



ESCUELA POLITÉCNICA



UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

INGENIERÍA INFORMÁTICA EN INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

**Aplicación de técnicas de predicción de anomalías en turbinas eólicas
combinando redes neuronales *U-net* y *LSTM***

Jorge del Castillo Gómez

Junio, 2024



ESCUELA POLITÉCNICA



UNIVERSIDAD DE EXTREMADURA

ESCUELA POLITÉCNICA

INGENIERÍA INFORMÁTICA EN INGENIERÍA DEL SOFTWARE

TRABAJO FIN DE GRADO

**Aplicación de técnicas de predicción de anomalías en turbinas eólicas
combinando redes neuronales *U-net* y *LSTM***

Autor: Jorge del Castillo Gómez

Tutor: Félix Rodríguez Rodríguez

Co-Tutor: Jose Luis González Sánchez

Índice general

Resumen	1
1. Introducción	3
1.1. Motivación del presente trabajo	6
1.2. Objetivos	7
1.3. Organización temporal del trabajo	9
1.4. Estructura del documento	11
2. Antecedentes	13
2.1. Trabajos previos en series temporales de datos y en la predicción de fallos en motores	14
2.2. Trabajos previos en el reconocimiento de acciones humanas en vídeos	24
3. Entorno de trabajo	29
3.1. <i>Hardware</i>	29
3.2. <i>Software</i>	30
4. Conocimiento del dominio físico y de los datos	33
4.1. Entorno físico	33
4.1.1. Partes de una turbina eólica	34
4.1.2. Sistema SCADA de captura de datos	36
4.1.3. Modelo físico de una turbina eólica	39
4.2. Análisis de los datos	47

ÍNDICE GENERAL

4.2.1. Ángulo de <i>Pitch</i>	48
4.2.2. Velocidad del Viento	50
4.2.3. Potencia	52
4.2.4. Correlación entre las variables	57
5. Implementación y desarrollo de la arquitectura <i>U-Net + LSTM</i>	59
5.1. Preprocesado de los datos	60
5.1.1. Conversión de las series temporales de datos a imágenes	60
5.1.2. Proceso de etiquetado del conjunto de imágenes	63
5.2. Sistema de autoetiquetado. Arquitectura <i>U-Net</i>	66
5.2.1. Por qué la arquitectura <i>U-Net</i>	67
5.2.2. Estructura de la arquitectura <i>U-Net</i>	68
5.2.3. Arquitectura <i>U-Net</i> previa	71
5.3. Modelado <i>baseline</i> inicial	76
5.3.1. Generación de la serie temporal de etiquetas	76
5.3.2. División del conjunto de datos	77
5.3.3. Modelo de regresión con <i>WEKA</i>	78
5.3.4. Modelos de regresión con red neuronal <i>LSTM</i>	80
5.3.4.1. Red neuronal <i>LSTM</i> no <i>stateful</i>	80
5.3.4.2. Red neuronal <i>LSTM stateful</i> en diferido	84
5.3.5. Consecuencias del modelado <i>baseline</i>	87
5.4. Modelado con la arquitectura <i>U-Net + LSTM</i>	87
5.4.1. Configuración de los hiperparámetros	91
5.5. Modelado con la arquitectura <i>ConvLSTM</i>	94
6. Resultados	97
6.1. Procedimiento de la fase de prueba	97
6.2. Definición del resultado deseado	98
6.3. Resultados de la arquitectura <i>U-Net + LSTM</i> propuesta	99
6.4. Resultados de la arquitectura <i>ConvLSTM</i> propuesta	107

ÍNDICE GENERAL

7. Conclusiones y trabajos futuros	109
7.1. Conclusiones	109
7.1.1. Conclusiones relacionadas con el proyecto	109
7.1.2. Conclusiones personales	110
7.2. Trabajos futuros	111
Anexos	113
A. Conceptos fundamentales del <i>Deep Learning</i>	115
A.1. Introducción a <i>Deep Learning</i>	116
A.2. Interpretación matemática y geométrica de una neurona	119
A.2.1. Interpretación matemática de una neurona	120
A.2.2. Interpretación de una neurona artificial mediante la hipótesis del <i>manifold</i>	122
A.3. Proceso iterativo de aprendizaje de una Red Neuronal	125
A.3.1. Optimizador y función pérdida	126
A.3.2. Hiperparámetros	129
A.3.3. Métricas	131
A.4. Tipos de Redes Neuronales	134
A.4.1. Perceptrón Multicapa	134
A.4.2. RN Convolucionales	135
A.4.3. RN Recurrentes	140
B. Desarrollos complementarios	142
B.1. Análisis del conjunto de etiquetas generado por la arquitectura <i>U-Net</i>	142
B.2. Estructura por capas del modelo <i>U-Net + LSTM</i> propuesto	146
Bibliografía y referencias	153

Índice de tablas

4.1. Variables y unidades de medida asociadas.	39
5.1. Descripción de las seis etiquetas con el número de imágenes asociado.	66
5.2. Comparación entre las etiquetas reales de las imágenes y las generadas por el modelo <i>U-Net</i> para las 25 turbinas eólicas.	77
5.3. Distribución de las etiquetas en la serie temporal de la turbina eólica <i>WT2</i> en los años 2016 y 2017.	78
B.1. Recuento del número de etiquetas de las 25 series temporales de etiquetas generadas por el modelo <i>U-Net</i>	143
B.2. Recuento del número de etiquetas en el año 2017 de las 25 series temporales generadas por el modelo <i>U-Net</i>	144
B.3. Recuento del número de etiquetas en el año 2016 de las 25 series temporales generadas por el modelo <i>U-Net</i>	145

Índice de figuras

1.1.	Emisión de CO_2 por la actividad industrial de 1870 a 2022	4
1.2.	Generación de energía por tecnologías en 2030	5
1.3.	Distribución de la energía eléctrica en España en 2022	6
1.4.	Planificación temporal del proyecto por fases.	10
2.1.	Ejemplo de unidad de textura generada mediante el patrón LBP sobre una matriz de vecindad 3×3	15
2.2.	Ejemplo de Espectro de Textura de una imagen	16
2.3.	Conversión de una señal a una imagen de tamaño $M \times N$	17
2.4.	Cálculo numérico de una unidad de textura a partir de una matriz de entrada 3×3	18
2.5.	Configuración de parámetros de 3 operadores LBP diferentes	18
2.6.	Metodología propuesta en la tarea de detección y clasificación de anomalías en turbinas eólicas	20
2.7.	Validación cruzada en K iteraciones con $K = 4$	21
2.8.	Arquitectura U -Net	22
2.9.	Evaluaciones del modelo U -Net con otros modelos de clasificación mediante las métricas exactitud y $F1$ -Score	23
2.10.	Etiquetado mediante ventana deslizante vs etiquetado denso	24
2.11.	Resultados de la exactitud y $f1$ -score sobre la evaluación del modelo U -Net	25
2.12.	Arquitectura base $LSTM-CNN$	26

ÍNDICE DE FIGURAS

2.13. Arquitectura con mecanismo de atención	27
2.14. Resultados para la métrica de precisión de los modelos base y con atención	27
4.1. Principales componentes de una turbina eólica	34
4.2. Arquitectura por niveles de un sistema <i>SCADA</i>	37
4.3. Coeficiente de potencia C_p como una función entre el <i>tip-speed ratio</i> λ y el ángulo de las palas β	43
4.4. Coeficientes de Betz asociado a 3 diferentes tipos de turbinas respecto al <i>tip-speed ratio</i>	44
4.5. Regiones ideales en la curva de potencia de una turbina eólica.	45
4.6. Curva de potencia de una turbina eólica marcada con 5 tipos de anomalías en su comportamiento.	46
4.7. Eficiencia de producción de una turbina eólica.	47
4.8. Ángulo de <i>pitch</i> medio asociado a la turbina eólica durante 8 años. . .	48
4.9. Ángulo de <i>pitch</i> medio asociado a la turbina eólica en relación con cada mes.	49
4.10. Ángulo de <i>pitch</i> medio asociado a la turbina eólica durante un día (9 horas).	49
4.11. Velocidad media del viento de la turbina eólica durante 8 años.	50
4.13. Velocidad media del viento asociado a la turbina eólica durante un día (9 horas).	51
4.12. Velocidad media del viento de la turbina eólica en relación con cada mes.	51
4.14. Potencia media generada por la turbina eólica durante 8 años.	52
4.16. Potencia media generada por la turbina eólica durante un día (9 horas). .	53
4.15. Potencia media generada por la turbina eólica en relación con cada mes.	53
4.17. Curva de la potencia media generada en función del ángulo de <i>pitch</i> . .	54
4.18. Curva de la potencia media generada en función de la velocidad media del viento.	55

ÍNDICE DE FIGURAS

4.19. Curva de la potencia máxima generada en función de la velocidad media del viento.	56
4.20. Curva de la potencia mínima generada en función de la velocidad media del viento.	56
4.21. Matriz de correlación de las 5 variables más importantes del <i>dataset</i> . .	58
5.1. Conversión de una señal a una imagen de tamaño $M \times N$	61
5.2. Partición con y sin solapamiento mediante la ventana de tiempo deslizante.	62
5.3. Ventana de tiempo de 64 <i>timestamps</i> de una serie temporal de potencia activa media convertida a imagen.	64
5.4. Clasificación de la curva de potencia media activa de una turbina eólica.	65
5.5. Arquitectura <i>U-Net</i>	69
5.6. Arquitectura <i>U-Net</i> utilizada como punto de partida.	72
5.7. Matriz de confusión del modelo <i>U-Net</i> tomado como punto de partida.	74
5.8. Arquitectura por capas <i>U-Net</i> tomada como punto de partida.	75
5.9. Proceso de autoetiquetado realizado por medio de la <i>U-Net</i>	76
5.10. Comportamiento del modelo de regresión con <i>WEKA</i> en un intervalo de tiempo concreto del conjunto de datos del año 2016.	80
5.11. Arquitectura <i>LSTM</i> no <i>stateful baseline</i> inicialmente propuesta.	83
5.12. Predicción de secuencias de 10 valores futuros con el modelo <i>LSTM</i> no <i>stateful baseline</i> sobre el conjunto de datos del año 2016.	84
5.13. Predicción de secuencias de 10 valores futuros con el modelo <i>LSTM</i> <i>stateful baseline</i> sobre el conjunto de datos del año 2016.	86
5.14. Esquema de la arquitectura <i>U-Net + LSTM</i> propuesta.	88
5.15. Conjunto de capas transferidas desde el modelo <i>U-Net</i> previo a la arquitectura <i>U-Net + LSTM</i> propuesta.	90
5.16. Estructura en capas de la segunda fase <i>LSTM</i> del modelo <i>U-Net + LSTM</i> propuesto.	92
5.17. Estructura en capas del modelo <i>ConvLSTM</i> propuesto	95

ÍNDICE DE FIGURAS

6.1. Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 1 epoch.	100
6.2. Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 2 epochs.	101
6.3. Rango ampliado (600 – 900) de predicciones de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 2 epochs.	102
6.4. Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 4 epochs.	103
6.5. Rango ampliado (75 – 600, etiquetas 0 – 2) de predicciones de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 4 epochs.	104
6.6. Rango ampliado (675 – 975) de predicciones de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 4 epochs.	104
6.7. Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>U-Net + LSTM</i> y 10 epochs.	105
6.8. Rango ampliado de predicciones 700 - 950 sobre las etiquetas con valor 5 y 0.	106
6.9. Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo <i>ConvLSTM</i>	107
A.1. Diagrama de un perceptrón con cinco señales de entrada	117
A.2. Taxonomía de la IA, <i>DL</i> y <i>ML</i>	118
A.3. Representación gráfica de los datos y la recta que los separa	121
A.4. Representación gráfica de la función sigmoide	122
A.5. Representación gráfica de la suma de dos vectores	123
A.6. Representación gráfica de la translación de un tensor <i>K</i>	124
A.7. Representación gráfica de la transformación afín	124
A.8. Esquema del proceso de aprendizaje de una red neuronal	126

ÍNDICE DE FIGURAS

A.9. Ejemplo de función convexa $y = x^2$	127
A.10. Representación gráfica de una función pérdida que depende de dos parámetros de la red.	128
A.11. Perceptrón Multicapa con una capa de entrada, dos capas ocultas y una capa de salida.	135
A.12. A la izquierda, patrones generales aprendidos por una capa convolucional más general. A la derecha, patrones complejos de una capa convolucional más profunda.	136
A.13. Operación de convolución sobre un tensor de entrada 5×5 con un kernel 3×3	137
A.14. Operación de <i>max-pooling</i> sobre un tensor de entrada 4×4	138
A.15. Operación de convolución transpuesta sobre una entrada 2×2 y con un filtro 2×2	139
A.16. Operación de aplanamiento de un tensor 3×3 en un vector.	140
A.17. Desenrollado temporal de una neurona recurrente.	141
B.1. Primera parte de la arquitectura <i>U-Net + LSTM</i>	146
B.2. Segunda parte de la arquitectura <i>U-Net + LSTM</i>	147
B.3. Tercera parte de la arquitectura <i>U-Net + LSTM</i>	148



ÍNDICE DE FIGURAS

Resumen

La energía eólica desempeña un papel crucial en la transición hacia fuentes de energía más limpias y sostenibles. Es fundamental el mantenimiento adecuado de las máquinas y la anticipación de posibles fallos internos para prevenir riesgos de seguridad y minimizar los costos operativos asociados con las reparaciones.

En este Trabajo Fin de Grado se aborda la predicción de anomalías en turbinas eólicas situadas en un parque eólico, empleando técnicas de procesamiento de series temporales de datos en *Deep Learning*.

A través de la implementación y evaluación de diversos modelos, se ha demostrado que los enfoques tradicionales y simples son insuficientes para predecir de manera satisfactoria estas anomalías. Sin embargo, a pesar de no alcanzar los resultados deseados, se ha establecido una base sólida para futuras investigaciones. El modelo más complejo propuesto en el trabajo, utilizando una combinación de *U-Net* y *LSTM*, ha mostrado prometedores indicios, señalando el camino hacia la exploración de modelos similares en futuros trabajos. Aunque el proyecto actual no haya logrado los resultados esperados, proporciona la dirección para futuras investigaciones en el campo de la predicción de anomalías en turbinas eólicas.



RESUMEN

Capítulo 1

Introducción

Si echamos la mirada atrás, a lo largo de la historia la raza humana ha denotado egoísmo, ambición, ansiedad y poder. Tras el estallido de la segunda revolución industrial, un grupo selecto de personas, empapadas de estas cuatro cualidades, alcanzaron grandes fortunas injustamente, sin escrúpulos: “El fin justifica los medios”, decían. No tuvieron ningún reparo en emitir cantidades inmensas de gases de efecto invernadero en la atmósfera, como el dióxido de carbono. Es el presente y las futuras generaciones venideras las que sufrirán las consecuencias del maltrato ambiental que sigue recibiendo el planeta.

La Figura 1.1, elaborada por [Statista \(2023b\)](#), refleja la evolución de las emisiones globales de CO_2 procedentes de la actividad industrial y de los combustibles fósiles desde la segunda revolución industrial de 1870 hasta 2022. La gráfica muestra una tendencia creciente de los niveles de CO_2 en el último siglo, una situación más que preocupante y alarmante.

