Figuras 6.3 y 6.4 se muestra el resultado de aplicar los modelos Restormer, NAFNet y deblurGAN respectivamente a una misma imagen. La imagen original se sitúa en la primera fila, y las imágenes restauradas en la segunda. Observando estas imágenes podemos afirmar que el modelo Restormer obtiene un mejor resultado que los otros dos en la tarea de *deblurring*.

Imagen original (imagen de [47])

Restaurada por Restormer

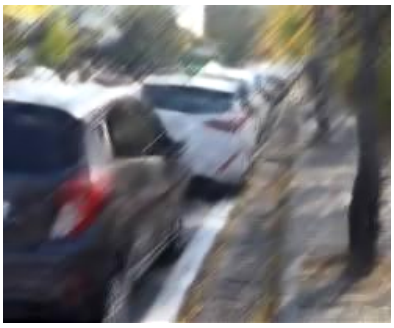
Restaurada por NAFNet

Restaurada por DeblurGAN

Figura 6.3: Primera comparación visual de mejora entre modelos

Imagen original (imagen de [40])

Restaurada por Restormer



Restaurada por NAFNet

Restaurada por DeblurGAN

Figura 6.4: Segunda comparación de mejora entre modelos

Capítulo 7

Herramienta de visualización

Para facilitar el acceso a los modelos entrenados para la tarea de *deblurring* se ha desarrollado la aplicación *Deepblurring*, una herramienta de visualización que permitirá a los usuarios cargar sus propias imágenes, elegir uno de los modelos, y obtener la imagen previamente cargada restaurada por el modelo escogido.

7.1. Análisis del problema

Esta sección se centra en la comprensión del problema que se aborda en el proyecto, analizándolo en base a los requisitos que lo caracterizan.

7.1.1. Alcance

En la Figura 1.2 se delimitó el alcance del proyecto, y en esta subsección se delimita el alcance de la solución desarrollada para el problema del *blur*. Es decir, la aplicación de mejora de imágenes a través de modelos *Deep Learning*, *Deepblurring*. El alcance de esta solución, al igual que el alcance del proyecto, se modela con tres niveles de profundidad. Dicho alcance se recoge visualmente en la Figura 7.1

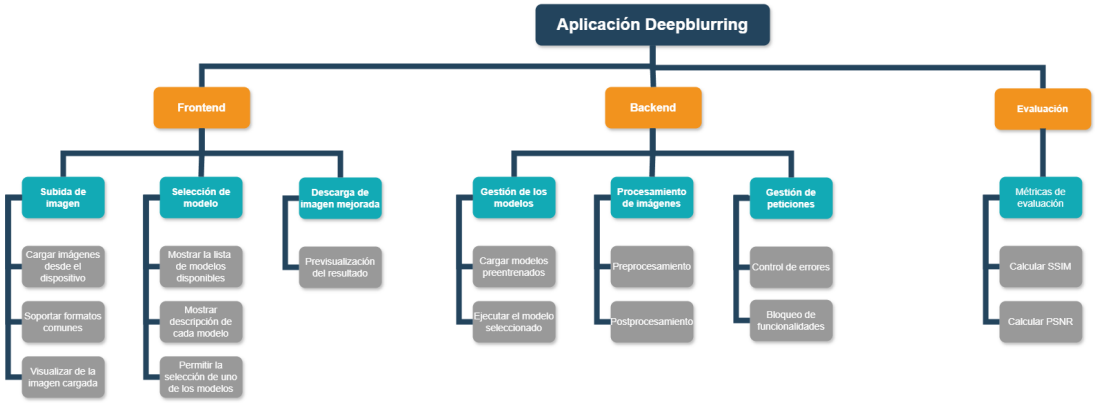


Figura 7.1: Alcance de la aplicación *Deepblurring*

El alcance de la aplicación se estructura en los siguientes bloques:

- **Modelar el *frontend*:** encargado de la interacción con el usuario. Incluye la subida de imágenes desde el dispositivo, la selección de un modelo de mejora entre los disponibles, la visualización previa del resultado y la descarga de la imagen mejorada.
    - *Limitaciones:* No se permite la edición avanzada de imágenes como recorte, zoom o rotación.
- **Gestionar el *backend*:** gestiona la lógica del procesamiento. Permite cargar y ejecutar modelos preentrenados, realizar el preprocesamiento y postprocesamiento de imágenes, así como gestionar peticiones, errores y restricciones de uso.
    - *Limitaciones:* No se contempla el entrenamiento desde cero de los modelos ni la subida de modelos personalizados por parte del usuario. El sistema está limitado a modelos previamente integrados.
- **Evaluar:** incorpora métricas objetivas para evaluar la calidad de las imágenes mejoradas. En concreto, se incluye el cálculo de las métricas SSIM y PSNR.
    - *Limitaciones:* La evaluación solo se realiza si el usuario proporciona una imagen original como referencia. No se incluye un sistema de autenticación ni un historial persistente de resultados o usuarios.

7.1.2. Stakeholders

En el contexto de la aplicación desarrollada para la mejora de imágenes mediante modelos de *Deep Learning*, se identifican los siguientes *stakeholders* principales, cada uno con intereses específicos en relación con el uso, funcionalidades y resultados ofrecidos por dicha herramienta:

- **Empresas industriales.** Organizaciones que podrían integrar o utilizar la aplicación como herramienta auxiliar para el análisis de imágenes obtenidas en sus procesos de producción.
  Intereses: Disponer de una solución ligera, accesible y funcional para evaluar visualmente la mejora de imágenes tomadas en planta, sin necesidad de desarrollar herramientas propias ni recurrir a terceros.
    - **Ingenieros y técnicos de mantenimiento.** Usuarios directos que emplean la aplicación para analizar imágenes capturadas en condiciones desfavorables (vibraciones, movimiento, desenfoque por foco).
      Intereses: Subir imágenes de maquinaria o componentes, aplicar un modelo de mejora, comparar resultados visuales y tomar decisiones informadas sobre mantenimiento preventivo o correctivo.
    - **Responsables de calidad y producción.** Aunque no interactúan directamente con la herramienta, pueden utilizar los resultados procesados para validar hallazgos o justificar decisiones operativas.
      Intereses: Obtener imágenes más claras para documentar anomalías, apoyar informes de calidad o mejorar la trazabilidad de incidencias visuales detectadas en la línea de producción.

- **Desarrolladores de soluciones basadas en IA.** Profesionales que exploran modelos de *deblurring* y buscan entornos de prueba accesibles para comparar rendimiento y resultados visuales.
  Intereses: Evaluar modelos preintegrados sin necesidad de montar entornos locales, realizar pruebas rápidas y observar los efectos visuales de cada arquitectura.
- **Clientes finales o usuarios del producto industrial.** No utilizan la aplicación directamente, pero se benefician de productos fabricados con mayor precisión visual gracias al uso de imágenes mejoradas.
  Intereses: Obtener productos más fiables, con menor tasa de defectos y mayor control de calidad derivado del uso de herramientas de mejora visual como la desarrollada.

7.1.3. Requisitos

En esta subsección se especifican los requisitos que se han identificado, diferenciándolos entre requisitos de usuario, funcionales, no funcionales y de información, que caracterizan el problema que aborda este proyecto.

Requisitos de usuario

Los requisitos de usuario describen las acciones o necesidades que el usuario espera poder realizar con el sistema.

- **RU-1.** El usuario debe poder subir imágenes desde su dispositivo.
- **RU-2.** El usuario debe poder seleccionar un modelo de mejora entre los disponibles.
- **RU-3.** El usuario debe poder aplicar la mejora a la imagen con el modelo seleccionado.
- **RU-4.** El usuario debe poder visualizar la imagen original y la mejorada.
- **RU-5.** El usuario debe poder descargar la imagen mejorada.

Se presenta en la Figura 7.2 el diagrama de casos de uso.

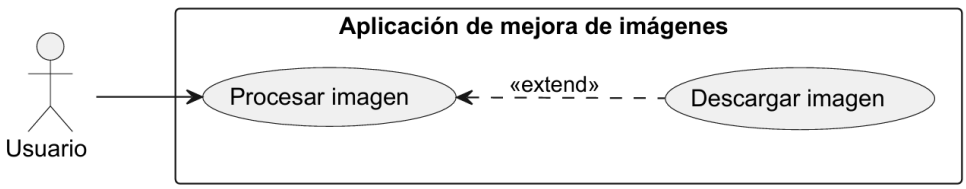


Figura 7.2: Diagrama de Casos de Uso

En las Tablas 7.1, 7.2, se recoge la especificación de los casos de uso. En la Tabla 7.3 se recoge la relación entre los requisitos de usuario y los casos de uso.

CU-1	**Procesar imagen**
Descripción	El usuario carga una imagen, selecciona un modelo y lanza el procesamiento. El sistema genera y muestra el resultado junto con las métricas de mejora.
Precondiciones	El usuario debe tener una imagen válida (JPG, JPEG o PNG), o un ZIP que contenga imágenes válidas.
Secuencia Normal	Paso 1 – El usuario pulsa en el botón para subir imagen. Paso 2 – El usuario selecciona una imagen o ZIP. Paso 3 – El sistema valida el archivo. Paso 4 – El usuario pulsa "Seleccionar modelo". Paso 5 – El sistema muestra los modelos disponibles. Paso 6 – El usuario selecciona uno. Paso 7 – El usuario pulsa "Aplicar modelo". Paso 8 – El sistema procesa la imagen o imágenes. Paso 9 – Se muestran las imágenes originales y mejoradas junto con sus métricas (SSIM y PSNR).
Postcondiciones	Se generan y visualizan las imágenes mejoradas. El sistema habilita la descarga.
Excepciones	▪ Imagen inválida (formato incorrecto). ▪ No hay modelos disponibles. ▪ Fallo en el procesamiento. ▪ Error al mostrar las imágenes.