En nuestros tiempos todo el mundo ha oído hablar alguna vez de la energía nuclear producida en centrales nucleares, de la energía hidráulica producida en los embalses, de la energía geotérmica que proviene del calor generado por las capas internas de la Tierra... No es necesario mencionar todas las fuentes de energías que aparecen en el *Telediario* y atrapan la atención de los televidentes. Sin embargo, debemos reflexionar sobre cuales no aparecen, cuales no tienen un interés mediático, no enriquecen y no

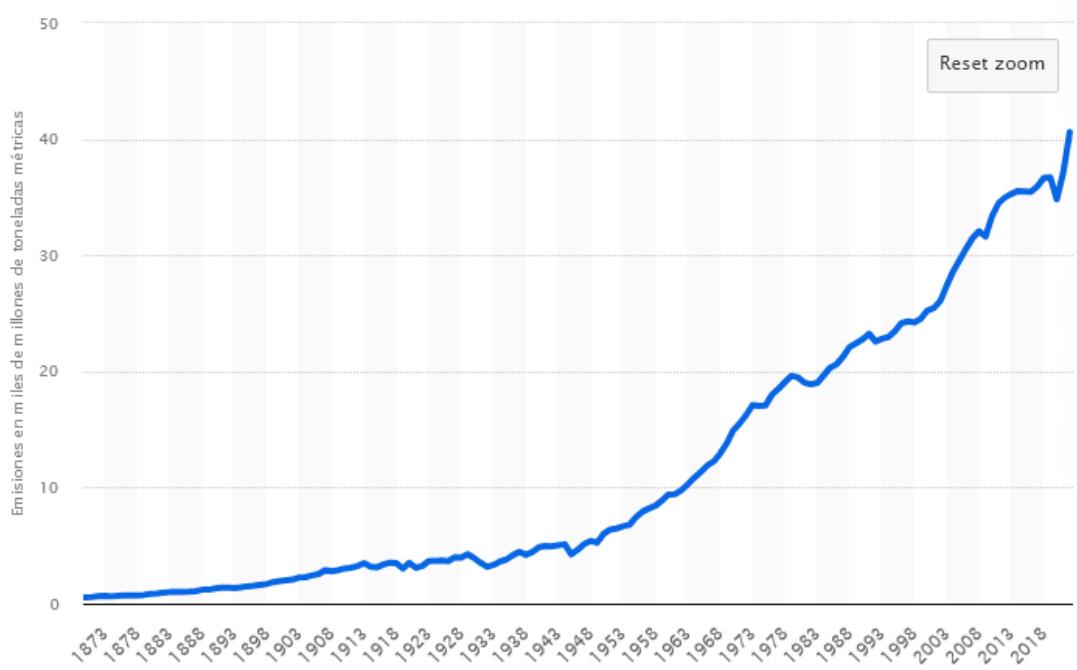


Figura 1.1: Emisión de CO_2 por la actividad industrial de 1870 a 2022. Fuente: [Statista \(2023b\)](#)

atraen votos. Hablo de las fuentes de energía renovables, y más concretamente, de la eólica.

No existe un interés social por la energía eólica. A contracorriente de la opinión popular, las universidades y sus grupos de investigación continúan la labor de maximizar la producción de electricidad a partir de fuentes de energía limpias, respetuosas con el medio ambiente. Desde los 2000s diversas potencias mundiales han establecido un objetivo común de priorizar la energía eólica en sus respectivos planes y estrategias energéticas para el año 2030.

Según recoge [United States Department of Energy \(2008\)](#) en el informe “*20% wind energy by 2030: increasing wind energy’s contribution to U.S. electricity supply*” (20% de energía eólica en 2030: aumentando la contribución de la energía eólica al suministro eléctrico de los Estados Unidos), el plan estratégico que sigue EE.UU. es generar el 20% de la electricidad del país procedente de la energía eólica en 2030. El plan que se propone desde 2008 supone una alteración de la generación de electricidad, como se muestra en la siguiente Figura 1.2. En el escenario de 2030, el viento proveerá

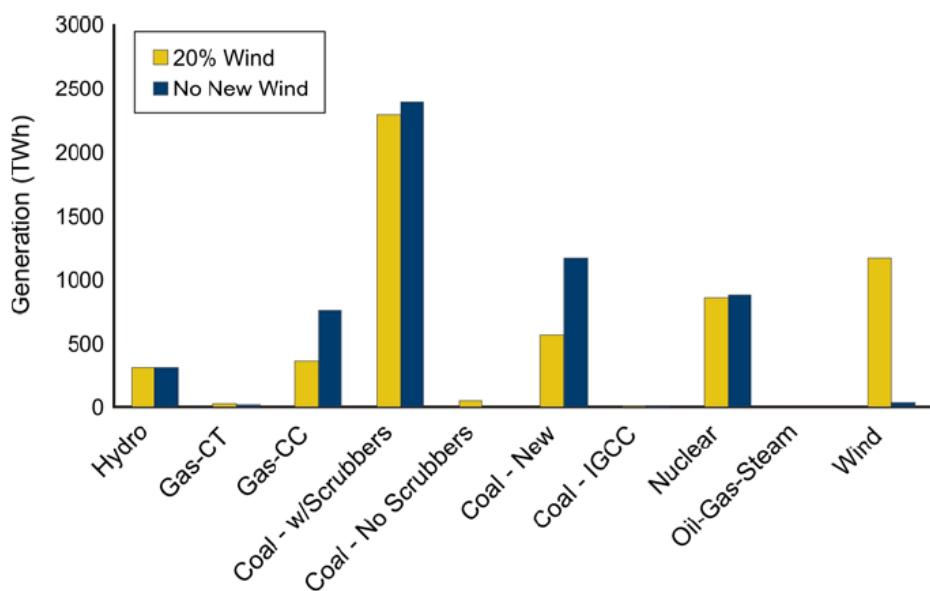


Figura 1.2: Generación de energía por tecnologías en 2030. Fuente: [United States Department of Energy \(2008\)](#)

la suficiente energía para sustituir el 50 % de la electricidad usando el consumo de gas natural y el 18 % del consumo de carbón.

La principal meta de EE.UU. es reducir la dependencia en los combustibles fósiles, sustituyéndolos por fuentes de energía renovable, como el viento. La Figura 1.2 respalda este escenario de generación de electricidad del país en 2030. Se puede observar la diferencia de *TWh*¹ entre *Coal - New* y *Gas-CC* respecto al plan estratégico previsto con el viento.

En 2006 el sector de la energía eólica europeo organizó la plataforma europea de energía eólica o *European Wind Energy Technology Platform (TPWind)*². El principal objetivo de esta iniciativa es reducir los costes sociales, medioambientales y tecnológicos de la energía eólica. *Wind (2008)* presenta un plan estratégico para convertir esta energía como la fuente predominante del mercado energético a nivel continental y liderar el mercado energético global. Definen el objetivo de cubrir el 10 % del consumo de energía mediante las plantas de turbinas eólicas *offshore*³.

¹**TWh**: es una unidad de energía equivalente a un billón de varios-hora.

²**TPWind**: es una iniciativa que reúne a diversos actores del sector eólico en Europa, incluyendo a la industria, investigación, los responsables políticos y otras partes interesadas.

³**Offshore**: *offshore* son las turbinas eólicas localizadas en el mar.

1.1. MOTIVACIÓN DEL PRESENTE TRABAJO

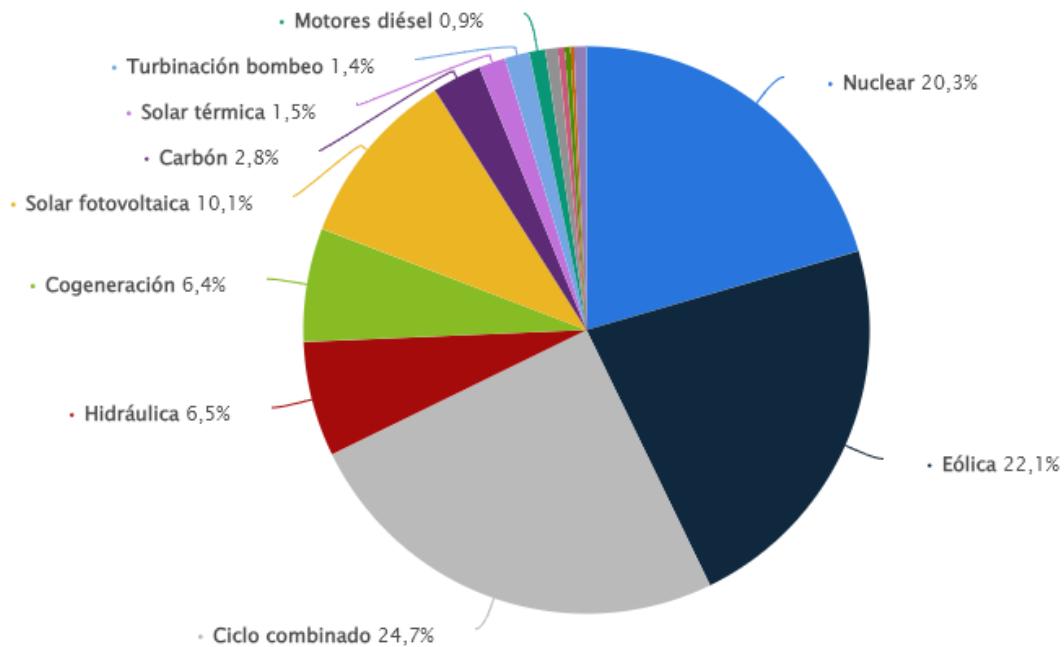


Figura 1.3: Distribución de la energía eléctrica en España en 2022. Fuente: [Statista \(2023a\)](#)

Un ejemplo claro de actuación es el indicado por Yang y cols. (2022) en su artículo “*Analysis of the Development of Distributed Wind Power in China*” (Análisis del desarrollo de la Energía Eólica Distribuida en China), en el que el plan energético de China es reducir el uso de combustibles fósiles, incrementando el consumo de fuentes no fósiles hasta un 25 % en 2030. Planean que las instalaciones de producción de energía eólica y solar alcancen los 1,2 billones de kW en 2030. Cómo maximizar la eficiencia y acelerar la transformación energética de China se convierte en un área de investigación destacada dentro del campo de las energías renovables.

En definitiva, existe una tendencia significativa por desarrollar localmente el sector de energías renovables.

1.1. Motivación del presente trabajo

En este punto ya se es consciente de la importancia en invertir e investigar en el sector de la energía eólica. Si nos fijamos en la Figura 1.3, también realizada por

1.2. OBJETIVOS

Statista (2023a), que refleja la distribución porcentual de la generación de energía eléctrica en España en 2022, existe una diferencia notable de la energía eólica respecto a sus homólogas del grupo de energías renovables. Además, si nos hacemos eco de lo que Black y cols. (2021) apuntan en su artículo “*Condition monitoring systems: a systematic literature review on machine-learning methods improving offshore-wind turbine operational management*” (Sistema de monitorización de condiciones: una revisión sistemática de la literatura sobre métodos de aprendizaje automático que mejoran la gestión operativa de turbinas *offshore*): las granjas de turbinas eólicas *offshore* necesitan de una inversión en el mantenimiento muy elevada, y el principal factor que permite el auge de este tipo de máquinas (ellos hablan en el entorno marino, *offshore*, pero también es trasladable al terrestre, *onshore*) es una metodología predictiva que contribuya a la monitorización de la salud estructural de todo el sistema, denominado monitorización de condiciones.

Por todo ello, no es nada descabellado plantear que, si a partir de la información recogida de la fuente de captación de datos, se logra inferir y predecir la salud de los componentes de la turbina, es del todo determinante y necesario establecer los procesos analíticos indispensables para implantar una tarea de mantenimiento exitosa basada en las irregularidades monitorizadas.

1.2. Objetivos

Bajo las circunstancias descritas en el comienzo del capítulo, el actual Trabajo Fin de Grado (TFG) toma relevancia. La detección y predicción de anomalías en turbinas eólicas juega un papel fundamental en la reducción de costos de mantenimiento y en el éxito del extenso ciclo de vida que estas máquinas deben tener. La disciplina *Deep Learning (DL)*, dentro de la rama *Machine Learning (ML)*, en el campo de la Inteligencia Artificial (IA), desempeña un papel clave en el presente trabajo.

El objetivo primordial del TFG se enfoca en desarrollar un sistema de predicción temprana de anomalías en turbinas eólicas mediante técnicas de *DL*. Para llevar a cabo este trabajo experimental, se definen una serie de objetivos específicos con el propósito

1.2. OBJETIVOS

de cumplirlos:

- Conocimiento previo de conceptos básicos de *Deep Learning*: historia, tipos de Redes Neuronales (RN), funciones de activación, optimizadores.
- Estudiar el funcionamiento interno de las turbinas eólicas, el modelo físico y los componentes mecánicos que tienen una mayor influencia en el rendimiento de la máquina.
- Conocer los sistemas *SCADA*⁴, herramientas claves de la automatización industrial, poniendo el foco en la obtención de datos sensoriales.
- Investigar sobre trabajos previos de *Machine Learning / Deep Learning* en el campo de detección de anomalías en turbinas eólicas.
 - Explorar trabajos previos relacionados con la transformaciones de series temporales de datos en imágenes sintéticas.
 - Explorar trabajos previos relacionados con la detección de fallos en las turbinas eólicas.
- Aprender técnicas de extracción y preprocesado de datos en el campo de *Machine Learning*.
- Analizar y comprender distintas técnicas de clasificación mediante RN.
- Diseñar e implementar modelos de aprendizaje profundo con RN.
- Investigar en técnicas optimización de modelos de *Deep Learning* y métricas de evaluación.

⁴*SCADA*: *Supervisory Control And Data Acquisition* – sistema para el Control Supervisor y de Adquisición de Datos.

1.3. ORGANIZACIÓN TEMPORAL DEL TRABAJO

1.3. Organización temporal del trabajo

Para la consecución de los objetivos se ha seguido la planificación temporal ilustrada en la Figura 1.4. En la imagen, la planificación se divide en 4 fases diferenciadas, que se desglosan entre los meses de octubre y abril:

- **Estudio:** en esta fase se abarcan las tareas de formación y aprendizaje autónomo, investigación y lecturas para adquirir el conocimiento necesario para desarrollar el proyecto.
- **Preprocesado de los datos:** previo a la fase de implementación, es crucial analizar los datos que se van a manejar, cómo están representados, qué información se puede extraer de ellos y procesarlos en un formato legible para las redes neuronales.
- **Implementación:** esta fase se centra en la implementación de diversos enfoques de técnicas de *ML* y *DL* para el desarrollo del sistema de predicción de anomalías en turbinas eólicas.
- **Memoria:** la fase final consiste en la redacción de la memoria del proyecto, una vez que el problema está definido y se ha adquirido el contexto suficiente.

Durante los meses de octubre y diciembre, el número de horas promedio estipulado es de 3 horas al día, mientras que en los meses de enero y abril se han aumentado entre 6 y 7 horas diarias.

1.3. ORGANIZACIÓN TEMPORAL DEL TRABAJO

Planificación del trabajo Fin de Grado

FASES	OCTUBRE	NOVIEMBRE	DICIEMBRE	ENERO	FEBRERO	MARZO	ABRIL
ESTUDIO	Lectura y estudio del libro de texto "Python Deep Learning" de Jordi Torres. Refuerzo del conocimiento a través de recursos en línea (blogs, videos, repositorios de código).	Lectura y estudio del libro de texto "Deep Learning with Python" de Francois Fleuret. Refuerzo del conocimiento a través de recursos en línea (blogs, videos, repositorios de código).	Estudio de artículos de investigación asociados al análisis de texturas y extracción de características en imágenes.	Investigación de artículos académicos relacionados con sistemas de detección y predicción de anomalías en turbinas eólicas.	Investigación de artículos de investigación asociados a redes neuronales LSTM para el procesamiento de series temporales de datos.	Lectura de artículos de investigación relacionados con arquitecturas U-Net y LSTM en el área de Reconocimiento de Acciones Humanas en videos (HAR).	Investigación de artículos académicos relacionados con modelos ConvLSTM.
PREPROCESADO DE LOS DATOS					Análisis y visualización de los datos.		
IMPLEMENTACIÓN						Implementación de la arquitectura de red neuronal U-Net + LSTM.	Implementación de la arquitectura de red neuronal ConvLSTM.
MEMORIA					Comienzo de Memoria.		

Figura 1.4: Planificación temporal del proyecto por fases.

1.4. ESTRUCTURA DEL DOCUMENTO

1.4. Estructura del documento

En el presente apartado, se le proporciona al lector la estructura que se sigue del resto del documento para facilitarle la comprensión del mismo. Las secciones restantes son:

- **Capítulo 2. Antecedentes.** En este capítulo, se reseñan trabajos previos existentes de *DL* sobre la detección temprana de anomalías en turbinas eólicas mediante redes neuronales, los cuales sirven de punto de partida para la realización del actual proyecto.
- **Capítulo 3. Entorno de trabajo.** Se detallan las especificaciones *hardware* y *software* que constituyen el entorno de trabajo para la realización del trabajo.
- **Capítulo 4. Conocimiento del dominio físico y de los datos.** Este capítulo pormenoriza los aspectos generales que han de conocerse para comprender correctamente el presente trabajo. En él, se abordan los sistemas involucrados en la obtención de datos y el análisis de los datos, así como el modelo físico de una turbina eólica.
- **Capítulo 5. Implementación y desarrollo de la arquitectura *U-Net + LSTM*.** Se describen aspectos claves como el análisis de los datos y el diseño completo de la arquitectura *U-Net* con capas *LSTM*⁵.
- **Capítulo 6. Resultados.** Se interpretan y significan los resultados obtenidos después de diseñar el sistema de predicción de anomalías en la fase de implementación.
- **Capítulo 7. Conclusiones y trabajos futuros.** Este último capítulo abarca las conclusiones del proyecto, posibles desviaciones futuras y reflexiones personales, ofreciendo al lector una visión completa del proceso llevado a cabo en el TFG.

⁵*LSTM*: Long Short Term Memory – Memoria a corto plazo

1.4. ESTRUCTURA DEL DOCUMENTO

- **Apéndice A. Conceptos fundamentales del Deep Learning.** Expone aquellos contenidos de apoyo que facilitan la comprensión del TFG. Se proporciona una visión detallada de los conceptos fundamentales de *DL*.
- **Apéndice B. Desarrollos complementarios.** Presenta contenidos adicionales que desglosan partes del trabajo que son importantes para ampliar la comprensión de la línea principal de investigación.

Capítulo 2

Antecedentes

En la presente capítulo, se detallan algunos trabajos previos de *Machine Learning* y *Deep Learning* que han tenido un impacto significativo en la industria eólica.

El apartado 2.1 aborda trabajos en la transformación de series temporales de datos sensoriales a imágenes sintéticas, así como en sistemas de predicción de anomalías en el comportamiento de las turbinas eólicas (donde ‘sensorial’ hace referencia a vibraciones de los componentes mecánicos).

El apartado 2.2 se centra en investigaciones recientes relacionadas con el reconocimiento de acciones humanas en vídeos, también conocido como *HAR* (*Human Action Recognition*), dentro del área de visión por computador¹. En el campo de la energía eólica, no existen estudios enfocados en la predicción temprana de anomalías de las turbinas eólicas utilizando arquitecturas convolucionales con mecanismos de atención. Estas investigaciones en el campo *HAR*, complementando el conocimiento base expuesto en el primer apartado, aportan un enfoque innovador e investigador al presente trabajo.

¹**Visión por computador:** conjunto de técnicas y herramientas que permiten a las máquinas captar imágenes del mundo real, procesarlas y generar información a través de ellas.

2.1. Trabajos previos en series temporales de datos y en la predicción de fallos en motores

Uno de los desafíos más complicados que presenta la industria eólica es la fiabilidad y consistencia de los datos procedentes de la monitorización de las turbinas eólicas. El ruido generado por los componentes mecánicos, durante un tiempo, ha sido un reto difícil de solucionar.

El origen de la solución al ruido en señales de los componentes mecánicos se remonta a principios de la década de 1990, mediante un operador de texturas sobre imágenes. Sin embargo, hasta el día de hoy, no existe un consenso en la definición formal del concepto de textura.

[Wang y He \(1990\)](#) en su investigación “*Texture classification using texture spectrum*” (“Clasificación de texturas usando el espectro de texturas”), exponen una técnica de análisis de texturas en imágenes denominada “*Local Binary Pattern*” o *LBP* (Patrón Local Binario). En este trabajo, definen la textura como una forma de identificar la superficie de un fenómeno que aparece en una imagen. Explican que la textura de una imagen puede descomponerse en unidades más pequeñas llamadas unidades de textura o “*Texture Unit*”. Cada unidad de textura la representan por 8 elementos, los cuales toman uno de tres posibles valores (0, 1, 2) obtenidos a partir de una región de píxeles 3×3 de la imagen de entrada.

La Figura 2.1, del mismo artículo de [Wang y He \(1990\)](#), muestra un ejemplo de transformación de una región de 3×3 píxeles en una unidad de textura. Una unidad de textura es considerada la unidad completa más pequeña que define mejor el aspecto de textura en todas las ocho direcciones desde el píxel central. La idea es calcular el valor que le corresponde a cada uno de los 8 elementos en base al valor de intensidad del píxel central respecto a cada píxel vecino.

Argumentan que *LBP* es una técnica invariable ante cambios de la luz incidente en las imágenes porque las variaciones en la luz conservan las relaciones entre las intensidades de los píxeles. En otras palabras, el ruido no afecta a la extracción de

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

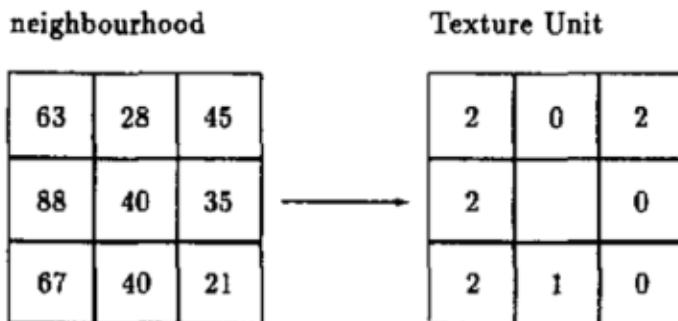


Figura 2.1: Ejemplo de unidad de textura generada mediante el patrón *LBP* sobre una matriz de vecindad 3×3 . Fuente: [Wang y He \(1990\)](#)

texturas de una imagen mediante *LBP*.

Considerando una unidad de textura con 8 elementos, cada uno de los cuales puede tomar uno de tres posibles valores, existen un total de $3^8 = 6561$ unidades de textura diferentes. Si una unidad de textura representa la textura de una región, las estadísticas de todas las unidades de textura definen la información de textura completa de una imagen. La distribución de ocurrencias de las unidades de textura se denomina “*Texture Spectrum*” o Espectro de Texturas. La Figura 2.2, también de [Wang y He \(1990\)](#), nos ofrece un ejemplo de un espectro de textura que caracteriza una imagen “A” determinada. Los autores del artículo concluyen que se pueden resolver tareas de clasificación de texturas de una imagen mediante el procesamiento de la información presente en el espectro de textura.

[Shahriar y cols. \(2013\)](#), en su artículo “*Fault diagnosis of induction motors utilizing local binary pattern-based texture analysis*” (“Diagnóstico de fallos de motores de inducción utilizando el analizador de texturas - Patrón Binario Local”), describe que la tarea de predecir anomalías en motores de inducción es complicada debido al ruido presente en la señal generada por los componentes mecánicos circundantes. Proponen un enfoque de trabajo que emplea un análisis bidimensional mediante *LBP*. En primer lugar, las señales del motor se convierten a imágenes en escala de grises, y luego se aplica *LBP* para discriminar las texturas de estas imágenes. El resultado se introduce a una máquina de soporte vectorial (*SVM*) para identificar posibles fallos.

En el trabajo se incluye un método de partición de la serie temporal de una señal en

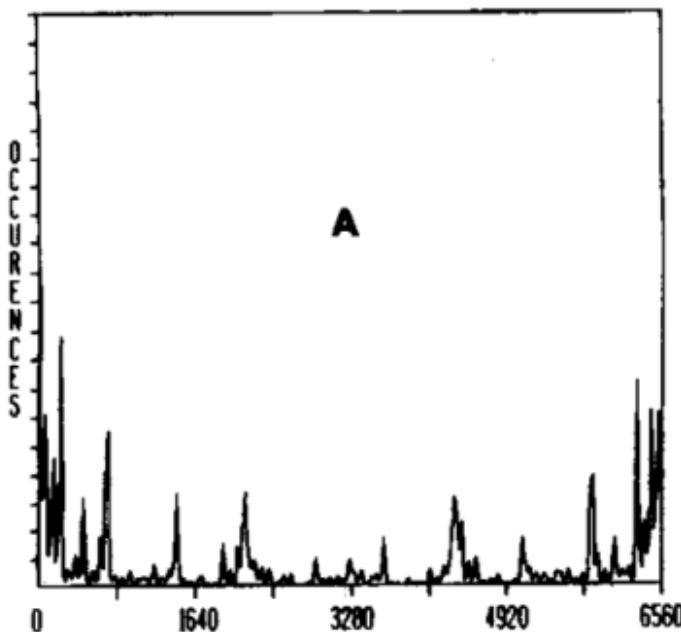


Figura 2.2: Ejemplo de Espectro de Textura de una imagen "A". Fuente: Wang y He (1990)

sucesivas muestras con N valores discretos (en DL , a cada dato de una serie temporal se le denomina *timestamp*), dando lugar a M muestras de tamaño N . Cada una de las M muestras forman una fila en la matriz, resultando en un tamaño $N \times M$. En la Figura 2.3, se representa la partición de una señal en una matriz que cumple la función de imagen.

En el contexto de la detección temprana de anomalías, se enfatiza la importancia de seleccionar una longitud de tiempo t lo suficientemente extensa para cada muestra. Esto permite que cada muestra de la señal pueda contener posibles anomalías. No obstante, se advierte que un valor extremadamente grande t podría disminuir la capacidad del sistema para detectar anomalías de manera temprana porque a medida que aumenta el tamaño de la señal, el modelo entrenado requerirá más información del pasado para prever el comportamiento futuro. Esto va en contra de la premisa inicial de lograr una detección temprana. Describen que el valor mínimo que debe tomar t es $t_{\min} = \frac{2}{f_{\min}}$, donde f_{\min} es la frecuencia de vibración más baja de la

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

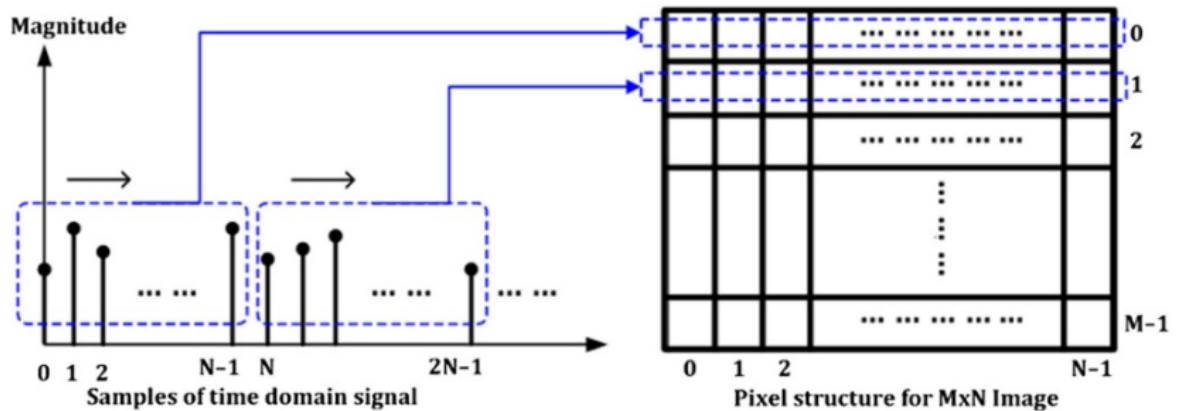


Figura 2.3: Conversión de una señal a una imagen de tamaño $M \times N$. Fuente: [Shahriar y cols. \(2013\)](#)

*Transformada de Fourier*². Así se garantiza que al menos una imagen contenga información correspondiente a dos ciclos completos de la señal.

Sobre la imagen generada de la señal, aplican el patrón *LBP*. Shahriar y cols. (2013) ofrece una definición formal del funcionamiento del patrón *LBP* que clasifica cada píxel de una imagen mediante la comparación de los valores de intensidad de sus P vecinos con el valor del centro, y convierte el resultado en un patrón de código que se puede representar en decimal. La definición formal viene dada por la expresión de la Ecuación 2.1:

$$\text{LBP}_{P,R}(x_c, y_c) = \sum_{p=0}^{P-1} s(g_p - g_c) 2^p \quad (2.1)$$

g_p es el valor gris del píxel P vecino del central c y g_c es el valor gris del píxel central de la región que se está analizando.

La Figura 2.4, realizada por Shahriar y cols. (2013), describe visualmente el cálculo de la unidad de textura. Los píxeles vecinos ($P = 8$) se definen con los valores [85, 99, 21, 86, 13, 12, 57, 54]. El píxel central c toma el valor 54. Para cada píxel vecino p_i , si $p_i \geq c$, entonces se inserta el valor 1 en la matriz intermedia llamada “*Threshold*”; en caso contrario, se inserta un 0. El código binario resultante se puede representar en

²**Transformada de Fourier:** es una operación matemática que convierte una función del tiempo, una variable continua, en una función de la frecuencia, revelando las frecuencias que componen la señal original.

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

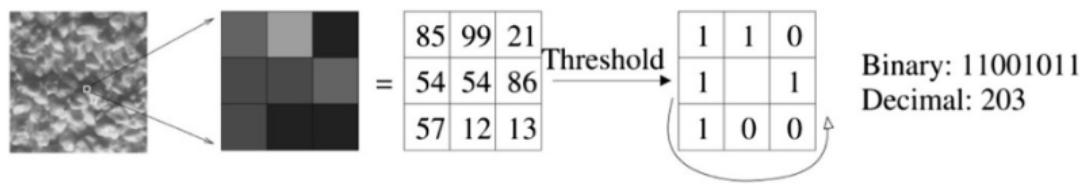


Figura 2.4: Cálculo numérico de una unidad de textura a partir de una matriz de entrada 3×3 . Fuente: [Shahriar y cols. \(2013\)](#)

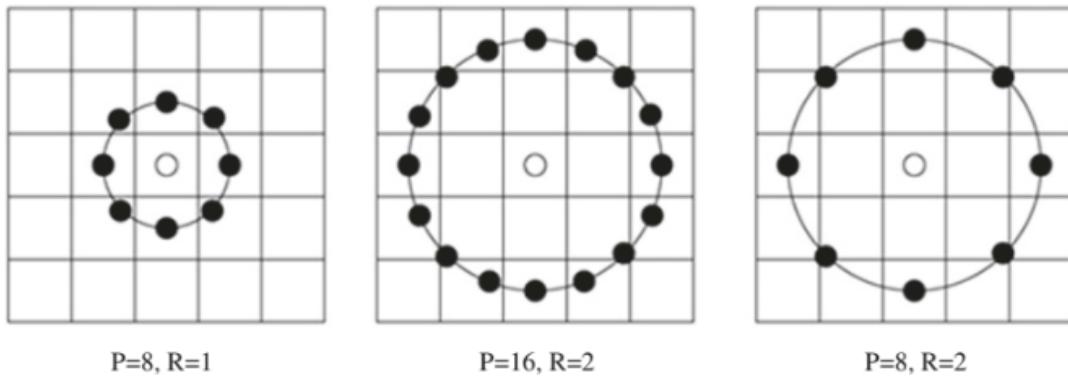


Figura 2.5: Configuración de parámetros de 3 operadores *LBP* diferentes. Fuente: [Shahriar y cols. \(2013\)](#)

formato decimal.