Tabla 7.1: CU-1: Procesar imagen

CU-2	Descargar imagen mejorada
Descripción	El usuario puede descargar las imágenes mejoradas generadas por el sistema tras el procesamiento.
Precondiciones	Haber procesado correctamente al menos una imagen.
Secuencia Normal	Paso 1 – El usuario pulsa el botón "Descargar". Paso 2 – El sistema genera el archivo y lo transfiere al dispositivo del usuario.
Postcondiciones	El archivo con las imágenes mejoradas queda guardado localmente en el dispositivo del usuario.
Excepciones	▪ Fallo al generar el archivo. ▪ Error durante la descarga.

Tabla 7.2: CU-2: Descargar imagen mejorada

Caso de Uso	Requisitos de Usuario asociados
CU-1	RU-1, RU-2, RU-3, RU-4
CU-2	RU-5

Tabla 7.3: Trazabilidad entre requisitos de usuario y casos de uso

Requisitos funcionales y de información

Los requisitos funcionales especifican los comportamientos concretos que el sistema debe llevar a cabo para dar soporte a esos requisitos de usuario, mientras que los requisitos de información detallan los datos que el sistema necesita manejar o presentar para cumplir con dichas funciones. En la Tabla 7.4 se recogen los requisitos funcionales y de información asociados a los requisitos de usuario. Cabe mencionar que la redacción más rigurosa para los requisitos funcionales debería comenzar por expresiones como "el sistema deberá". Análogamente, los requisitos de información deberían contener en su redacción expresiones tales como "el sistema dispondrá de los datos", "la siguiente información estará disponible", o "se utilizarán los siguientes datos" entre otras expresiones. Por cuestiones de espacio, en la Tabla 7.4 se condensará dicha redacción.

Requisito de Usuario	Requisitos Funcionales Asociados	Requisitos de Información Asociados
RU-1. Subir imagen desde el dispositivo	▪ **RF-1.1** Validar tipo de archivo (JPG, PNG) ▪ **RF-1.2** Guardar la imagen temporalmente	▪ **RI-1.** Tipos de archivo válidos ▪ **RI-2.** Tamaño máximo permitido
RU-2. Seleccionar modelo de mejora	▪ **RF-2.1** Mostrar lista de modelos disponibles	▪ **RI-3.** Lista de modelos con nombre y descripción
RU-3. Aplicar mejora de imagen	▪ **RF-3.1** Procesar la imagen con el modelo seleccionado ▪ **RF-3.2** Gestionar errores durante el proceso	▪ **RI-4.** Imagen original ▪ **RI-5.** Modelo aplicado
RU-4. Visualizar imagen original y mejorada	▪ **RF-4.1** Mostrar la imagen original cargada ▪ **RF-4.2** Mostrar la imagen mejorada tras el procesamiento	▪ **RI-6.** Imagen de entrada ▪ **RI-7.** Imagen de salida
RU-5. Descargar imagen mejorada	▪ **RF-5.1** Ofrecer botón para descarga	▪ **RI-8.** Imagen final procesada

Tabla 7.4: Relación entre requisitos de usuario, funcionales y de información

Requisitos no funcionales

Los requisitos no funcionales definen las cualidades o condiciones que debe cumplir el sistema, como la usabilidad, rendimiento, seguridad o disponibilidad, y no hacen referencia directa a funcionalidades concretas.

Categoría	Requisito No Funcional (RNF)
Rendimiento	**RNF-1.** El sistema debe procesar y devolver una imagen mejorada en menos de 10 minutos para imágenes de hasta 2500x2500 píxeles.
Usabilidad	**RNF-2.** La interfaz debe ser clara e intuitiva, permitiendo completar el flujo de mejora de imagen en menos de 5 clics.
Compatibilidad	**RNF-3.** La aplicación debe funcionar correctamente en los navegadores más comunes (Chrome, Firefox, Edge).
Fiabilidad	**RNF-4.** El sistema debe gestionar errores de entrada (formato, tamaño) sin interrumpir la experiencia del usuario.
Mantenibilidad	**RNF-5.** El sistema debe permitir añadir nuevos modelos de mejora sin necesidad de reestructurar la arquitectura existente.

Tabla 7.5: Requisitos no funcionales

7.2. Diseño de la herramienta

7.2.1. Arquitectura

En esta subsección se describe la arquitectura de la solución desarrollada desde dos perspectivas complementarias: la lógica y la física. La arquitectura lógica hace referencia a los módulos funcionales y su interacción, mientras que la arquitectura física detalla la disposición de dichos componentes sobre los recursos del sistema.

Arquitectura lógica

La arquitectura lógica, representada en la Figura 7.3, describe la organización interna del sistema en términos de módulos funcionales.

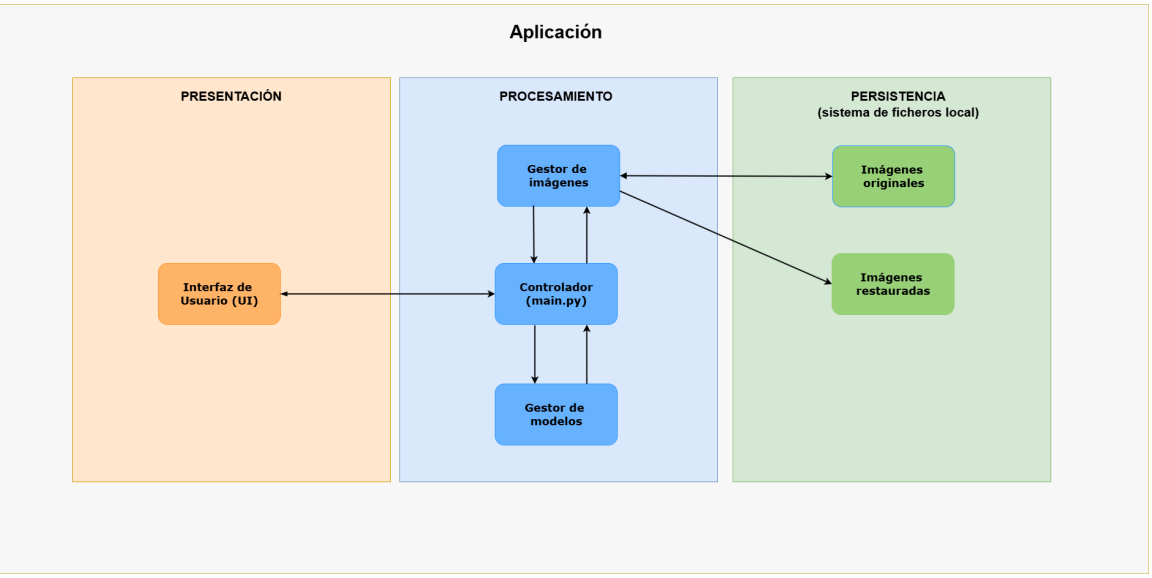


Figura 7.3: Arquitectura lógica

En este caso, la solución se organiza como una aplicación monolítica que se ejecuta localmente. Su estructura incluye las siguientes tres componentes:

- **Presentación**: Esta componente incluye la interfaz con la que el usuario se comunica con la aplicación tanto para cargar sus imágenes y seleccionar el modelo deseado, como para visualizar las imágenes restauradas.
- **Procesamiento**: Esta componente incluye un controlador, que actúa de nexo entre los diferentes elementos, y que además se comunica con la interfaz de usuario. Estos elementos son dos: el gestor de imágenes, que recibe las imágenes del usuario y devuelve las imágenes restauradas, almacenando todas ellas en el sistema de ficheros de la aplicación. Y el gestor de modelos, que recibe la imagen a mejorar, y qué modelo ha sido seleccionado y se encarga de realizar la inferencia sobre la imagen y devolver el resultado al controlador.
- **Persistencia**: Esta componente incluye el sistema de ficheros local, que almacena las imágenes cargadas por el usuario y sus versiones mejoradas. Este almacenamiento evoluciona dinámicamente: se crean nuevos archivos a medida que el usuario sube imágenes y se guardan los resultados de los modelos.

Desde una perspectiva estructural, las componentes están dentro de la aplicación pero desacopladas. El controlador actúa como orquestador: canaliza las acciones del usuario hacia el sistema de archivos y los modelos, y gestiona la interfaz. Desde una perspectiva funcional, la solución ofrece un flujo completo que permite al usuario subir una imagen, seleccionar un modelo, aplicar el proceso de mejora y descargar el resultado.

Arquitectura física

La arquitectura física, representada en la Figura 7.4, describe cómo se despliega y ejecuta la solución sobre la infraestructura física.