[Shahriar y cols. \(2013\)](#) describen también varias configuraciones de *LBP* para evaluar su rendimiento en el sistema de predicción de fallos que proponen. La Figura 2.5 contiene tres configuraciones diferentes que dependen de la longitud que puede tomar cada patrón codificado, representada por el parámetro P ; así como la distancia entre los píxeles vecinos P y el píxel central.

Algunas ventajas que ofrece el patrón *LBP* son las siguientes:

- Permite describir la información de textura de una vecindad local en relación a su píxel central, localizando patrones de texturas locales.
- En la definición formal, $g_p - g_c$ no se ve afectado por cambios en la luminosidad. Es invariante al escalado de grises porque lo crucial es el signo positivo o negativo de la diferencia de valores, no los valores exactos en sí.

Una vez se computa *LBP* para cada píxel, la imagen de entrada de tamaño $M \times N$

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

queda representada por un histograma H ; es el descriptor LBP de esa imagen. En el artículo, formalmente describen el descriptor H con la forma de la Ecuación 2.2:

$$H(\tau) = \sum_{y_c=1}^{M-1} \sum_{x_c=1}^{N-1} f(LBP_{P,R}(x_c, y_c), \tau) \quad (2.2)$$

Los descriptores LBP se utilizan como entrada en un algoritmo de clasificación conocido como “*Support Vector Machine*” (*SVM*) o Máquina de Vectores Soporte. Las *SVM* son un tipo de algoritmo de aprendizaje supervisado que se fundamenta en el concepto de hiperplano. Estos algoritmos representan los puntos de las muestras en un espacio vectorial y buscan definir un hiperplano que logre la mejor separación entre las clases en dicho espacio. Pertenece a la categoría de *métodos Kernel*³ y su propósito es facilitar la separación entre las clases al convertir los límites no lineales en lineales. La estrategia consiste en aprovechar un clasificador binario *SVM*, el cual aprenderá a discriminar cada tipo de anomalía respecto al resto de casos de fallos.

[Ruiz y cols. \(2018\)](#), en su artículo “*Wind turbine fault detection and classification by means of image texture analysis*” (“Detección y clasificación de fallos en turbinas eólicas mediante técnicas de análisis de texturas”), proponen una metodología que se fundamenta del trabajo publicado de [Shahriar y cols. \(2013\)](#). La Figura 2.6, extraída de su trabajo, muestra un esquema de la metodología presente. Inicia con un conjunto de datos recuperados por un sistema simulado, el cual se transforma en series temporales de datos mediante técnicas de procesamiento de señales. Estas series se convierten en imágenes grises. Posteriormente, se miden 13 variables del entorno simulado, generando así 13 imágenes de señales por muestra. A estas imágenes se aplica un extractor de características de textura, extrayendo 65 características por cada una.

El trabajo expone dos aspectos técnicos cruciales para combatir el sobreajuste del modelo de predicción. En primer lugar, la “*K-fold validation*” o validación cruzada (*K*), un método de validación que consiste en dividir en *K* unidades el conjunto de datos. Durante cada iteración de la validación cruzada, una partición se emplea para validar

³**Métodos Kernel:** son técnicas que permiten aplicar algoritmos lineales en espacios de características transformadas de manera no lineal. Son útiles cuando los datos no son linealmente separables pero pueden ser separados en un espacio de características de dimensión mayor.

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

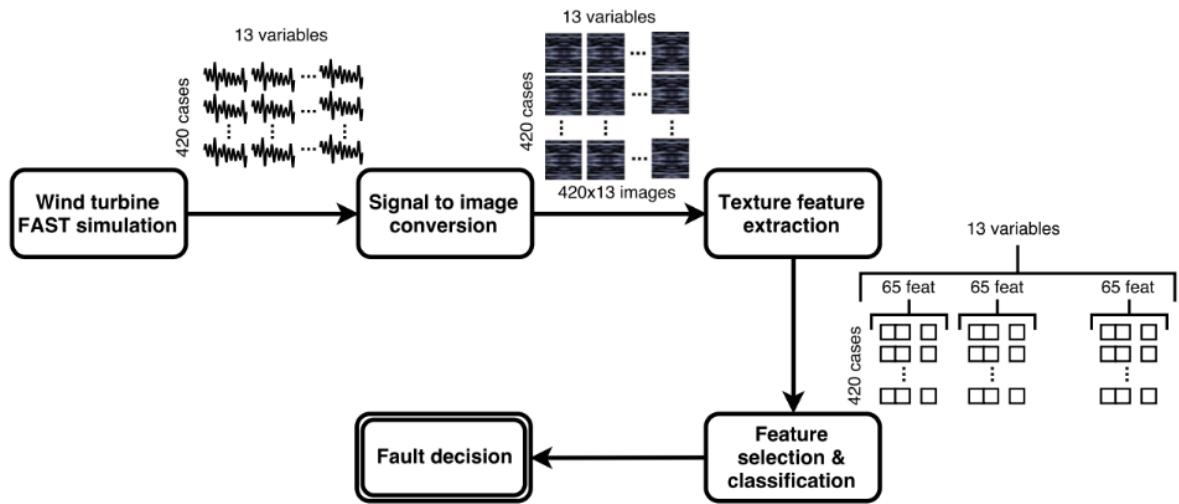


Figura 2.6: Metodología propuesta en la tarea de detección y clasificación de anomalías en turbinas eólicas. Fuente: [Ruiz y cols. \(2018\)](#)

el modelo y el resto de particiones se usan para su entrenamiento. Un ciclo completo de validación cruzada se completa cuando el modelo ha sido entrenado y evaluado por todas las K particiones. La idea general de la validación cruzada es evaluar la capacidad de generalización del modelo en diferentes particiones del conjunto de datos, evitando validar en cada iteración del entrenamiento con el mismo conjunto de validación. En la siguiente Figura 2.7 se visualiza una iteración completa de validación cruzada de K iteraciones con $K = 4$.

El segundo aspecto técnico es la selección de características, una fase que selecciona las características de textura más importantes que deben de ser interpretadas por el algoritmo de clasificación. Es importante reducir la dimensionalidad de los datos de entrada para que el modelo converja más rápido y obtenga un mejor resultado, seleccionando aquellas características con mayor relevancia. En el trabajo [Ruiz y cols. \(2018\)](#), se utiliza la técnica “*Maximum Relevance Minimum Redundancy*” (*MRMR*), traducida como “máxima relevancia, mínima redundancia”. Este enfoque se emplea para seleccionar un conjunto de características que contenga la mayor información posible con la menor redundancia. Busca mantener un conjunto diverso de características que ofrezcan información única.

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

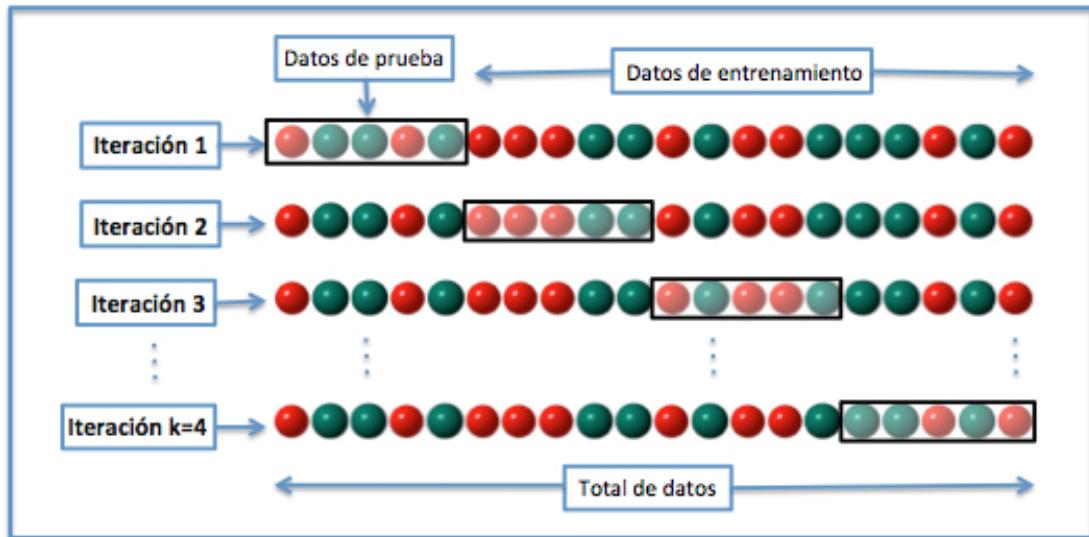


Figura 2.7: Validación cruzada en K iteraciones con $K = 4$. Fuente: [Wikipedia \(2024\)](#)

Ruiz y cols. (2018) concluyen presentando cinco algoritmos de clasificación para evaluar cuál ofrece mejores resultados: *K-Nearest*, *LDA* (*Linear Discriminant Analysis*), *Tree* (Árbol), *Bag Tree* (Árbol de bolsas), *LSVM* (*Linear Support Vector Machine*), y se comparan sus rendimientos en base a dos métricas, exactitud⁴ (*accuracy*) y el área bajo la curva *ROC*⁵.

Soother y cols. (2021) publican en su investigación titulada “*A Novel Method Based on UNET for Bearing Fault Diagnosis*” (“Un Nuevo Método basado en *UNET* para el Diagnóstico de Fallos en Rodamientos”) un modelo de diagnóstico de fallos en los rodamientos que procesa las imágenes de señales vibratorias mediante una arquitectura de red convolucional *U-Net*. Esta arquitectura aborda uno de los problemas principales presentes en este tipo de modelos de predicción de anomalías: el problema de etiquetado de la ventana deslizante. Cuando se convierten las señales a imágenes, la señal se divide en ventanas y algunas pueden presentar múltiples anomalías. De esta manera los modelos tenían dificultades para identificar el tipo de fallo. En el estudio, se explica que la arquitectura *U-Net* resuelve este problema mediante la clasificación de

⁴**Exactitud** (Apéndice A): es una métrica que mide la proporción de predicciones correctas con respecto al total de predicciones.

⁵**Curva ROC**: es una métrica que ofrece una información más relevante de la tarea de clasificación comparando los falsos positivos y negativos y los verdaderos positivos y negativos.

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

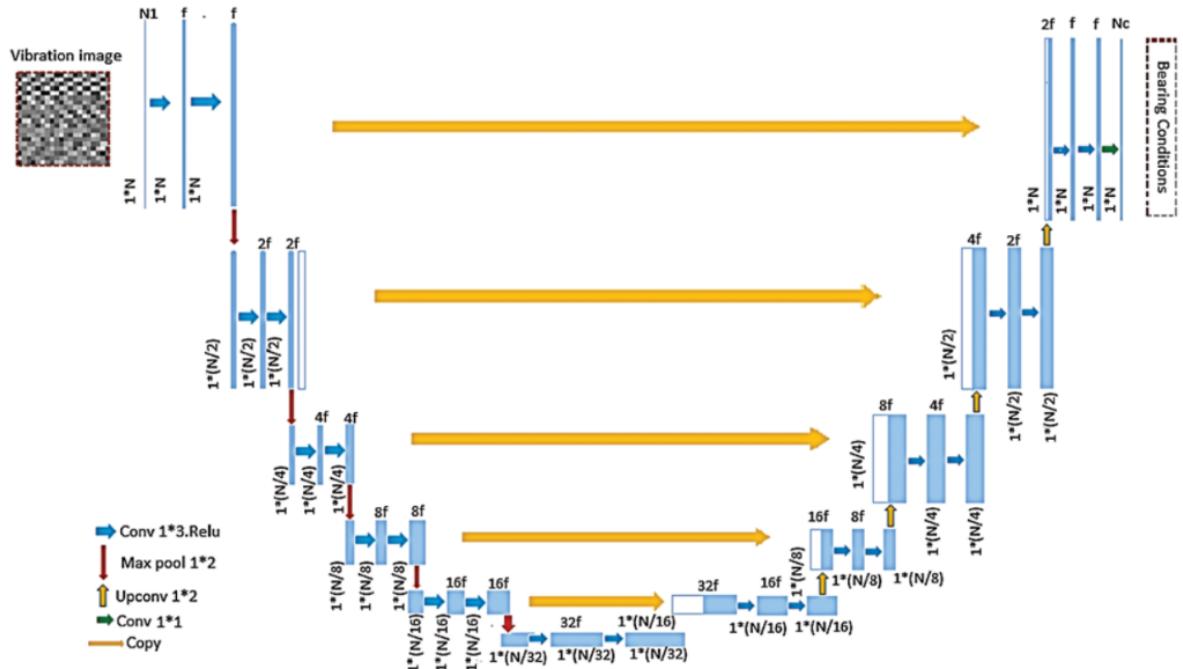


Figura 2.8: Arquitectura *U-Net*. Fuente: [Soother y cols. \(2021\)](#)

cada píxel de la imagen de la señal, lo que permite clasificar varias anomalías presentes en varios píxeles para una sola ventana de la señal.

La arquitectura *U-Net* propuesta por [Soother y cols. \(2021\)](#) queda reflejada en la Figura 2.8, y en ella se puede observar que está formada por dos fases claramente diferenciadas. La fase de contracción o “*contracting path*” (la mitad izquierda de la red) está formada por múltiples capas convolucionales y de *max-pooling* que se encargan de captar las características de la imagen de entrada. En los estudios mencionados anteriormente, la función de esta fase es análoga a la del operador de texturas *LBP*. La fase de expansión o “*expanding path*” (correspondiente a la mitad derecha de la red) incluye capas de *up-sampling* para mantener la resolución de la imagen de salida segmentada similar a la imagen de entrada. En el [Capítulo 5. Implementación y desarrollo de la arquitectura *U-Net + LSTM*](#) se detalla el origen y el funcionamiento interno de esta arquitectura.

La investigación de [Soother y cols. \(2021\)](#) expone unos resultados prometedores de la arquitectura *U-Net* en la detección de fallos en el ámbito de la energía eólica.

2.1. TRABAJOS PREVIOS EN SERIES TEMPORALES DE DATOS Y EN LA PREDICCIÓN DE FALLOS EN MOTORES

Indicator	UNET (%)	LeNet-5 (%)	ResNet-50 (%)	FCN (%)
Accuracy	98.91	96.74	95.43	97.61
F1-Score	99.00	96.80	95.50	97.60

Figura 2.9: Evaluaciones del modelo *U-Net* con otros modelos de clasificación mediante las métricas exactitud y *F1-Score*. Fuente: [Soother y cols. \(2021\)](#)

Se comparan los resultados obtenidos por *U-Net* y otros modelos de aprendizaje profundo propuestos en investigaciones recientes y entrenados con el mismo conjunto de datos. Estos modelos incluyen *FCN*⁶, *LeNet5*⁷, *ResNet-50*⁸, y fueron evaluados con las métricas de exactitud y *F1-Score*⁹. La Figura 2.9 refleja la comparativa de los resultados de estas métricas realizada por [Soother y cols. \(2021\)](#) para cada uno de los modelos. Destaca el resultado excepcional del modelo *U-Net* de una exactitud del 98.91 % y una *F1-Score* del 99 %.

En el contexto del presente trabajo, la arquitectura *U-Net* constituye la primera fase, que consiste en automatizar el etiquetado de las imágenes convertidas de señales sin etiquetar procedentes de la turbina eólica (véase el [Capítulo 5. Implementación y desarrollo de la arquitectura *U-Net + LSTM*](#) para entrar en detalle de la arquitectura empleada). La creación y configuración de un modelo de aprendizaje profundo es un arte en el que existen numerosas opciones a considerar. Se parte de la arquitectura *U-Net* proporcionada por el trabajo previo de [Cambero Rojas \(2023\)](#), que propone un modelo *U-Net* con un resultado óptimo, el cual se basa, a su vez, en el trabajo previo realizado por [Delgado Muñoz \(2022\)](#), que aplica por primera vez un modelo *U-Net* sobre el conjunto de datos del proyecto *Anemoi*.

⁶**FCN(Full Convolutional Network)**: es una arquitectura de RN diseñada para tareas de segmentación en imágenes. A diferencia de las redes convencionales, *FCN* elimina las capas completamente conectadas, permitiendo la entrada de imágenes de cualquier tamaño y produciendo salidas de igual dimensión.

⁷**LeNet5**: fue una de las primeras redes convolucionales propuesta por [LeCun y cols. \(1998\)](#) y es una red convolucional simple.

⁸**ResNet-50**: es una arquitectura de RN convolucional introducida por *Microsoft Research* en 2015. Se caracteriza por su estructura profunda con conexiones residuales.

⁹**F1-Score (Apéndice A)**: es una medida que combina la precisión y la exhaustividad (recall) en un solo valor. La precisión se refiere a la proporción de instancias positivas correctamente clasificadas entre todas las instancias positivas. La exhaustividad mide la misma proporción respecto al total de instancias que realmente son positivas.

2.2. TRABAJOS PREVIOS EN EL RECONOCIMIENTO DE ACCIONES HUMANAS EN VÍDEOS

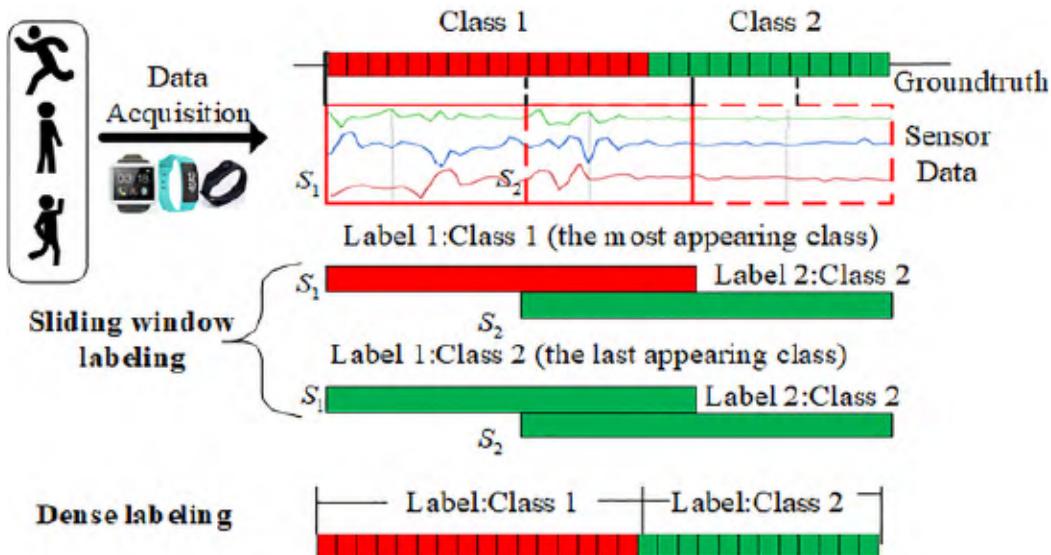


Figura 2.10: Etiquetado mediante ventana deslizante vs etiquetado denso. Fuente: [Zhang y cols. \(2019\)](#)

2.2. Trabajos previos en el reconocimiento de acciones humanas en vídeos

Los estudios vistos en la sección anterior extraen patrones clave a través del tiempo de las señales mediante la conversión de las señales en imágenes. Sin embargo, durante el resto del entramado de los sistemas propuestos, no se incluyen mecanismos de atención que puedan capturar información relevante en las series temporales de datos.

A continuación, se explican dos estudios relacionados con la disciplina “*Human Activity Recognition*” o *HAR* (Reconocimiento de Acciones Humanas) sobre los cuales se construye el presente trabajo.

[Zhang y cols. \(2019\)](#), en su investigación “*Human Activity Recognition Based on Motion Sensor Using U-Net*” (“Reconocimiento de actividades humanas basadas en un sensor de movimiento usando *U-Net*”), proponen un sistema de reconocimiento de acciones humanas mediante la arquitectura *U-Net* aplicada sobre vídeos, un tipo de serie temporal de imágenes. Describen una técnica de etiquetado denominada “*dense labeling*”, o etiquetado denso, para solucionar el problema de etiquetado de la ventana deslizante, comentado en el apartado anterior. La Figura 2.10, proveniente de esta

2.2. TRABAJOS PREVIOS EN EL RECONOCIMIENTO DE ACCIONES HUMANAS EN VÍDEOS

Dataset	Indicator	SVM	DT	LSTM	CNN	CovLSTM	FCN	SegNet	Mask R-CNN	U-Net
WISDM	Acc	0.813	0.853	0.938	0.941	0.948	0.879	0.957	0.862	0.964
	Fw	0.848	0.850	0.937	0.941	0.947	0.881	0.957	0.863	0.965
UCI HAPT	Acc	0.918	0.807	0.920	0.903	0.893	0.912	0.915	0.908	0.921
	Fw	0.922	0.806	0.917	0.901	0.889	0.912	0.913	0.906	0.922
OPP Gesture	Acc	0.785	0.700	0.8316	0.831	0.817	0.911	0.914	0.908	0.947
	Fw	0.827	0.698	0.8414	0.934	0.818	0.910	0.912	0.907	0.947
Sanitation	Acc	0.817	0.803	0.8024	0.864	0.825	0.640	0.846	0.697	0.889
	Fw	0.820	0.804	0.8028	0.863	0.823	0.637	0.846	0.696	0.889

Figura 2.11: Resultados de la exactitud y *f1-score* sobre la evaluación del modelo *U-Net*. Fuente: [Zhang y cols. \(2019\)](#)

investigación, ilustra la diferencia entre el etiquetado denso y el etiquetado mediante ventana deslizante. La idea que proponen consiste en etiquetar cada píxel de manera independiente al resto de píxeles. El aspecto temporal de la serie de datos se encuentra en la secuencia de imágenes o *frames* de un vídeo.

Este informe implementa por primera vez la arquitectura *U-Net* para el reconocimiento de acciones humanas en vídeos. Dada la novedad de esta implementación, es necesario comparar los resultados de otros modelos destacados en la misma tarea. La Figura 2.11 presenta una comparación de las métricas *accuracy* y *f1-score* entre varios modelos realizada por [Zhang y cols. \(2019\)](#), destacándose la arquitectura *U-Net* por obtener unos resultados superiores a las expectativas.

[Orozco y cols. \(2021\)](#), en su artículo “*CNN-LSTM* con Mecanismo de Atención suave para el Reconocimiento de Acciones Humanas en Vídeos” publica una arquitectura *CNN-LSTM* para el reconocimiento de acciones humanas en vídeos. Las Redes Neuronales Convolucionales 3D (*CNN*) no solo son capaces de capturar características espaciales en imágenes, sino también capturar la evolución temporal entre fotogramas consecutivos. Las Redes Neuronales Recurrentes, como lo son las *LSTM* (*Long Short Term Memory*), aportan un mecanismo de atención exitoso sobre una serie temporal de datos. Estas redes se utilizan para aprender dinámicas temporales complejas. Para obtener más detalles sobre las redes neuronales *CNN* y *LSTM*, se

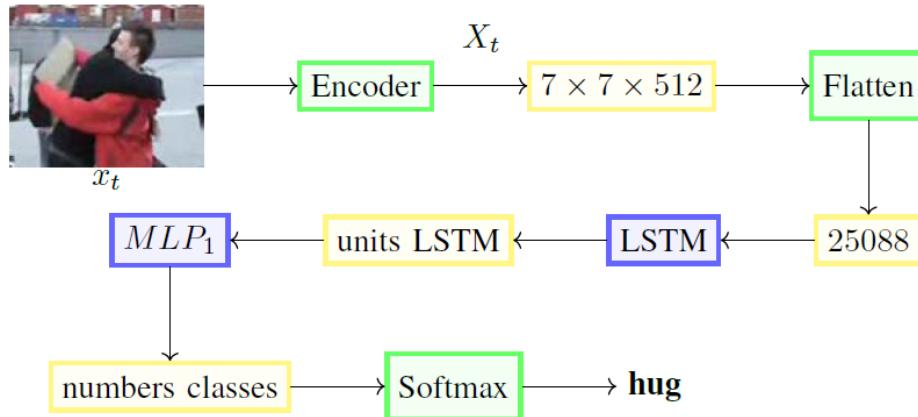


Figura 2.12: Arquitectura base *LSTM-CNN*. Fuente: [Orozco y cols. \(2021\)](#)

remite al [Apéndice A](#) de esta memoria.

La Figura 2.12, extraída del trabajo de [Orozco y cols. \(2021\)](#), presenta un sistema *HAR* base formado por: (i) Fase de entrada, donde el vídeo se normaliza a un total de 40 fotogramas; (ii) Fase *CNN*, en la que una red preentrenada *VGG16* extrae las características del vídeo; y (iii) Fase *LSTM* junto con una capa densa con un nodo por cada clase para la clasificación final.

Sobre la arquitectura *CNN-LSTM*, base de la figura anterior, se implementa un nuevo mecanismo de atención aplicado a la salida del *Encoder* (Codificador), tomando como base la naturaleza de las unidades *LSTM*. La Figura 2.13, también de [Orozco y cols. \(2021\)](#), ilustra el enfoque base con un nuevo mecanismo de atención, que tiene como objetivo focalizar una mayor atención en una parte específica de la imagen relacionada con la actividad humana a predecir. De esta forma, se facilita a las posteriores capas *LSTM* la tarea de predecir la acción humana sobre la dimensión temporal del resto de *frames*.

El estudio concluye con una comparación de la métrica de precisión (*precision*) entre el modelo base y el modelo que incluye el mecanismo de atención. Si observamos la Figura 2.14, en su trabajo, [Orozco y cols. \(2021\)](#) muestran cómo la segunda aproximación con el mecanismo de atención logra una precisión más elevada, con valores de 47,14 % y 87,33 %, en comparación con el modelo base.

2.2. TRABAJOS PREVIOS EN EL RECONOCIMIENTO DE ACCIONES HUMANAS EN VÍDEOS

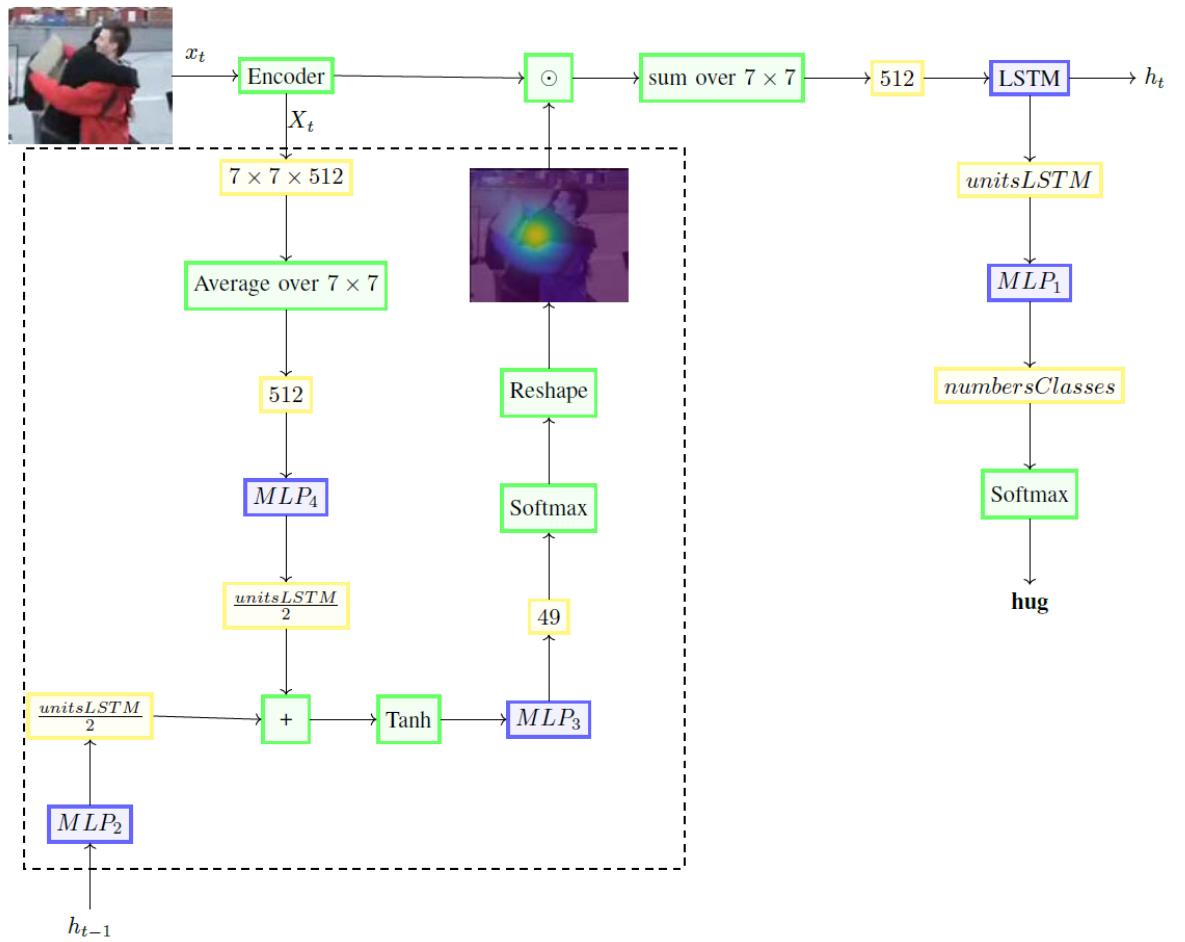


Figura 2.13: Arquitectura con mecanismo de atención. Fuente: Orozco y cols. (2021)

		Precision	Recall
HMDB-51	Enfoque base	39,56 %	41,67 %
	Enfoque con atención	47,14 %	48,21 %
UCF-101	Enfoque base	71,94 %	72,02 %
	Enfoque con atención	87,33 %	89,97 %

Figura 2.14: Resultados para la métrica de precisión de los modelos base y con atención. Fuente: Orozco y cols. (2021)

El presente TFG emplea las diversas técnicas asociadas a los trabajos mencionados en el actual apartado. Es de suma importancia destacar que el enfoque principal y punto de partida más significativo se centra en la combinación de los estudios Zhang y cols. (2019) y Orozco y cols. (2021) para implementar un mecanismo de atención con redes



2.2. TRABAJOS PREVIOS EN EL RECONOCIMIENTO DE ACCIONES HUMANAS EN VÍDEOS

neuronales *LSTM* sobre la arquitectura convolucional *U-Net*. Esta implementación se detalla más adelante en el [Capítulo 5. Implementación y desarrollo de la arquitectura *U-Net + LSTM*](#)

Capítulo 3

Entorno de trabajo

En este capítulo se describe el entorno de trabajo en el que se desarrolla el presente proyecto. Se exponen las especificaciones *hardware* y el *software* empleado en el Capítulo 5. Implementación y desarrollo de la arquitectura *U-Net + LSTM*

3.1. *Hardware*

Para la implementación del proyecto se utilizó la máquina propia del autor. El computador es un *HP* basado en *x64* de 2014, con un procesador *Intel^(R) Core^(TM) i7-4790S* CPU @ 3.20 GHz, una memoria *RAM* de 16 GB, un disco duro de estado sólido *SSD* de 465 GB, un disco *HDD* de 2.7 TB y una tarjeta gráfica *NVIDIA GeForce GTX 1050 Ti*.