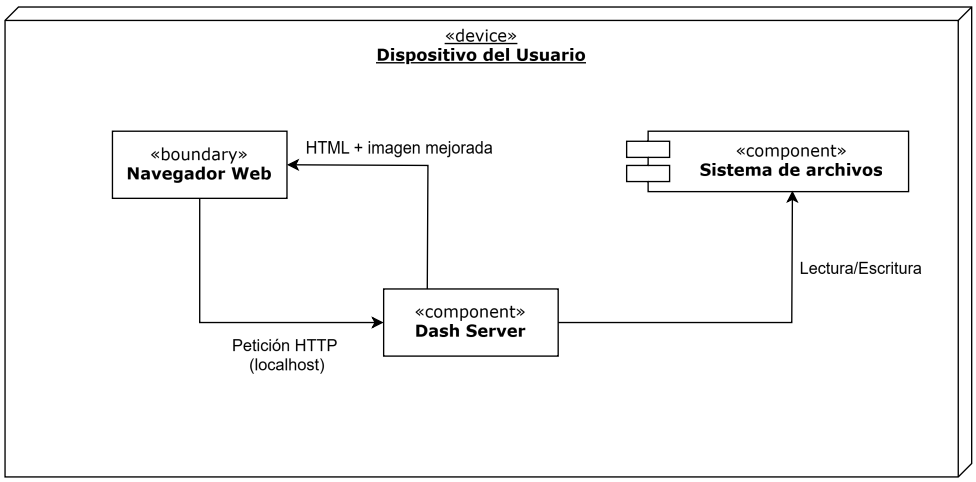


Figura 7.4: Arquitectura física

En este caso, la solución funciona de manera completamente local, sin necesidad de acceso a servidores externos ni conexión a bases de datos remotas. El entorno de ejecución incluye los siguientes elementos:

- **Navegador web**: interfaz mediante la cual el usuario interactúa con la aplicación (a través de localhost:8050). Envía las solicitudes al servidor y muestra las respuestas visuales.
- **Servidor local**: instancia lanzada desde `main.py` que ejecuta la lógica de la aplicación, usando la librería Dash. Procesa todas las peticiones entrantes, ejecuta los modelos seleccionados, accede al sistema de archivos y construye las respuestas HTML con las imágenes resultantes.
- **Sistema de archivos local**: alberga los directorios `/assets`, `/imagenes`, `/modelos`, y el propio `main.py`. Funciona como unidad de almacenamiento persistente para imágenes y modelos.

Desde el punto de vista dinámico, la interacción funcional se inicia cuando el usuario abre la interfaz desde su navegador. El navegador envía una solicitud al servidor local, que responde con la interfaz gráfica. A través de esta interfaz, el usuario puede subir una imagen, seleccionar un modelo y solicitar la mejora. Cada acción genera una nueva interacción entre el navegador y el servidor. El servidor, a su vez, accede al sistema de archivos para leer la imagen, cargar el modelo adecuado y guardar la imagen procesada, que finalmente es devuelta al usuario para su descarga o visualización.

Estructura lógica de almacenamiento

La solución no utiliza bases de datos relacionales ni sistemas de almacenamiento persistente complejos, dado que su ejecución es completamente local y orientada a procesamiento de imágenes

de forma puntual. Por tanto, la estructura lógica de almacenamiento se basa en el sistema de archivos del dispositivo donde se ejecuta la aplicación. Los datos se organizan en una serie de directorios que cumplen funciones específicas:

- **imagenes/entrada/**: almacena temporalmente las imágenes cargadas por el usuario a través de la interfaz. Cada imagen es registrada con identificadores únicos para evitar colisiones.
- **imagenes/salida/**: contiene las imágenes resultantes tras el procesamiento con el modelo seleccionado. Estas imágenes se generan dinámicamente y están disponibles para su visualización y descarga.
- **modelos/**: contiene los directorios correspondientes a los modelos de *Deep Learning* disponibles. Estos directorios no se modifican en tiempo de ejecución, sino que se encuentran en el dispositivo del usuario y cargan según la elección del mismo.
- **assets/**: aunque no almacena datos del usuario, se incluye como parte del almacenamiento lógico, ya que contiene los recursos visuales necesarios para renderizar correctamente la interfaz.

Los datos se almacenan de forma estructurada en carpetas según su tipo y propósito: entrada del usuario, salida procesada, modelos aplicables y recursos visuales. Este enfoque permite una gestión sencilla y trazable de los datos durante la ejecución del sistema, facilitando además su limpieza o reutilización posterior.

7.2.2. Prototipo de la interfaz de usuario

La Figura 7.5 muestra un prototipo de la interfaz de usuario diseñada para la aplicación. Esta interfaz está orientada a la simplicidad de uso, permitiendo completar el proceso de mejora de una imagen en pocos pasos. El usuario puede cargar una imagen desde su dispositivo, seleccionar uno de los modelos de *Deep Learning* disponibles, aplicar el procesamiento y visualizar o descargar el resultado desde la misma pantalla.

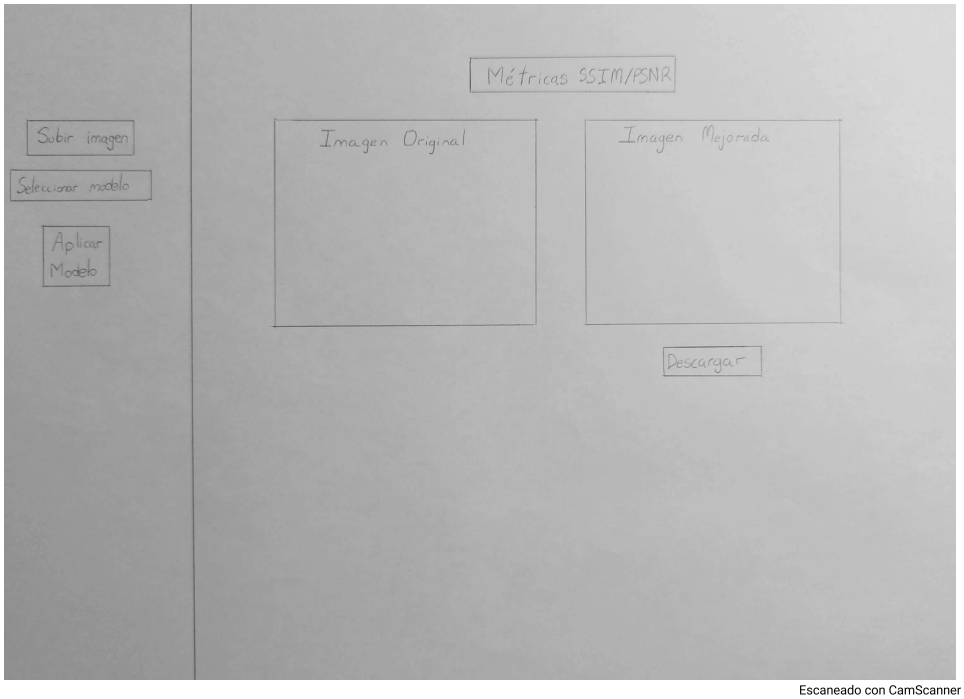


Figura 7.5: Prototipo de la interfaz de usuario

7.3. Desarrollo de la herramienta

El desarrollo de la aplicación se ha realizado utilizando el framework *Dash*, que permite construir interfaces web interactivas en *Python* sin necesidad de utilizar *JavaScript*. Esta elección responde a la necesidad de integrar directamente los modelos de *Deep Learning* entrenados previamente y facilitar su uso a través de una interfaz gráfica intuitiva. A lo largo del desarrollo se han tomado diversas decisiones orientadas a mejorar la experiencia de usuario y garantizar la robustez del sistema. Uno de los principales problemas encontrados fue la gestión eficiente de múltiples imágenes y sus versiones procesadas, que se resolvió mediante una estructura de carpetas dinámicas. También se detectaron conflictos entre múltiples callbacks de *Dash*, especialmente en la gestión de eventos duplicados, lo cual se solucionó reorganizando algunos callbacks para que compartieran salidas comunes. Por otro lado, se presentaron desafíos de compatibilidad entre los modelos y la lógica del servidor, dado que cada modelo requería una forma distinta de invocación y tratamiento de imágenes. Para unificar este proceso, se definió una interfaz común que encapsula las llamadas a los distintos modelos según la opción seleccionada por el usuario. En la Figura 7.6 se muestra la vista principal del usuario al entrar a la aplicación.