Sin embargo, debido a las limitaciones de *hardware* en el procesamiento de secuencia de imágenes, se optó por complementar el trabajo utilizando *Google Colaboratory*¹. Se prefirió la infraestructura de *Google Colaboratory* debido a la disponibilidad de la *GPU T4*, la cual facilita significativamente el procesamiento de imágenes. Es importante destacar que, aunque *Google Colaboratory* ofrece recursos adicionales, también tiene sus propias limitaciones de tiempo de uso.

¹**Google Colaboratory:** es un servicio de Google alojado de *Jupyter Notebook* que no requiere configuración y permite ejecutar código *Python*, adecuado para aprendizaje automático y ciencia de datos.

3.2. SOFTWARE

3.2. Software

Se trabaja con el lenguaje de programación de *Python*². Se selecciona este lenguaje porque incluye dos de las librerías más populares en la creación de redes neuronales (RN): *TensorFlow* y *Keras*.

La máquina local se utiliza para ejecutar *scripts* con prototipos de modelos para evaluar si sus arquitecturas son correctas. Para ello, se procede a instalar *Anaconda* con la versión 3.12.0 de *Python*. *Anaconda* es una distribución de *Python* que ofrece una serie de herramientas útiles en ciencia de datos, análisis de los datos y desarrollo de aplicaciones que trabajan con datos. Incluye un conjunto de bibliotecas instaladas que son cruciales en el desarrollo de proyectos de análisis de datos. Además, incluye un gestor de paquetes “*conda*” que permite gestionar paquetes *software* fácilmente. También te permite crear entornos virtuales de *Python* para preparar un entorno de trabajo las dependencias y paquetes de forma aislada. La instalación de las dependencias y librerías necesarias para la consecución de este proyecto se instalan manualmente en el *software* de *Anaconda*.

El entorno de *Google Colaboratory* contiene multitud de paquetes de *software* y librerías preinstalados. A continuación, se listan las librerías de *Python* más destacables que se han empleado:

- **Pandas**³: biblioteca de software de código abierto que recoge las funciones esenciales para la manipulación y análisis de datos en *Python*. Esta biblioteca fue creada como una herramienta de alto nivel para el análisis y manipulación de datos. Es una de las bibliotecas más populares en el campo de la *Ciencia de Datos* en *Python*.
- **Numpy**⁴: esta biblioteca se utiliza para cálculos matemáticos y científicos. Ofrece un conjunto de operaciones sobre objetos *arrays*, los cuales son más

²*Python*: es un lenguaje de programación de alto nivel, orientado a objetos que se moldea bien para tareas de ciencia de datos: preprocessado de datos, visualización, aprendizaje automático, profundo, etc

³**Pandas**: Página Oficial: <https://pandas.pydata.org>

⁴**Numpy**: Página Oficial: <https://numpy.org>

3.2. SOFTWARE

flexibles que las listas por defecto de *Python*. Esta biblioteca fue creada en 2005 y es una de las dependencias más destacadas en la manipulación y preprocesamiento de datos.

- **Scikit-learn (Sklearn)**⁵: biblioteca de aprendizaje automático de código abierto diseñada para *Python*. Proporciona herramientas simples y eficientes para el análisis de datos: algoritmos de clasificación, regresión, *clustering* y reducción de dimensionalidad, así como herramientas para la evaluación de modelos y preprocesamiento de datos.
- **Matplotlib**⁶: biblioteca de trazado de gráficos de bajo nivel en *Python*. Permite visualizar datos almacenados en objetos *arrays* de la librería *Numpy* y proporciona muchas funciones para personalizar la apariencia de los gráficos.
- **TensorFlow**⁷: biblioteca de código abierto desarrollada por Google para el aprendizaje automático, diseñada para construir modelos de redes neuronales. *TensorFlow* ofrece varios niveles de abstracción. La compilación y entrenamiento de modelos mediante la *API* de alto nivel de *Keras* ayuda a que comenzar a usar *TensorFlow* y el aprendizaje automático sea sencillo.
- **Keras**⁸: *API* de alto nivel de *TensorFlow* para construir y entrenar modelos de aprendizaje profundo. Ofrece a los científicos de datos una vía de alto nivel más cómoda para construir redes neuronales sin tener que preocuparse por aspectos de implementación de bajo nivel o el funcionamiento interno de las redes neuronales.

⁵**Sklearn**: Página Oficial: <https://scikit-learn.org/stable/>

⁶**Matplotlib**: Página Oficial: <https://matplotlib.org>

⁷**TensorFlow**: Página Oficial: <https://www.tensorflow.org>

⁸**Keras**: Página Oficial: <https://keras.io>



3.2. SOFTWARE

Capítulo 4

Conocimiento del dominio físico y de los datos

Antes de adentrarnos en la etapa de implementación, este capítulo tiene como objetivo proporcionar al lector el contexto necesario para comprender de manera satisfactoria el resto del documento.

En primer lugar, en el apartado **4.1 Entorno físico** se presentan los actores clave en el proceso de obtención de datos, abordando dos preguntas fundamentales: ¿cómo opera internamente una turbina eólica? y ¿cuál es la procedencia de los datos? Posteriormente, en el apartado **4.2 Análisis de los datos**, se profundiza en la tarea de análisis de la fuente de datos capturados por el sistema *SCADA*, marcando una fase crucial en el ámbito de la Ciencia de Datos¹, previa al inicio de la implementación.

4.1. Entorno físico

En concordancia al propósito de este trabajo, que consiste en proporcionar un sistema de detección de anomalías en turbinas eólicas, el presente apartado se centra en explicar los componentes básicos de una turbina eólica y su naturaleza física.

¹**Ciencia de datos:** es el campo de estudio que combina la experiencia en un dominio específico, habilidades de programación y conocimientos en matemáticas y estadísticas para extraer conocimientos significativos a partir de los datos.

4.1. ENTORNO FÍSICO

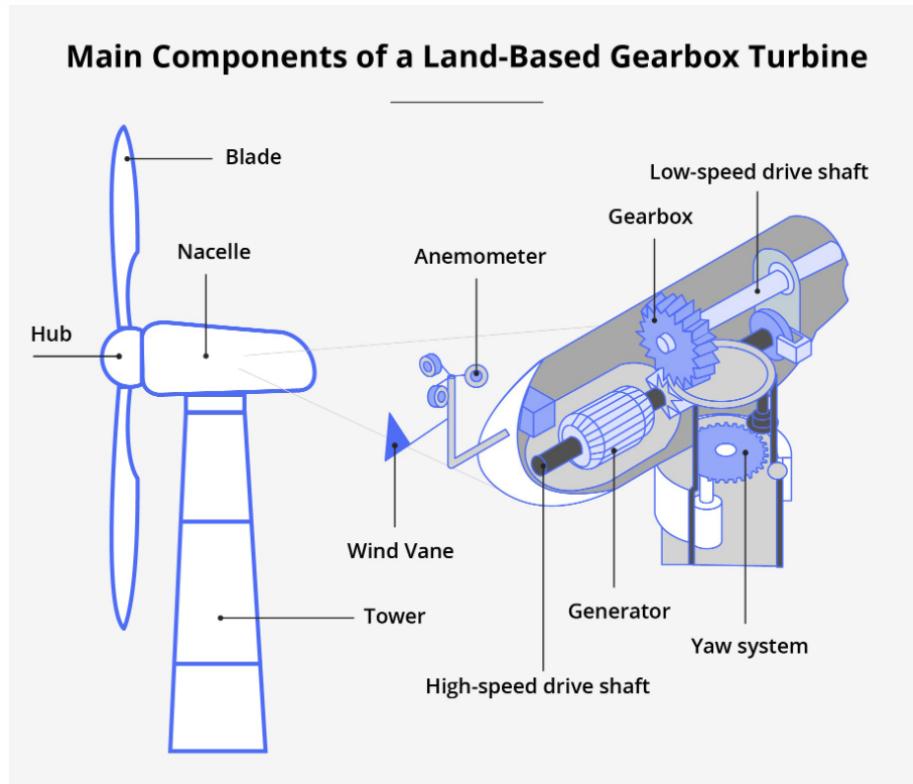


Figura 4.1: Principales componentes de una turbina eólica. Fuente: [Tyme \(2023\)](#)

4.1.1. Partes de una turbina eólica

Los datos empleados en este proyecto, en manos del *Centro Extremeño de Investigación, Innovación Tecnológica y Supercomputación, CénitS*, de la fundación *COMPUTAEX*, y provenientes de la empresa energética *Naturgy*, fueron recopilados en un parque de turbinas eólicas *onshore* (en tierra) ubicado en el suroeste de España, desde el año 2009 hasta el 2017. El conjunto de datos, denominado *dataset*, consta de 132 variables y cada una mide una característica en un momento específico. Estas características son registradas por un conjunto de sensores dispuestos en diversas partes de la turbina eólica cada 10 minutos, generando así una nueva marca de tiempo o *timestamp* que se almacena en la serie temporal. El comportamiento de estas máquinas se supervisa mediante un sistema *SCADA*, el cuál será abordado en detalle en el siguiente apartado 4.1.2.

La turbina eólica es una máquina que consta de una multitud de piezas. La Figura

4.1. ENTORNO FÍSICO

4.1, obtenida de [Tyme \(2023\)](#), muestra los componentes principales, que detallaremos para obtener una comprensión general de la estructura:

- **Torre (Tower):** permite al rotor acceder a altas velocidades del viento a una gran altura respecto al nivel de la tierra. Están construidas con un acero tubular para soportar con éxito el peso de la góndola, cubo y las palas.
- **Góndola (Nacelle):** la góndola alberga los componentes mecánicos esenciales para transformar la energía cinética generada por el movimiento del rotor, impulsado por las palas, en energía eléctrica. Aunque se puede percibir como pequeña desde la distancia, su peso puede alcanzar las 50 toneladas.
- **Árboles de transmisión (Drive shafts):** constituyen los elementos fundamentales del tren de transmisión. El árbol de transmisión de baja velocidad (*low-speed drive shaft*) es un componente que vincula la caja de cambios con la caja (*hub*) de la turbina, siendo la parte del rotor que gira a la misma velocidad que las palas. Como se ha mencionado en el componente anterior, la clave reside en aumentar las revoluciones por minuto (rpm) de las palas para maximizar la producción de energía eléctrica. Por otro lado, el árbol de transmisión de alta velocidad (*high-speed drive shaft*) es más pequeño, rota a una velocidad mayor y establece la conexión entre la caja de cambios y el generador.
- **Caja de cambios (Gearbox):** el árbol de transmisión de baja velocidad (*low-speed drive shaft*) es un rotor que gira a una velocidad lenta similar a la velocidad rotacional de las palas, y la función de la caja de cambios es convertir esta velocidad de rotación lenta en una velocidad alta necesaria para accionar el generador.
- **Generador (Generator):** es la parte más importante de una turbina eólica, ya que convierte la energía cinética del movimiento del rotor en energía eléctrica

4.1. ENTORNO FÍSICO

mediante las leyes de la electrodinámica².

- **Caja (Hub)**: es la parte del rotor que une las palas con el árbol de transmisión lento. Su peso ronda las 7 - 14 toneladas para sostener el peso de las palas en movimiento.
- **Sistema de orientación (Yaw system)**: ubicado debajo de la góndola, este sistema es responsable de alinear la turbina conforme la dirección del viento, la cual viene determinada por la veleta y anemómetro. Su función consiste en ajustar la góndola para capturar la energía cinética de la manera más efectiva posible.
- **Palas (Blades)**: son grandes y ligeras para resistir rachas altas de viento. Se ajustan por el ángulo de *pitch* o *pitch system*, que optimizan la velocidad del rotor para evitar daños en la turbina causados por fuertes rachas de viento.

4.1.2. Sistema SCADA de captura de datos

Un sistema *SCADA*, siglas en inglés de *Sistema de Supervisión, Control y Adquisición de Datos (Supervisory Control And Data Acquisition)*, es un sistema *software* y *hardware* diseñado para controlar procesos industriales, basados en la recopilación en tiempo real de datos procedentes de localizaciones remotas para controlar equipos y condiciones. Proporciona las herramientas necesarias a las organizaciones para tomar decisiones apoyadas en los datos procedentes de sus procesos industriales.

Este sistema ofrece diversas funciones, entre las que destacan:

- Control y supervisión de todos los sistemas remotos.
- Procesamiento de datos y generación de informes.
- Proporciona un sistema de alarmas con un registro de incidencias.

²**Electrodinámica**: es la rama de la física que estudia la relación entre la electricidad y el magnetismo y como ambas fuerzas interactúan entre sí.