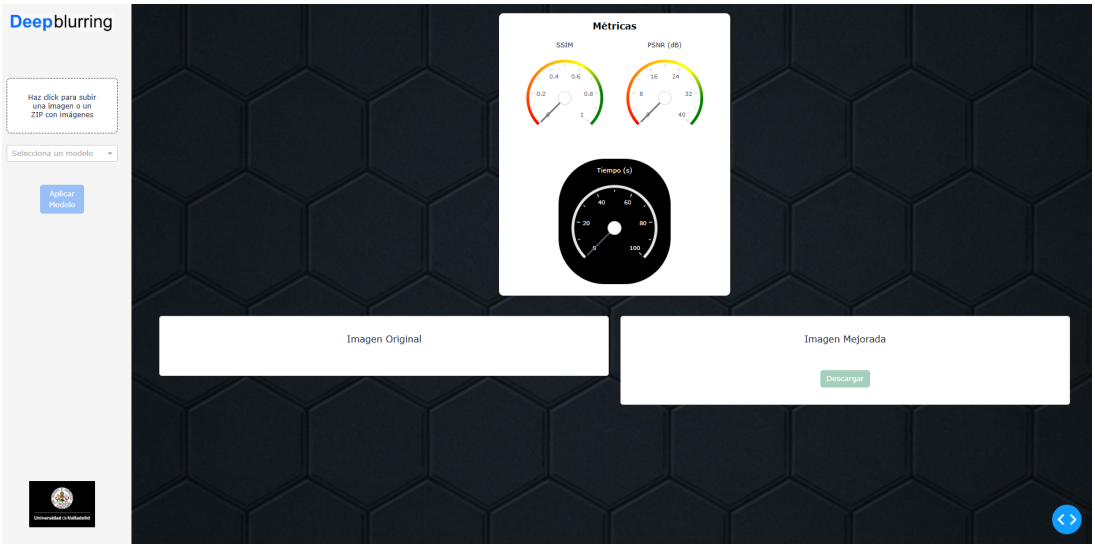


Figura 7.6: Aplicación *Deepblurring*

7.3.1. Casos de prueba

Además de los modelos de restauración, se diseñaron casos de prueba específicos para validar la aplicación desarrollada, asegurando su correcto funcionamiento, robustez ante errores y adecuación a los requisitos funcionales. Estas pruebas se orientaron mayoritariamente desde una perspectiva de caja negra, evaluando la respuesta del sistema frente a distintas entradas y acciones del usuario, sin acceder al código interno. Sin embargo, también se realizaron algunas pruebas de caja blanca para validar el comportamiento esperado de los callbacks y del flujo de datos entre los distintos componentes. A continuación se describen los principales casos de prueba:

- **Prueba 1: Carga de un único archivo de imagen válido**

  Entrada: Imagen de resolución máxima 1024×1024 píxeles.
  Resultado esperado: Se muestra la imagen original en el panel izquierdo y el botón "Aplicar Modelo" queda habilitado si se ha seleccionado previamente un modelo.
  Criterio de aceptación: La imagen debe cargarse correctamente y visualizarse sin errores ni distorsiones.

- **Prueba 2: Carga de archivo ZIP con múltiples imágenes**

  Entrada: Archivo ZIP con 5 imágenes PNG.
  Resultado esperado: Se muestra la primera imagen, se activan las flechas de navegación, y se inicializa la sesión correctamente.
  Criterio de aceptación: El usuario puede navegar entre imágenes y aplicar el modelo seleccionado a todas ellas.

- **Prueba 3: Procesamiento sin modelo seleccionado**

  Entrada: Imagen válida, pero sin seleccionar ningún modelo.
  Resultado esperado: El botón de procesado permanece deshabilitado.
  Criterio de aceptación: La interfaz debe evitar cualquier acción si no se han definido todos los parámetros necesarios.

- **Prueba 4: Aplicación del modelo y visualización de resultados**

  Entrada: Imagen cargada y modelo Restormer seleccionado.
  Resultado esperado: Se muestra la imagen mejorada, junto con métricas de PSNR y SSIM.
  Criterio de aceptación: Los resultados deben aparecer en los paneles correspondientes, sin errores de codificación ni fallos en la visualización.

- **Prueba 5: Descarga de la imagen mejorada**

  Entrada: Imagen procesada con éxito.
  Resultado esperado: Al hacer clic en el botón de descarga, el navegador descarga la imagen mejorada con el nombre correspondiente.
  Criterio de aceptación: El archivo descargado debe ser válido, tener el formato correcto y coincidir con la imagen mostrada.

- **Prueba 6: Manejo de errores en la navegación**

  Entrada: Carga de archivo ZIP, selección de modelo, y clics repetidos en los botones de navegación antes del procesamiento.
  Resultado esperado: La aplicación no muestra imágenes procesadas hasta que se haya aplicado el modelo.
  Criterio de aceptación: Se garantiza que las imágenes mejoradas no se soliciten si no existen, evitando errores de lectura o referencias inexistentes.

Estas pruebas han permitido validar la correcta interacción entre los componentes visuales y la lógica de servidor, confirmando que el sistema es funcional, tolerante a errores de usuario, y proporciona respuestas coherentes y claras en distintos escenarios de uso.

Ejecución de las pruebas

Las pruebas funcionales descritas en la subsección anterior fueron ejecutadas de forma manual sobre la aplicación completa, en su entorno final de despliegue local. Se comprobó que el comportamiento de la interfaz de usuario, los flujos de datos internos y las respuestas del sistema ante las acciones esperadas cumplen correctamente con los requisitos establecidos. Todas las pruebas fueron superadas con éxito, sin detectarse errores funcionales ni bloqueantes. En los casos límite definidos, como la carga de archivos no válidos o el intento de procesar sin seleccionar modelo, la aplicación respondió de forma robusta, mostrando el comportamiento esperado o evitando acciones incorrectas mediante la desactivación de botones o elementos interactivos. Con el objetivo de no extender innecesariamente la memoria, no se incluyen capturas de pantalla en esta sección. No obstante, para facilitar la comprensión del uso del sistema por parte del lector, se adjunta en el Apéndice B un Manual de Usuario que describe el funcionamiento completo de la aplicación paso a paso, desde la carga de imágenes hasta la visualización y descarga de resultados.

En la Figura 7.7 se muestra una captura de pantalla de la aplicación tras haber procesado una imagen borrosa procedente de un libro de texto. Se puede apreciar como el texto original es difícil de leer, pero tras procesar dicha imagen en la aplicación, se obtiene un texto completamente legible.

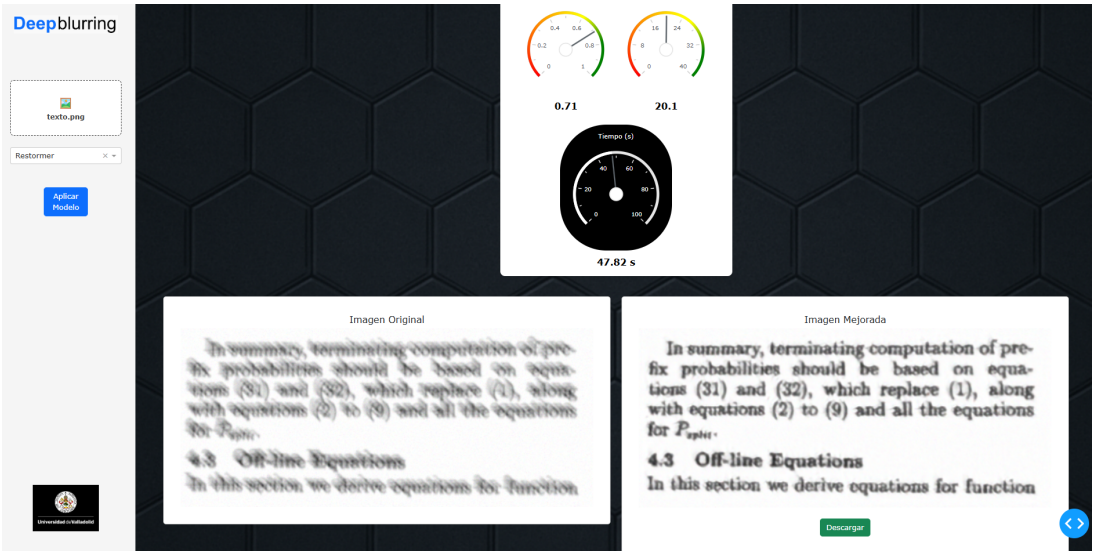


Figura 7.7: Prueba de la aplicación *Deepblurring*

Parte III

Conclusiones

Capítulo 8

Conclusiones y trabajo futuro

8.1. Conclusiones

Con una visión crítica, se lleva a cabo un balance general del Trabajo Fin de Grado. Se analiza si se han alcanzado o no los distintos objetivos planteados junto con un análisis de los riesgos a *posteriori*, al tiempo que se reflexiona sobre la idoneidad de la metodología empleada. Asimismo, se incorpora una valoración personal sobre la experiencia de haber desarrollado este proyecto.

8.1.1. Perspectiva del proyecto

Objetivos

A lo largo del desarrollo de este Trabajo Fin de Grado se han abordado los objetivos definidos inicialmente (ver Sección 1.2), los cuales se detallan a continuación junto con las principales conclusiones extraídas:

- **OBJ-01.** Se ha realizado un estudio detallado del fenómeno del desenfoque en imágenes, abordando sus principales causas, sus efectos sobre la percepción visual y las dificultades que introduce en tareas automatizadas de inspección industrial. Además, se ha analizado la tipología más habitual de desenfoque en contextos reales.
- **OBJ-02.** Se han explorado las principales arquitecturas basadas en *Deep Learning* utilizadas para tareas de *deblurring*. En concreto, se han seleccionado y analizado tres arquitecturas: *NAFNet*, *Restormer* y *DeblurGAN*. Para cada una de ellas, se han revisado sus fundamentos, sus estructuras internas y sus ventajas e inconvenientes frente a distintos tipos de desenfoque.
- **OBJ-03.** Se ha implementado una aplicación funcional capaz de llevar a cabo la tarea de *deblurring* mediante los modelos *Deep Learning* seleccionados. Gracias a esta aplicación y a la investigación previa de las arquitecturas consideradas, se han podido aplicar diferentes modelos de *Deep Learning* sobre imagenes en entornos industriales reales. En particular, sobre imágenes de piezas industriales.
- **OBJ-04.** Los modelos han sido entrenados con datasets genéricos como *GoPro* y posteriormente evaluados con imágenes industriales. Esta evaluación ha permitido comparar su capacidad para mejorar la calidad visual de las imágenes en escenarios reales. Como resultado, se ha concluido que *Restormer* es el modelo más eficaz de los tres, mostrando un mejor rendimiento

cuantitativo (en métricas como *SSIM* y *PSNR*) y cualitativo en la restauración de bordes, texturas y detalles. Por lo tanto, se concluye este trabajo afirmando la aplicabilidad de estos modelos de *Deep Learning* para realizar la tarea de *deblurring* tanto con imágenes genéricas, como con imágenes de entornos industriales.

Tras este análisis, afirmamos que se han cumplido los objetivos específicos establecidos inicialmente, y con ello el objetivo general del proyecto.

Análisis de Riesgos

Durante la planificación inicial del proyecto se identificaron varios riesgos potenciales que podrían afectar al desarrollo o a los resultados del mismo (ver Sección 2.4). A continuación, se presenta un análisis de dichos riesgos junto con su evaluación final y el impacto real observado durante la ejecución del trabajo:

- **RI-01.** *El proyecto se retrasa con respecto a la planificación inicial.* Si bien algunas tareas sufrieron ligeros retrasos, estos estaban contemplados en la planificación inicial como márgenes de seguridad.

- **RI-02.** *Se posee insuficiente experiencia con las herramientas técnicas utilizadas.* Este riesgo se materializó parcialmente, especialmente en relación con el uso de plataformas como *Google Colab* para el entrenamiento de modelos de *Deep Learning*, que no se había utilizado previamente. No obstante, se dedicó un período específico de aprendizaje a estas herramientas, lo que permitió solventar la inexperiencia sin comprometer los objetivos del proyecto.

- **RI-03.** *Las técnicas de deblurring no consiguen su objetivo completamente.* Este riesgo no se llegó a materializar. Los modelos seleccionados y entrenados fueron capaces de mejorar significativamente la calidad de las imágenes, cumpliendo con el objetivo principal del proyecto en términos de nitidez y restauración visual.

- **RI-04.** *La GPU utilizada no es lo suficientemente potente.* Este riesgo sí se materializó, dado que la GPU inicialmente disponible resultó insuficiente para el entrenamiento eficiente de los modelos. Sin embargo, gracias a una planificación económica previsora, se había contemplado la posibilidad de necesitar recursos adicionales. Se contó con una GPU proporcionada por el Departamento de Informática, lo que permitió continuar sin afectar negativamente a los objetivos, aunque esto requirió algunas horas de trabajo más que las que se habían planificado.

- **RI-05.** *Posible dificultad en la replicabilidad de los resultados debido a la variabilidad en los modelos de Deep Learning y los datasets.* Aunque este riesgo estaba presente, se logró mitigarlo de forma eficaz. A pesar de entrenar los modelos con el dataset genérico *GoPro*, se evaluaron posteriormente con imágenes industriales completamente distintas, lo que permitió comprobar su comportamiento en condiciones no vistas y garantizar una cierta capacidad de replicabilidad y generalización.

Metodología de trabajo

La metodología ASAP ha sido una pieza indispensable para la realización de este proyecto. Ha permitido mantener un ritmo de trabajo constante y con continua retroalimentación por parte de los tutores y de la comunidad, lo que ha ido mejorando el incremento *sprint* a *sprint*.

8.1.2. Perspectiva personal

Este Trabajo Fin de Grado me ha permitido tomar conciencia de un problema técnico con gran presencia en entornos industriales y reales: el desenfoque en imágenes y sus consecuencias sobre procesos automáticos como la inspección visual o la extracción de información. Gracias al desarrollo de esta solución, he comprendido tanto la complejidad del problema como el potencial que ofrecen las técnicas basadas en *Deep Learning* para abordarlo.

Durante el proyecto, he trabajado con arquitecturas avanzadas de *Deep Learning* como *Restormer*, *NAFNet* y *DeblurGAN*, lo que me ha permitido ampliar notablemente mis conocimientos en este campo. Empezando desde una base limitada, he aprendido a entrenar, adaptar y aplicar modelos de *Deep Learning* sobre conjuntos de datos reales y sintéticos, adquiriendo experiencia práctica con herramientas como *PyTorch*, *Google Colab* y *Dash* para construir tanto el núcleo de procesamiento como la interfaz de usuario de la aplicación. Además, este trabajo me ha ayudado a familiarizarme con conceptos clave como la evaluación cuantitativa mediante métricas como *SSIM* y *PSNR*, el preprocesamiento de datos, la organización modular del código y la integración de modelos entrenados en una aplicación funcional.

Dado que este es el primer proyecto real del que soy responsable, el trabajo también me ha servido como punto de partida para mi futuro desarrollo profesional. He aprendido a gestionar un proyecto de principio a fin, a trabajar con bibliografía técnica y científica, y a redactar una memoria estructurada que sintetice el trabajo de forma rigurosa y fundamentada.

8.2. Trabajo futuro

Las tres arquitecturas consideradas en este proyecto pertenecen a la rama de Aprendizaje Supervisado (ver Subsección 3.2.2). Sin embargo, es posible buscar soluciones al problema del *blur* empleando Aprendizaje No Supervisado. Por ello, separamos la propuesta de líneas futuras de trabajo en función del tipo de aprendizaje que requieren los modelos que realizarán la tarea de *deblurring*.

Propuestas sobre los modelos supervisados

Con base en las limitaciones detectadas en la Subsección 5.2, se proponen diversas estrategias para mejorar el rendimiento de la solución.

1. En primer lugar, se considera la posibilidad de combinar modelos especializados en distintos tipos de desenfoque, permitiendo así una aproximación más robusta a situaciones complejas o ambiguas. Este enfoque de ensamblado o selección dinámica de modelos podría mejorar la capacidad de generalización del sistema.

2. Además, se propone ajustar y ampliar el conjunto de datos utilizado para el entrenamiento, incorporando ejemplos más variados y realistas, incluyendo desenfoques múltiples, diferentes condiciones de iluminación y ruido. Esta diversificación del entrenamiento puede ayudar a mejorar su capacidad de adaptación a entornos industriales reales, que suelen presentar características muy distintas a los datasets genéricos utilizados en el ámbito académico.

3. Otro área de mejora sería la de los hiperparámetros del entrenamiento (como la tasa de aprendizaje o el número de iteraciones), junto con un refinamiento de las estrategias de regularización para evitar sobreajuste o convergencias prematuras. Como ideas concretas de mejora, se proponen:

    - Incrementar el número de iteraciones de entrenamiento.
    - Aumentar el tamaño del *batch*, si la memoria disponible lo permite, lo que puede facilitar una estimación más estable del gradiente.

Exploración de modelos no supervisados

Una línea de trabajo especialmente prometedora consiste en explorar modelos de *Deep Learning* no supervisados para la tarea de *deblurring*, que no requieren imágenes nítidas de referencia. Esto solventaría una de las principales limitaciones de los modelos actuales, ya que en entornos industriales no siempre se dispone de pares de imágenes borrosas/nítidas. Algunos modelos no supervisados relevantes para esta tarea son:

- **Deep Image Prior**: ha demostrado capacidad para restaurar imágenes sin necesidad de entrenamiento con imágenes nítidas, aprovechando la estructura del propio modelo [53]. Para restaurar una imagen borrosa comienza creando una imagen con ruido aleatorio. La red va ajustando dicho ruido iterativamente para que logre recuperar detalles coherentes de la imagen borrosa introducida tales como líneas, bordes o formas geométricas. Se ha probado este modelo sobre las mismas imágenes usadas para evaluar los modelos supervisados, con resultados satisfactorios. Pues, a pesar de que este modelo no necesita entrenar con las imágenes nítidas para restaurar las imágenes borrosas, consigue llevar a cabo la tarea de *deblurring* con éxito. Dado que no entra dentro del alcance de este proyecto, no se han reflejado en esta memoria los resultados de la evaluación de este modelo, pero se propone como línea futura de investigación. Su repositorio en GitHub puede consultarse en [52].

- **SelfDeblur**: propone un mecanismo auto-regresivo que aprende a restaurar imágenes desenfocadas sin supervisión explícita [39]. Su repositorio en GitHub puede consultarse en [62].

- **CycleGAN**: se ha utilizado para traducción de dominios sin pares, y podría aplicarse a la tarea de *deblurring* entrenando sobre conjuntos de imágenes desenfocadas y nítidas sin necesidad de correspondencia directa [66]. Su repositorio en GitHub puede consultarse en [65].

El estudio de estos enfoques junto con una evaluación comparativa frente a los modelos supervisados, constituye una dirección interesante para futuros proyectos o ampliaciones de este.