4.1. ENTORNO FÍSICO

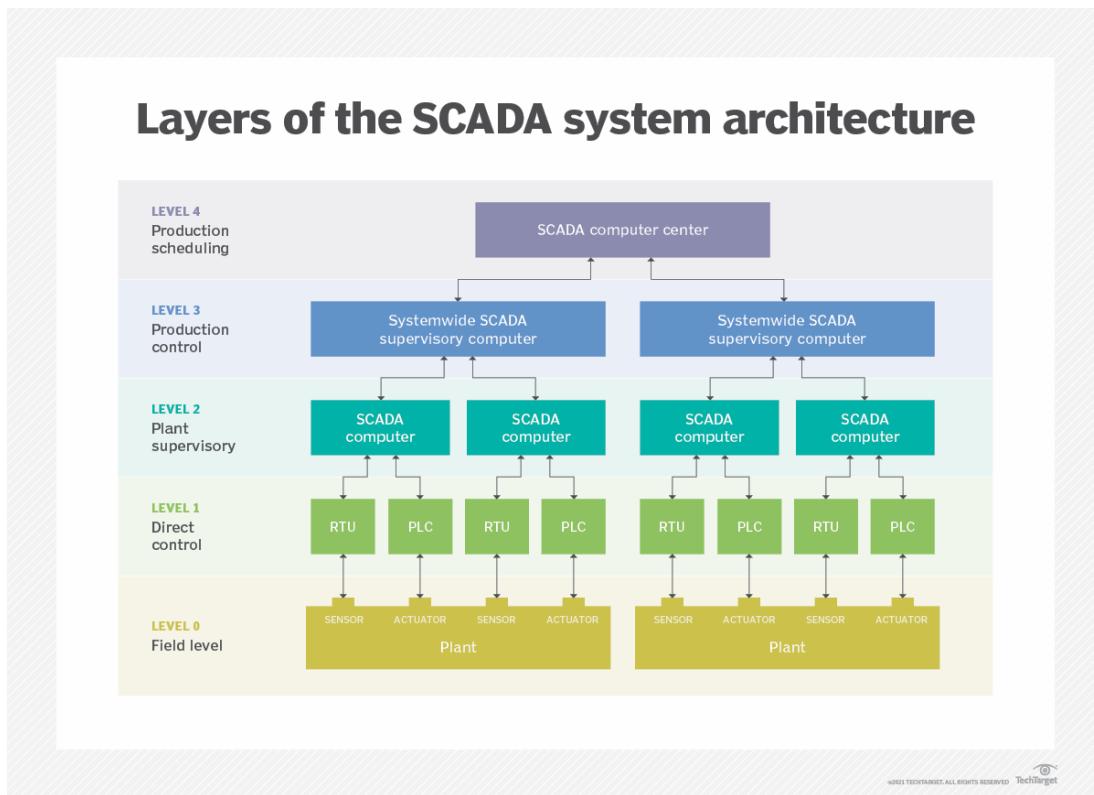


Figura 4.2: Arquitectura por niveles de un sistema SCADA. Fuente: [Wikipedia \(2022\)](#)

- Genera datos históricos sobre el funcionamiento de los sistemas remotos.
- Proporciona una interfaz *HMI*³, permitiendo la interacción entre el personal operador con la máquina.

La Figura 4.2 presenta una visualización por niveles de la arquitectura general de SCADA. En el nivel 0 (nivel de campo), se encuentran dos dispositivos de campo: los sensores, encargados de recopilar datos del proceso físico, y los actuadores, responsables de ejecutar acciones en el proceso físico.

El nivel 1 (nivel de control) integra unidades remotas *RTU*, *Remote Terminal Units* y las *Programmable Logic Controllers*, *PLC*. Las *RTU* conectan los objetos de los cuales se obtienen datos con los terminales remotos, mientras que los *PLC* son autómatas programables que realizan un control local de dispositivos y procesos, supervisando las operaciones en tiempo real.

³**HMI**: son las siglas de *Human-Machine Interface* (Interfaz Humano-Máquina) y se refieren a un panel que permite a un usuario comunicarse con una máquina, *software* o sistema.

4.1. ENTORNO FÍSICO

El nivel 2 (nivel de supervisión) supervisa y controla los *PLC* y unidades remotas. Reciben los datos del nivel inferior y envían planes de actuación para regular y optimizar el proceso industrial. Además, presenta una interfaz persona-máquina *HMI*.

En el nivel 3 (nivel de producción) se encuentran sistemas de gestión, los cuales utilizan información recopilada por el sistema *SCADA* destinados a producir informes o alertas al nivel superior de producción.

El nivel 4 o nivel de producción raramente se cita en sistemas *SCADA*. Está vinculado a un nivel superior que se centra en la interconexión de distintos procesos de negocio.

Las principales ventajas que ofrecen estos sistemas son:

- **Escalabilidad:** los sistemas *SCADA* son más escalables, proporcionan una mayor disponibilidad de los datos e integran la computación en la *nube* repartiendo las cargas de trabajo.
- **Interoperabilidad:** estos sistemas dependen de *software* y *hardware* propietarios, por lo tanto los clientes se vuelven dependientes del proveedor. Prevalece la flexibilidad y elección del proveedor y la mejora de la competencia.
- **Comunicaciones:** estos sistemas se apoyan en protocolos modernos de comunicación. Estos protocolos maximizan la accesibilidad a los datos y control de *SCADA*.

En la Tabla 4.1, que se presenta a continuación, se enumeran las variables más importantes recopiladas por los sensores del sistema *SCADA*, tal como se encuentran en el conjunto de datos proporcionado por el centro CénitS/COMPUTAEX. Además, la tabla proporciona las unidades de medida correspondientes a cada variable.

4.1. ENTORNO FÍSICO

Variable	Unidad de Medida
Potencia Eléctrica Activa	<i>kW</i>
Velocidad media del viento	<i>m/s</i>
Temperatura del rodamiento del engranaje	<i>°C</i>
Temperatura del Rodamiento del Aerogenerador	<i>°C</i>
Revoluciones por minuto del Generador	<i>rpm</i>
Revoluciones por minuto del Rotor	<i>rpm</i>
Ángulo de <i>pitch</i> de las Palas	°
Energía producida para la Red Eléctrica	<i>kW</i>
Dirección de la Góndola	°
Temperatura de la Góndola	<i>°C</i>

Tabla 4.1: Variables y unidades de medida asociadas.

4.1.3. Modelo físico de una turbina eólica

En esta sección se explica el modelo físico de una turbina eólica con el objetivo de vincular sus fundamentos físicos a la identificación de las variables más relevantes en el conjunto de datos. El propósito es seleccionar las variables físicas que tienen un mayor impacto en la predicción temprana de anomalías en una turbina eólica.

La función principal de una turbina eólica es aprovechar la fuerza del viento para generar electricidad. El modelo del sistema de conversión de energía eólica consta de una serie de pasos destinados a lograr la transformación de la energía cinética en energía eléctrica.

4.1. ENTORNO FÍSICO

La energía cinética se define como la energía adquirida por un cuerpo en movimiento. Se trata del trabajo necesario para acelerar un cuerpo de una masa determinada desde el reposo hasta una velocidad específica. Esta definición se expresa mediante la Ecuación 4.1, proporcionada por Rafaat y Hussein (2018) en su artículo “Maximización de potencia y control de un sistema de turbina eólica de velocidad variable mediante búsqueda de extremos”.

$$KE = \frac{1}{2}mV^2 \quad (4.1)$$

La energía cinética (KE , por sus siglas en inglés de *Kinetic Energy*) se expresa en julios (J). La masa del cuerpo en movimiento se representa como m y se mide en kilogramos (kg), mientras que la velocidad a la que se desplaza dicho cuerpo se denota como V y se mide en metros por segundo (m/s).

La energía cinética asociada al viento se convierte en energía mecánica a través del rotor de las palas. Esta energía mecánica se transmite mediante el tren de transmisión que conecta el rotor con el generador, y este último transforma la energía mecánica en energía eléctrica.

En teoría, la energía mecánica es la suma de la energía cinética y la energía potencial. La energía potencial se refiere a la energía asociada a la posición de un cuerpo en un campo de fuerza. Sin embargo, en el contexto de una turbina eólica, la energía potencial no suele estar directamente relacionada con la operación principal de la turbina, que es la conversión de la energía cinética del viento en energía mecánica y eléctrica. Por lo tanto, para simplificar los cálculos, la energía potencial no se tiene en cuenta al calcular la energía mecánica.

Cuando se evalúa la eficiencia del funcionamiento de una turbina eólica, es imprescindible tener en cuenta la variable del tiempo. La energía cinética se puede expresar en términos de potencia, que mide la rapidez con la que se realiza un trabajo o se transfiere la energía en el tiempo. Su cálculo viene dado por la Ecuación 4.2:

$$P = \frac{E}{t} \quad (4.2)$$

4.1. ENTORNO FÍSICO

P es la potencia, medida en vatios (W); E es la energía cinética, medida en julios (J); t es el tiempo, medido en segundos (s). Al combinar la Ecuación 4.1 y la Ecuación 4.2, la definición de potencia se expresa como la energía cinética por unidad de tiempo, Ecuación 4.3:

$$P = \frac{1}{2} \dot{m} V^2 = \frac{1}{2} \frac{dm}{dt} V^2 \quad (4.3)$$

donde:

- V es la velocidad del viento, medida en metros por segundo (m/s).
- m es la masa de aire que fluye perpendicularmente a las palas de la turbina eólica, generando un área circular. La masa se puede expresar de la siguiente manera: $m = \rho A V$

con:

- ρ la densidad del aire, medida en kilogramos por metro cúbico (kg/m^3).
- A el área de barrido de la pala, medida en metros cuadrados (m^2).
- V la velocidad del viento, medida en metros por segundo (m/s).

Según Rafaat y Hussein (2018), la potencia generada por una turbina eólica se expresa mediante la Ecuación 4.4:

$$P_{\text{avail}} = \frac{1}{2} \rho A V_\infty^3 = 0,5 \pi \rho R_t^2 V_\infty^3 \quad (4.4)$$

donde:

- P es la potencia de viento generada por la turbina, medida en vatios (W).
- ρ es la densidad del aire, medida en kilogramos por metro cúbico (kg/m^3).
- $\pi R_t^2 = A$ es el área barrida por el rotor de la turbina.
- V es la velocidad del viento, medida en m/s .

4.1. ENTORNO FÍSICO

La potencia expresada en la Ecuación 4.4 representa la potencia máxima teórica que podría generar una turbina eólica sin tener en cuenta las fuerzas de rozamiento y los posibles comportamientos defectuosos de sus componentes mecánicos. Sin embargo, el modelo físico de una turbina eólica presenta una restricción crucial en la producción de potencia eléctrica, conocida como el *límite de Betz*, propuesto por Albert Betz (1885-1968), físico alemán pionero en el estudio de la energía eólica. Este límite establece que la potencia máxima teórica que puede ser absorbida por una turbina eólica no puede superar el 59 %.

La potencia teórica final que puede generar una turbina eólica se expresa mediante la expresión de la Ecuación 4.5:

$$P_{\text{mech}} = P_{\text{avail}} C_p(\lambda, \beta) \quad (4.5)$$

en el que $C_p(\lambda, \beta)$ es el *coeficiente de Betz*, donde β representa el ángulo de las palas y λ es el *tip-speed-ratio (TSR)*, en español, cociente de velocidad punta. El *tip-speed ratio* es un parámetro crucial en la aerodinámica de las turbinas eólicas, indicando la relación entre la velocidad de la punta de las palas y la velocidad del viento que las atraviesa. Cuando tiene un valor bajo, las palas giran más lentamente en comparación con la velocidad del viento, y a medida que aumenta, giran más rápidamente.

El *coeficiente de Betz* es un parámetro variable que depende del ángulo de *pitch* (ángulo de las palas) y permite ajustar el movimiento de las palas en relación con la velocidad del viento. Según Petković y cols. (2013), para lograr el seguimiento del punto de potencia máxima de la turbina eólica, la velocidad de rotación del eje de la turbina debe adaptarse de manera óptima con respecto a la variable velocidad del viento. Esto se logra mediante la variación del ángulo de *pitch* de las palas, con el objetivo de mantener el coeficiente de potencia óptimo de la turbina eólica.

La Figura 4.3, proporcionada por Petković y cols. (2013), muestra la relación entre el *coeficiente de Betz*, el *tip-speed-ratio* y los diferentes ángulos de *pitch* o ángulo de las palas. Se observa que cuando las palas se colocan en la dirección del viento, con un

4.1. ENTORNO FÍSICO

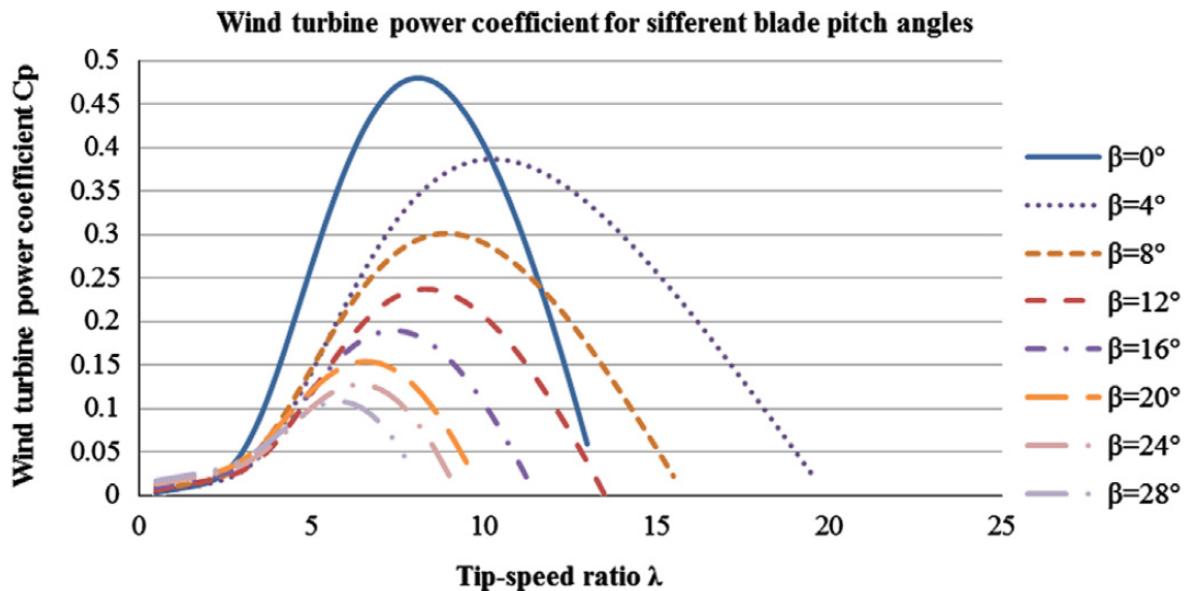


Figura 4.3: Coeficiente de potencia C_p como una función entre el *tip-speed ratio* λ y el ángulo de las palas β . Fuente: [Petković y cols. \(2013\)](#)

ángulo de 0° , la potencia generada por la turbina eólica alcanza su máximo.

Eriksson y cols. (2008), en su artículo “*Evaluation of different turbine concepts for wind power*” (“Evaluación de diferentes conceptos de turbinas para la energía eólica”), describen, utilizando la Figura 4.4, tres tipos de turbinas distintas que operan en diferentes *tip-speed-ratios*. El *coeficiente de Betz* es un parámetro cuyo valor máximo depende de la tecnología empleada en la turbina eólica y los valores típicos de C_p suelen rondar alrededor de 0,40.

Se puede concluir que tanto la velocidad del viento como el ángulo de *pitch* son elementos cruciales en el proceso de maximizar la potencia generada por una turbina eólica. A través del estudio del *coeficiente de Betz*, junto con la potencia generada, estas tres variables se convierten en componentes esenciales que desempeñarán un papel clave en el etiquetado de los datos y en el entrenamiento del sistema propuesto en el [Capítulo 5. Implementación y desarrollo de la arquitectura U-Net + LSTM](#).