Parte IV

Apéndices

Apéndice A

Glosario

- **Adversarial Loss:** Función de pérdida utilizada en redes generativas adversariales (GANs) que permite mejorar la calidad de imágenes generadas, optimizando la similitud con imágenes reales.
- **Algoritmos de clasificación:** Métodos de aprendizaje supervisado que asignan una categoría a cada instancia de datos dentro de un conjunto de clases predefinido.
- **Algoritmos de clustering:** Métodos de aprendizaje no supervisado que agrupan datos similares en conjuntos llamados clústeres sin necesidad de etiquetas previas.
- **Algoritmos de regresión:** Métodos de aprendizaje supervisado que predicen valores numéricos continuos basándose en patrones extraídos de datos históricos.
- **Aprendizaje automático (Machine Learning):** Rama de la inteligencia artificial que desarrolla algoritmos capaces de aprender patrones a partir de datos y hacer predicciones sin ser programados explícitamente.
- **Aprendizaje multietiqueta:** Enfoque de clasificación en el que una instancia de datos puede pertenecer a múltiples categorías simultáneamente.
- **Aprendizaje no balanceado:** Problema en el que el conjunto de datos de entrenamiento presenta una distribución desigual entre las clases de interés, lo que puede afectar el rendimiento de los modelos.
- **Aprendizaje no supervisado:** Tipo de aprendizaje automático en el que un modelo analiza datos no etiquetados para descubrir patrones y estructuras sin intervención humana.
- **Aprendizaje por refuerzo:** Paradigma de aprendizaje en el que un agente toma decisiones en un entorno para maximizar una recompensa acumulativa basada en su comportamiento.
- **Aprendizaje por transferencia:** Técnica en la que un modelo previamente entrenado en una tarea se reutiliza y adapta a un nuevo problema con datos similares.
- **Aprendizaje semisupervisado:** Método de aprendizaje que combina datos etiquetados y no etiquetados para mejorar la precisión del modelo cuando la disponibilidad de etiquetas es limitada.

- **Red *encoder-decoder*:** Redes neuronales utilizadas para aprender representaciones comprimidas de los datos, empleadas en tareas de restauración de imágenes como la eliminación del ruido o el *deblurring*.
- **Blur (Desenfoque):** Degradación de una imagen causada por factores como movimiento de la cámara, vibraciones o falta de enfoque, reduciendo la nitidez y los detalles visuales.
- **Character Error Rate (CER):** Métrica utilizada en imágenes con contenido textual para evaluar la precisión de la restauración midiendo los errores de carácter.
- **Convolución:** Operación matemática fundamental en el procesamiento de imágenes y en redes neuronales convolucionales. Consiste en aplicar un filtro (o *kernel*) sobre una imagen para extraer características locales, como bordes, texturas o patrones, mediante el desplazamiento del filtro a lo largo de la imagen y el cálculo de productos escalares en cada posición.
- **Deep Learning (Aprendizaje Profundo):** Rama del aprendizaje automático basada en redes neuronales profundas con múltiples capas que extraen características complejas de los datos.
- **Deblurring:** Técnica de restauración de imágenes que busca eliminar el desenfoque (*blur*) para recuperar detalles y mejorar la nitidez de una imagen degradada.
- **Deblurring ciego (Blind Deblurring):** Proceso de eliminación del desenfoque cuando el *kernel* de desenfoque es desconocido y debe ser estimado junto con la imagen restaurada.
- **Deblurring no ciego (Non-Blind Deblurring):** Método de restauración de imágenes en el que el *kernel* de desenfoque es conocido y se utiliza para reconstruir la imagen nítida.
- **Deconvolución (Deconvolution):** Proceso matemático que busca invertir el efecto de la convolución.
- **Descenso del gradiente:** Algoritmo de optimización utilizado para minimizar funciones de pérdida en modelos de aprendizaje automático ajustando los pesos de la red neuronal.
- **Error Ratio (ER):** Métrica utilizada en visión por computadora para evaluar la proporción de error en imágenes restauradas.
- **Función de activación:** Función matemática que introduce no linealidad en una red neuronal, permitiendo modelar relaciones complejas entre las entradas y salidas.
- **Función de pérdida (Loss Function):** Función matemática que mide la diferencia entre la imagen generada por el modelo y la imagen real, guiando el entrenamiento de la red neuronal.
- **GAN (Red Generativa Adversarial):** Modelo de aprendizaje profundo compuesto por dos redes neuronales (generador y discriminador) que compiten para generar imágenes sintéticas realistas.
- **GoPro dataset:** Conjunto de datos ampliamente utilizado para entrenar modelos de *deblurring*, que contiene pares de imágenes borrosas y nítidas generadas con cámaras GoPro.

- **Industria 4.0:** Concepto que define la cuarta revolución industrial basada en la digitalización, la automatización, la inteligencia artificial y la interconexión de dispositivos en la manufactura y la producción.
- **Kernel de desenfoque (Blur Kernel):** Función matemática que modela el desenfoque aplicado a una imagen, representando cómo los píxeles de la imagen nítida han sido distribuidos en la imagen borrosa.
- **Mean Squared Error (MSE):** Métrica de evaluación que mide la diferencia promedio al cuadrado entre los valores predichos y los valores reales de una imagen.
- **Mecanismo de atención (Attention Mechanism):** Técnica utilizada en redes neuronales que permite enfocar el aprendizaje en regiones específicas de una imagen, mejorando la calidad de la restauración en áreas de interés.
- **Multi-Scale Networks (Redes Multi-Escala):** Arquitecturas de redes neuronales que procesan una imagen en diferentes escalas para mejorar la recuperación de detalles en tareas de restauración.
- **Perceptual Loss:** Función de pérdida basada en redes preentrenadas como VGG19, que evalúa la calidad de la imagen restaurada comparándola en un espacio de características profundas.
- **Peak Signal-to-Noise Ratio (PSNR):** Métrica utilizada para evaluar la calidad de una imagen restaurada, comparándola con la imagen original sin desenfoque.
- **Red Neuronal Convolucional (CNN):** Arquitectura de red neuronal especializada en el procesamiento de imágenes y datos estructurados en forma de cuadrícula.
- **Red Neuronal Residual (ResNet):** Tipo de arquitectura de redes neuronales profundas que incorpora conexiones residuales para evitar el problema de la desaparición del gradiente y mejorar el entrenamiento de redes profundas.
- **Retropropagación:** Algoritmo de entrenamiento de redes neuronales que ajusta los pesos mediante el cálculo del gradiente de la función de pérdida con respecto a cada parámetro.
- **SSIM (Índice de Similitud Estructural):** Índice que mide la similitud estructural entre dos imágenes, utilizado como métrica de evaluación en la restauración de imágenes.
- **Skip Connections:** Conexiones utilizadas en redes neuronales profundas que permiten la propagación de información entre capas, mejorando la estabilidad y evitando la degradación del gradiente.

Apéndice B

Manual de Usuario

Este capítulo ofrece una guía práctica para ejecutar la aplicación **Deepblurring**, destinada a la mejora de imágenes con modelos basados en *Deep Learning*. Toda la información aquí descrita también se encuentra resumida en el archivo `README.txt` incluido en el directorio raíz de la aplicación.

B.1. Requisitos previos

Antes de ejecutar la aplicación, el usuario debe asegurarse de cumplir con los siguientes requisitos:

- Tener instalado **Python 3.8** o una versión posterior.
- Tener instalado **Anaconda**.
- Aunque hay otras alternativas, se recomienda utilizar el software **Visual Studio Code** como entorno.
- Una vez instalado Anaconda, configura VS Code para que use el intérprete de Python de Anaconda añadiendo las siguientes rutas a la variable de entorno `PATH` (Panel de Control → Sistema → Configuración avanzada del sistema → Variables de entorno):
  - `C:\Users\TU_USUARIO\anaconda3`
  - `C:\Users\TU_USUARIO\anaconda3\Scripts`
  - `C:\Users\TU_USUARIO\anaconda3\Library\bin`
  - `C:\Users\TU_USUARIO\anaconda3\condabin`
  - `C:\Users\TU_USUARIO\AppData\Local\Programs\Python\Python313`
  - `C:\Users\TU_USUARIO\AppData\Local\Programs\Python\Python313\Scripts`

  Abre Visual Studio Code, y sigue las instrucciones de la Sección B.2.

B.2. Despliegue de la aplicación

A continuación, se detalla el proceso completo para ejecutar la aplicación desde cero:

1. **Abrir la terminal y situarse en el directorio raíz de la aplicación:**

   ```
   cd ruta/completa/a/Aplicacion
   ```

2. **Crear un entorno virtual conda con Python 3.8:**

   ```
   conda create --name NOMBRE_ENTORNO python=3.8
   ```

3. **Modificar permisos**:

   ```
   Set-ExecutionPolicy RemoteSigned -Scope CurrentUser
   ```

4. **Iniciar conda**:

   ```
   conda init
   ```

5. **Cerrar y abrir la terminal tras el comando anterior, y activar el entorno:**

   ```
   conda activate NOMBRE_ENTORNO
   ```

6. **Instalar las dependencias necesarias utilizando el archivo `requirements.txt` proporcionado en el proyecto.**

   ```
   pip install -r requirements.txt
   ```

7. **Ejecutar la instalación del módulo basicsr a través del archivo setup.py de modelos/restormer:**

   ```
   cd modelos/restormer
   python setup.py develop --no_cuda_ext
   cd ../..
   ```

8. **Iniciar la aplicación:**

   ```
   python main.py
   ```

   Al ejecutar este último comando, se abrirá automáticamente una ventana del navegador con la interfaz gráfica de la aplicación.

B.3. Uso de la aplicación

Una vez abierta la interfaz, el usuario puede seguir los siguientes pasos para utilizar la herramienta:

1. Subir una imagen o un archivo `.zip` que contenga varias imágenes (ver Figura B.1).

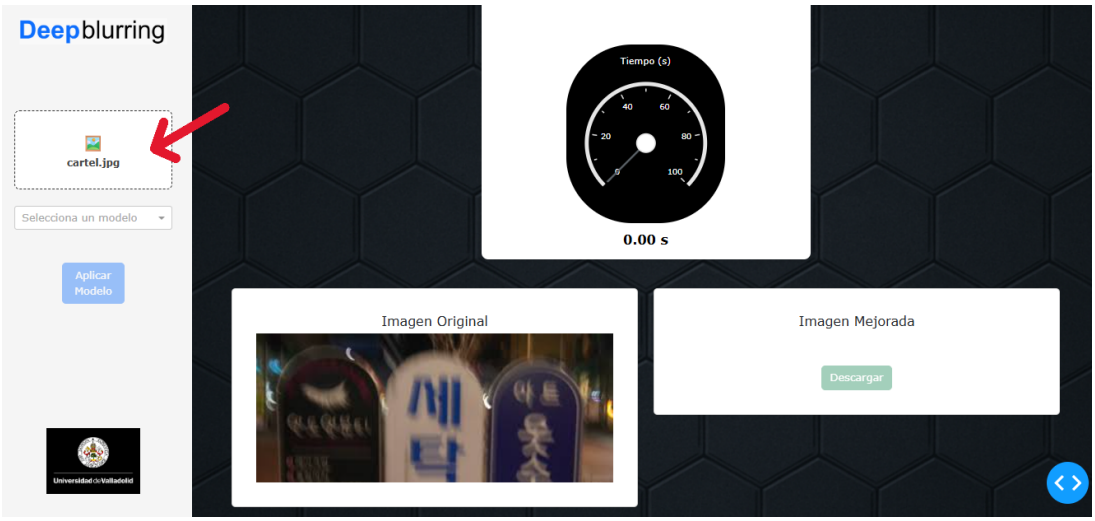


Figura B.1: Botón de subir imagen

2. Seleccionar uno de los modelos disponibles en el menú desplegable: `Restormer`, `NAFNet` o `DeblurGAN` (ver Figura B.2).

Figura B.2: Botón de seleccionar modelo

3. Pulsar el botón `Aplicar Modelo` (ver Figura B.3).

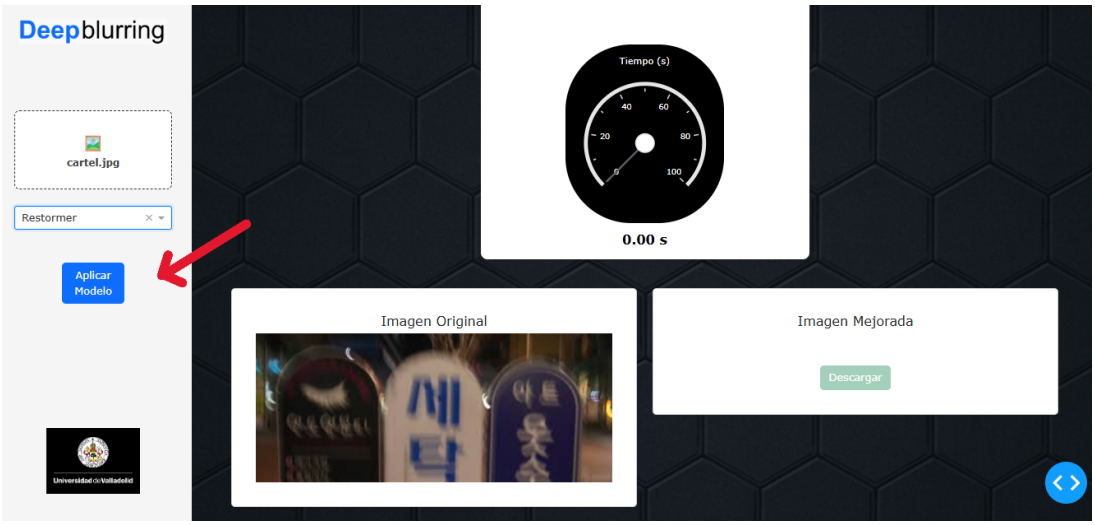


Figura B.3: Botón de aplicar modelo

4. Esperar unos segundos hasta que se muestre la imagen mejorada junto con las métricas de calidad SSIM, PSNR y tiempo. Tras esto, será posible realizar la descarga de la imagen restaurada (ver Figura B.4).

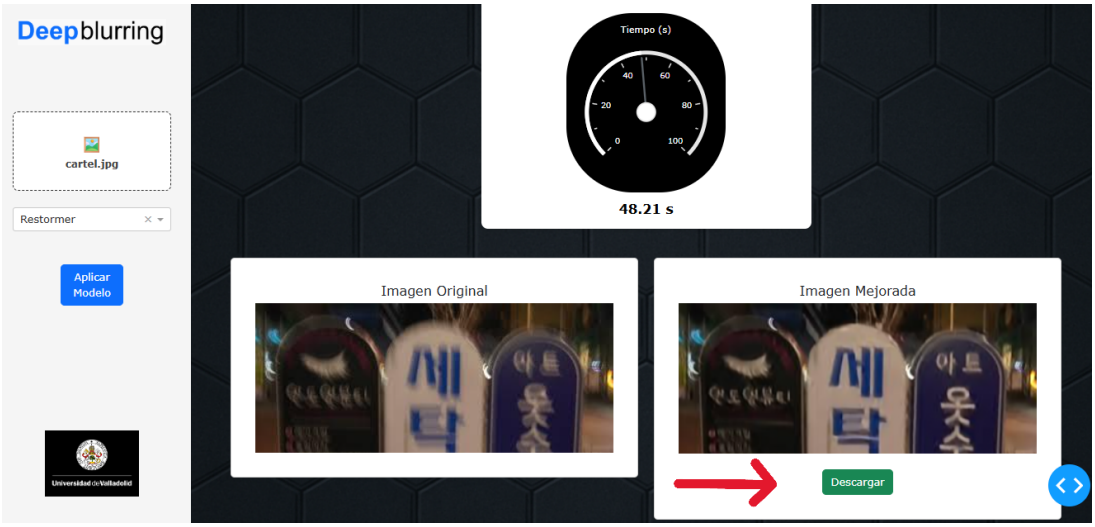


Figura B.4: Resultado de la aplicación

Las imágenes cargadas por el usuario se almacenan en la carpeta `imagenes/entrada`. Una vez procesadas, las imágenes mejoradas se guardan en `imagenes/salida` bajo el nombre original, seguido del sufijo `_modelo.png` (donde "modelo" es el nombre del modelo seleccionado por el usuario previamente).

Si el usuario sube una imagen con el mismo nombre que otra ya existente, el sistema genera automáticamente un nuevo nombre con sufijos numerados, como por ejemplo `foto(1).jpg`, `foto(2).jpg`, etc., para evitar sobrescribir archivos. El usuario también podrá navegar entre varias imágenes (en caso de subida mediante ZIP) y descargar cada imagen mejorada desde la propia interfaz web.

Bibliografía

[1] D. Martín de Andrés y N. Serrano Nieto. *«Apuntes Tema 1.2 Gestión de Proyectos Basados en las Tecnologías de la Información: El modelo PMI de la gestión de proyectos»*. Material docente. 2024-2025.

[2] Bepec Solutions. *«Deep learning is a type of machine learning that involves artificial neural networks with representation learning»*. 2018. URL: https://www.facebook.com/Bepecsolutions/posts/1696559853838655/.

[3] S. Bialek et al. «DanceCam: Atmospheric Turbulence Mitigation in Wide-Field Astronomical Images with Short-Exposure Video Streams». En: *arXiv preprint arXiv:2405.05250* (2024).

[4] S. A. Biyouki y H. Hwangbo. «A Comprehensive Survey on Deep Neural Image Deblurring». En: *Proceedings of an International Conference on Computer Vision and Image Processing*. 2023.

[5] A. Bregón Bregón. *«Apuntes tema 4 Sistemas Inteligentes»*. 2024-2025.

[6] A. Bregón Bregón. *«Apuntes tema 7 Sistemas Inteligentes»*. 2024-2025.

[7] A. Bregón Bregón. *«Apuntes tema 8 Sistemas Inteligentes»*. 2024-2025.

[8] K. Cho et al. «On the Properties of Neural Machine Translation: Encoder-Decoder Approaches». En: *arXiv preprint arXiv:1409.1259* (2014).

[9] Wikipedia contributors. «Mean Absolute Error». En: *Wikipedia, The Free Encyclopedia* (2024). URL: https://en.wikipedia.org/wiki/Mean_absolute_error.

[10] Wikipedia contributors. «Mean Squared Error». En: *Wikipedia, The Free Encyclopedia* (2024). URL: https://en.wikipedia.org/wiki/Mean_squared_error.

[11] J. Dong, S. Roth y B. Schiele. «Deep Wiener Deconvolution: Wiener Meets Deep Learning for Image Deblurring». En: *Advances in Neural Information Processing Systems* 33 (2020), págs. 1033-1044.

[12] Gobierno de España. *«¿Qué es la Inteligencia Artificial (IA)? - Plan de Recuperación, Transformación y Resiliencia»*. 2025. URL: https://planderecuperacion.gob.es/noticias/que-es-inteligencia-artificial-ia-prtr#:~:text=La%20inteligencia%20artificial%20(IA)%20es,el%20razonamiento%20y%20la%20percepci%C3%B3n..

[13] FutureLab. *«Introducción a las Redes Neuronales (Parte 1)»*. 2019. URL: https://futurelab.mx/redes%20neuronales/inteligencia%20artificial/2019/06/25/intro-a-redes-neuronales-pt-1/.

[14] I. Goodfellow, Y. Bengio y A. Courville. *«Deep Learning»*. MIT Press, 2016.

[15] I. Goodfellow et al. «Generative Adversarial Networks». En: *Advances in Neural Information Processing Systems*. Vol. 27. 2014.

[16] Scrum Guides. *«The 2020 Scrum Guide»*. 2020. URL: https://scrumguides.org/scrumguide.html.

[17] K. He et al. «Deep Residual Learning for Image Recognition». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, págs. 770-778.

[18] L. He, Y. Wang y Z. Xiang. «Support Driven Wavelet Frame-Based Image Deblurring». En: *arXiv preprint arXiv:1603.08108* (2016).

[19] A. Hore y D. Ziou. «Image quality metrics: PSNR vs. SSIM». En: *2010 20th International Conference on Pattern Recognition* (2010), págs. 2366-2369.

[20] Y. Huang, H. Zhang y D. Liang. «Plug-and-Play Gradient-Based Denoisers Applied to CT Image Reconstruction». En: *Applied Numerical Mathematics* 176 (2022).

[21] Project Management Institute. *«Guide to the Project Management Body of Knowledge (PMBOK Guide)»*. Sixth. Project Management Institute, 2017. ISBN: 9781628254518.

[22] KeepCoding. *«Función de Activación en Deep Learning»*. 2023. URL: https://keepcoding.io/blog/funcion-de-activacion-en-deep-learning/.

[23] K. Kim et al. «Improvement of Image Characteristics in High-Voltage Computed Tomography (CT) by Applying a Compressed-Sensing (CS)-Based Image Deblurring Scheme». En: *NDT E International* 84 (2016).

[24] O. Kupyn et al. «DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, págs. 8183-8192.

[25] R. Köhler et al. «Recording and Playback of Camera Shake: Benchmarking Blind Deconvolution with a Real-World Database». En: *European Conference on Computer Vision (ECCV)*. Springer, 2012.

[26] W.-S. Lai et al. «A Comparative Study for Single Image Blind Deblurring». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, págs. 1701-1709.

[27] K. Lakshmi, M. P. Singh y P. Priyanka. *«Figura 1. Recurrent Neural Networks»*. 2020. URL: https://www.researchgate.net/figure/Recurrent-Neural-Networks-41_fig1_350193297.

[28] A. Levin et al. «Understanding and Evaluating Blind Deconvolution Algorithms». En: *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2009, págs. 1964-1971.

[29] M. A. Martínez-Prieto et al. «Una metodología basada en prácticas ágiles para la realización de Trabajos Fin de Grado». En: *Actas de las XXVIII JENUI*. Vol. 8. 2023.

[30] W. S. McCulloch y W. Pitts. «A Logical Calculus of the Ideas Immanent in Nervous Activity». En: *Bulletin of Mathematical Biophysics* 5 (1943), págs. 115-133.

[31] S. Nah, T. H. Kim y K. M. Lee. *«Deep Multi-scale Convolutional Neural Network for Dynamic Scene Deblurring»*. 2017. URL: https://autonomousvision.github.io/motion-deblur/.

[32] S. Nah, T. H. Kim y K. M. Lee. «Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring». En: *CVPR*. Jul. de 2017.

[33] J. Narasimharao et al. «Restoration and Deblurring the Images by Using Blind Convolution Method». En: *Proceedings of Fourth International Conference on Computer and Communication Technologies* (2023), págs. 337-346.

[34] K. O'Shea y R. Nash. «An Introduction to Convolutional Neural Networks». En: *arXiv preprint arXiv:1511.08458* (2015).

[35] *«Problem Statement»*. Wikipedia. 2024. URL: https://en.wikipedia.org/wiki/Problem_statement.

[36] Proyecto IDIS. *«Red Generativa Antagónica (GAN)»*. 2023. URL: https://proyectoidis.org/red-generativa-antagonica-gan/.

[37] R. Raskar, A. Agrawal y J. Tumblin. «Coded Exposure Photography: Motion Deblurring Using Fluttered Shutter». En: *ACM Transactions on Graphics (TOG)*. Vol. 25. 3. ACM. 2006, págs. 795-804.

[38] L. Raut, R. Wakode y P. Talmale. «Overview on Kanban Methodology and its Implementation». En: *International Journal for Scientific Research Development* 3 (jul. de 2015), págs. 2518-2521.

[39] D. Ren et al. «Neural Blind Deconvolution Using Deep Priors». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[40] J. Rim et al. «Real-World Blur Dataset for Learning and Benchmarking Deblurring Algorithms». En: *European Conference on Computer Vision (ECCV)*. Springer, 2020, págs. 184-201.

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

[42] S. J. Russell y P. Norvig. *«Inteligencia Artificial: Un Enfoque Moderno»*. 2.ª ed. Madrid: Pearson Educación, 2004.

[43] *«Salario medio de un Científico de Datos en España»*. The Bridge. 2024. URL: https://thebridge.tech/blog/salario-data-scientist-espana.

[44] *«Salario medio de un Gestor de Proyectos en España»*. Glassdoor. 2024. URL: https://www.glassdoor.es/Sueldos/data-project-manager-sueldo-SRCH_KO0,20.htm.

[45] *«Salario promedio de un Analista de Datos en España»*. Talent.com. 2024. URL: https://es.talent.com/salary?job=analista+de+datos.

[46] O. Salinas, I. González y M. I. Morales. *«Figura 1. Descripción del funcionamiento de una red neuronal convolucional (CNN)»*. 2020. URL: https://www.researchgate.net/figure/Figura-1-Descripcion-del-funcionamiento-de-una-red-neuronal-convolucional-CNN-10_fig1_348825166.

[47] Z. Shen et al. «Human-Aware Motion Deblurring». En: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, págs. 5572-5581.

[48] Seguridad Social. *«Bases y tipos de cotización 2024»*. 2024. URL: https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores.

[49] S. Su et al. «Deep Video Deblurring for Hand-Held Cameras». En: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, págs. 1279-1288.

[50] L. Sun et al. «Edge-Based Blur Kernel Estimation Using Patch Priors». En: *IEEE International Conference on Computational Photography (ICCP)*. IEEE, 2013, págs. 1-8.

[51] X. Tao et al. «Scale-Recurrent Network for Deep Image Deblurring». En: *CVPR* (2018).

[52] D. Ulyanov, A. Vedaldi y V. Lempitsky. *«Deep Image Prior»*. Repositorio GitHub. 2018. URL: https://github.com/DmitryUlyanov/deep-image-prior.

[53] Dmitry Ulyanov, Andrea Vedaldi y Victor Lempitsky. «Deep Image Prior». En: *arXiv:1711.10925* (2017).

[54] A. Vaswani et al. «Attention is All You Need». En: *Advances in Neural Information Processing Systems (NeurIPS)*. 2017, págs. 5998-6008.

[55] J. Wang et al. «MCIDN: Deblurring Network for Metal Corrosion Images». En: *Applied Sciences* 14.11565 (2024). DOI: 10.3390/app142411565. URL: https://www.mdpi.com/journal/applsci.

[56] X. Wang, Y. Zhang, Q. Li et al. «Applying Deep Learning Image Enhancement Methods to Improve Person Re-Identification Models». En: *Neurocomputing* 533 (2023), págs. 152-162.

[57] Z. Wang et al. «Image quality assessment: From error visibility to structural similarity». En: *IEEE Transactions on Image Processing* 13.4 (2004), págs. 600-612.

[58] S. Waqas et al. «Restormer: Efficient Transformer for High-Resolution Image Restoration». En: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, págs. 5728-5739.

[59] Wikipedia. *«Industria 4.0»*. 2025. URL: https://es.wikipedia.org/wiki/Industria_4.0.

[60] Wikipedia. *«Red neuronal residual»*. 2025. URL: https://es.wikipedia.org/wiki/Red_neuronal_residual.

[61] S. Woo et al. «CBAM: Convolutional Block Attention Module». En: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, págs. 3-19.

[62] C. D. Wren. *«SelfDeblur: Self-Supervised Blind Image Deblurring»*. Repositorio GitHub. 2021. URL: https://github.com/csdwren/SelfDeblur.

[63] D. Wu et al. «L0 Regularized Image Deblurring Applied to Porosity Analysis in Industrial X-ray Computed Tomography». En: *11th Conference on Industrial Computed Tomography (iCT 2022)*. Feb. de 2022.

[64] K. Zhang et al. «Deep Image Deblurring: A Survey». En: (2023).

[65] J. Zhu et al. *«CycleGAN: Unpaired Image-to-Image Translation»*. Repositorio GitHub. 2017. URL: https://github.com/junyanz/CycleGAN.

[66] J.-Y. Zhu et al. «Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks». En: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017.

[67] Song-Chun Zhu y Ying Nian Wu. «Textures». En: *Computer Vision: Statistical Models for Marr's Paradigm*. Springer, 2023. URL: https://link.springer.com/chapter/10.1007/978-3-030-96530-3_3.