Conocer el comportamiento normal de una turbina eólica es clave para posteriormente etiquetar las series temporales de datos en clases que midan el rendimiento de la turbina, con el objetivo de entrenar un modelo de *DL* capaz de

4.1. ENTORNO FÍSICO

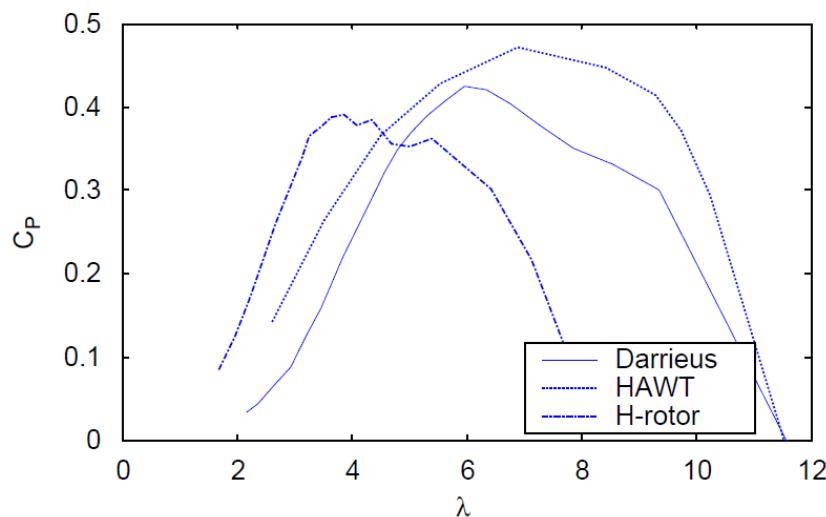


Figura 4.4: Coeficientes de Betz asociado a 3 diferentes tipos de turbinas respecto al *tip-speed ratio*. Fuente: Eriksson y cols. (2008)

predecir anomalías. Retornando a Rafaat y Hussein (2018), en su artículo describen que las regiones de operación de las turbinas eólicas suelen dividirse en cuatro regiones principales, como se observa en la Figura 4.5. La *Región 1* abarca desde el inicio hasta una velocidad mínima del viento llamada “*cut-in*”. En esta región, la turbina eólica no genera potencia; la generación de potencia comienza a partir de la velocidad del viento denominada “*cut-in speed*”. En la *Región 2*, la turbina se encuentra en una zona limitada entre la velocidad del viento “*cut-in*” y la velocidad nominal “*ratio speed*”. En la *Región 3*, las velocidades del viento son superiores a la velocidad nominal y son lo suficientemente altas como para impulsar a la turbina a maximizar la potencia generada; en este caso, se debe regular la velocidad de la turbina mediante el ángulo de *pitch*, manteniendo la potencia producida a niveles seguros y nominales. La *Región 4* ocurre cuando la turbina se apaga para evitar daños debido a elevadas velocidades del viento.

Bilendo y cols. (2022), en su investigación titulada “*Applications and modeling techniques of wind turbine power curve for wind farms—A review*” (“Aplicaciones y Técnicas de Modelado de la Curva de Potencia de Turbinas Eólicas para Parques Eólicos: Una Revisión”), explican que la curva de potencia teórica, como se muestra

4.1. ENTORNO FÍSICO

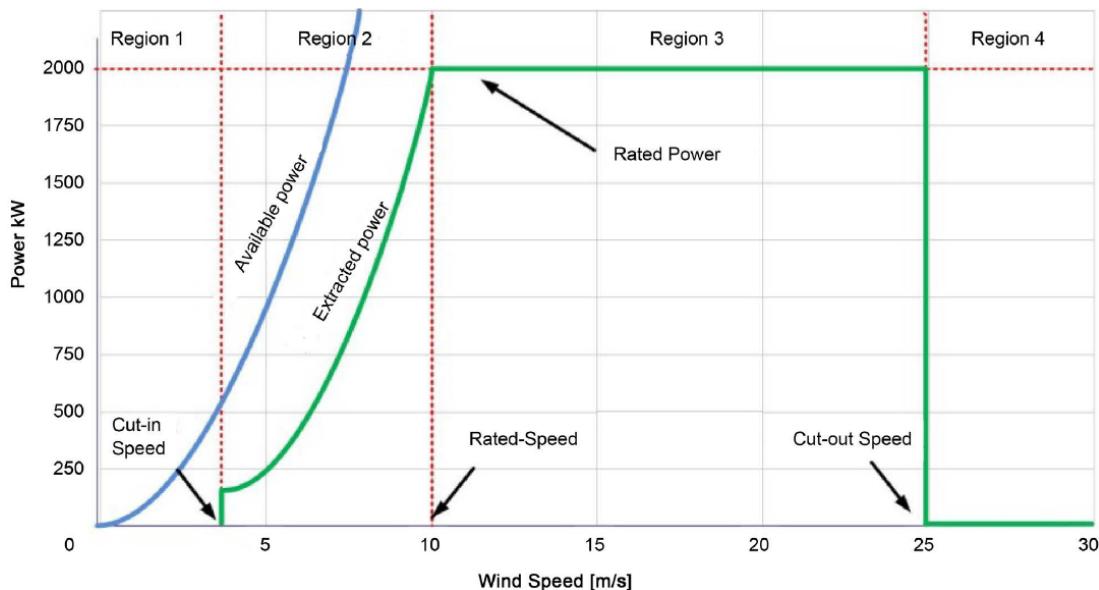


Figura 4.5: Regiones ideales en la curva de potencia de una turbina eólica. Fuente: Rafaat y Hussein (2018)

en la Figura 4.5, no siempre coincide con la curva de potencia práctica presente en los datos recopilados a través del sistema SCADA. Esto se debe a que, en la teoría, no se tiene en cuenta el desgaste de los componentes mecánicos de la máquina.

En el mismo estudio, se destaca que la monitorización del estado de la turbina eólica basada en la curva de potencia es fundamental para los operadores de parques eólicos. Las anomalías y las señales de fallos detectables a través de la curva de potencia pueden desempeñar un papel significativo en la detección temprana de fallos en el comportamiento de una turbina eólica. En la Figura 4.6, extraída de su trabajo, se muestran diferentes tipos de fallos en el comportamiento de una turbina eólica en la curva de potencia. Comparando la curva de potencia ideal proporcionada en la Figura 4.5 anterior con la curva práctica de la Figura 4.6, se pueden analizar puntos en la gráfica que no se corresponden con el comportamiento normal de la turbina eólica.

Es esencial tener presente nuestro objetivo principal: anticipar anomalías en las etapas iniciales mediante el procesamiento de series temporales de datos provenientes de una turbina eólica. Estamos abordando un problema de aprendizaje supervisado, donde el proceso de etiquetado de los datos de entrenamiento juega un papel esencial. En la sección 5.1 Preprocesado de los datos, se detalla minuciosamente el proceso

4.1. ENTORNO FÍSICO

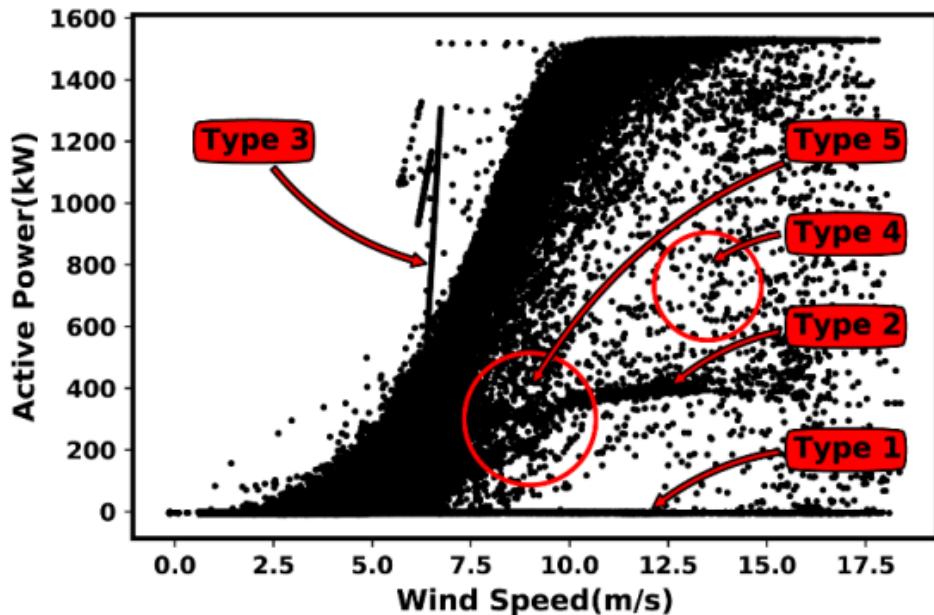


Figura 4.6: Curva de potencia de una turbina eólica marcada con 5 tipos de anomalías en su comportamiento. Fuente: [Bilendo y cols. \(2022\)](#)

de etiquetado. Como avance, es relevante destacar que dicho etiquetado se basará en la observación de la curva de potencia de las turbinas eólicas, según la metodología mencionada en [Bilendo y cols. \(2022\)](#), dividiendo la gráfica en distintas regiones de rendimiento. Este enfoque específico será crucial para mejorar la efectividad de nuestro modelo en la predicción de posibles anomalías. [Bilendo y cols. \(2022\)](#) con la Figura 4.7 presentan, mediante líneas rojas discontinuas, dos curvas de potencia que ayudan a comprender cómo se pueden clasificar los puntos de una curva en base a su posición en la gráfica. Una de ellas exhibe eficiencia en la producción de potencia, mientras que la otra muestra deficiencia dentro de la *Región 2*.

En resumen, este apartado sirve como punto de partida para comprender la importancia de tres variables clave presentes en el conjunto de datos proporcionado por *CénitS*: la potencia generada, la velocidad del viento y el ángulo de *pitch*. Y, aunque existe un mayor conjunto de variables de importancia estimable a la hora de analizar y predecir comportamientos, tal y como [Sánchez-Fernández y cols. \(2023\)](#) han logrado determinar en su reciente investigación, la elección de estas variables es fundamental en los resultados obtenidos por los distintos modelos aplicados en este

4.2. ANÁLISIS DE LOS DATOS

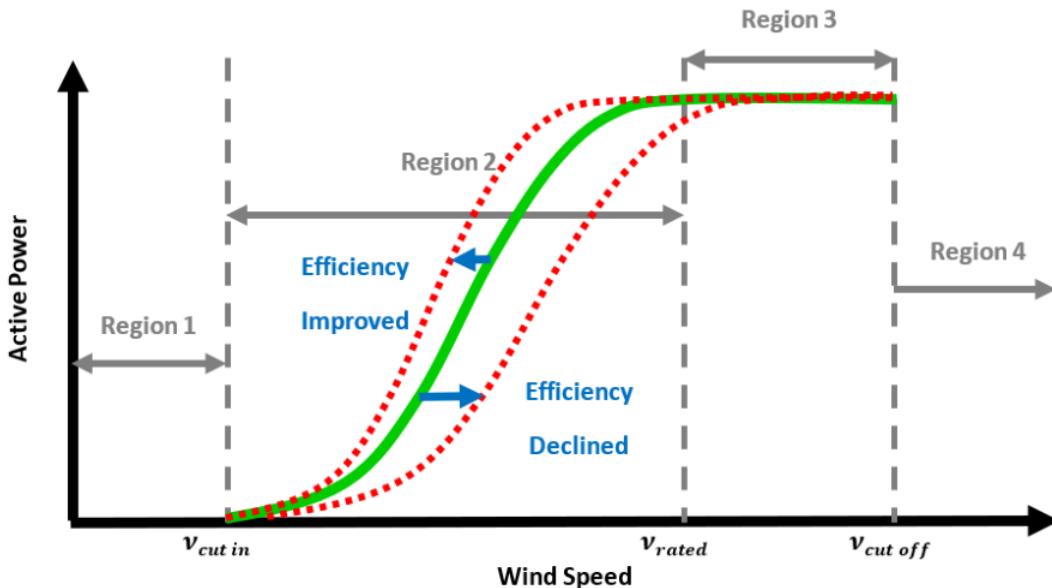


Figura 4.7: Eficiencia de producción de una turbina eólica. Fuente: [Bilendo y cols. \(2022\)](#)

trabajo y que se describirán más adelante en el [Capítulo 5. Implementación y desarrollo de la arquitectura U-Net + LSTM](#).

4.2. Análisis de los datos

En la sección actual, se lleva a cabo un análisis de las series temporales de datos provenientes del conjunto de datos proporcionado por la entidad *CénitS/COMPUTAEX*. El conjunto de datos utilizado en este trabajo recopila información de un parque eólico *onshore*, compuesto por un total de 25 turbinas eólicas. Este conjunto de datos consta de 132 variables medidas por los sensores instalados en las turbinas, gracias al sistema *SCADA*, abarcando el período desde 2009 hasta 2017. Durante este análisis de datos, se procesan las series temporales de la turbina eólica, centrándose en la turbina eólica identificada con el valor 1 (turbina *WTI*).

En el contexto del aprendizaje profundo, antes de entrar en la fase de modelado, resulta fundamental realizar un análisis de las variables más relevantes. Esto se lleva a cabo para recopilar información que pueda ser útil al diseñar la arquitectura neuronal.

4.2. ANÁLISIS DE LOS DATOS

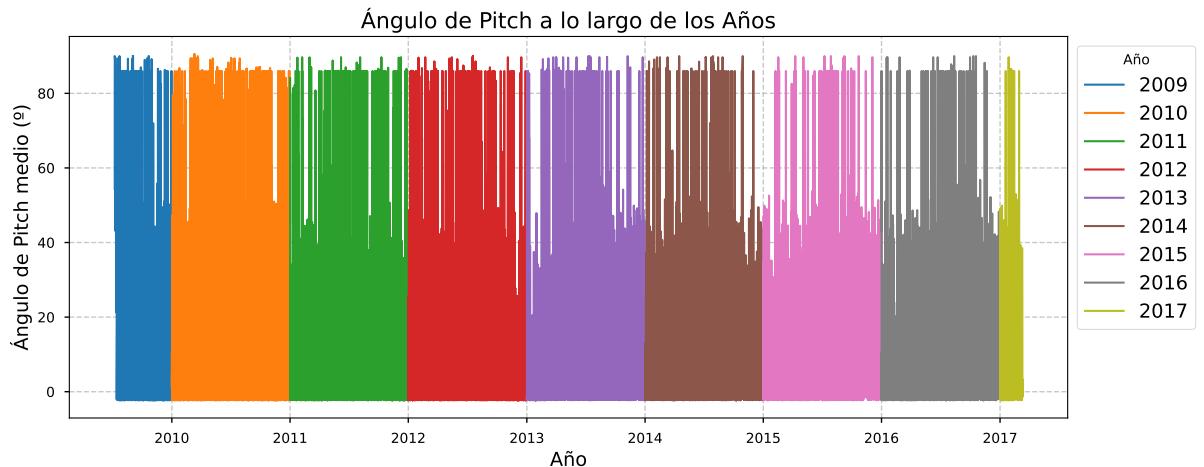


Figura 4.8: Ángulo de *pitch* medio asociado a la turbina eólica durante 8 años.

Considerando el estudio del modelo físico detallado en la sección anterior, las variables con las que trabajamos son el ángulo de *pitch*, la velocidad media del viento y la potencia media generada.

4.2.1. Ángulo de *Pitch*

El ángulo de *pitch* de las palas es un factor crucial para corregir la potencia generada por una turbina, ajustándose al comportamiento normal descrito en la curva de potencia detallada en la sección anterior. En situaciones de elevada intensidad de viento, es necesario controlar el ángulo de *pitch* para desactivar el mecanismo del rotor de la turbina y evitar posibles daños.

La Figura 4.8 visualiza el ángulo de *pitch* medio asociado a la turbina eólica a lo largo de un periodo de 8 años (desde 2009 hasta 2017). Se observa que el valor predominante del ángulo de *pitch* se sitúa en torno a $0^\circ - 15^\circ$, con el objetivo de maximizar el *coeficiente de Betz*, previamente mostrado en la Figura 4.3. Además, se aprecian picos de valores cercanos a los 90° , característicos de la detención del funcionamiento de la turbina frente a intensidades de viento elevadas.

En la Figura 4.9 se representa el ángulo de *pitch* medio en relación con cada mes del año. Se observa un comportamiento constante a lo largo de cada mes, aunque los puntos representados no muestran picos de valores que podrían detener

4.2. ANÁLISIS DE LOS DATOS

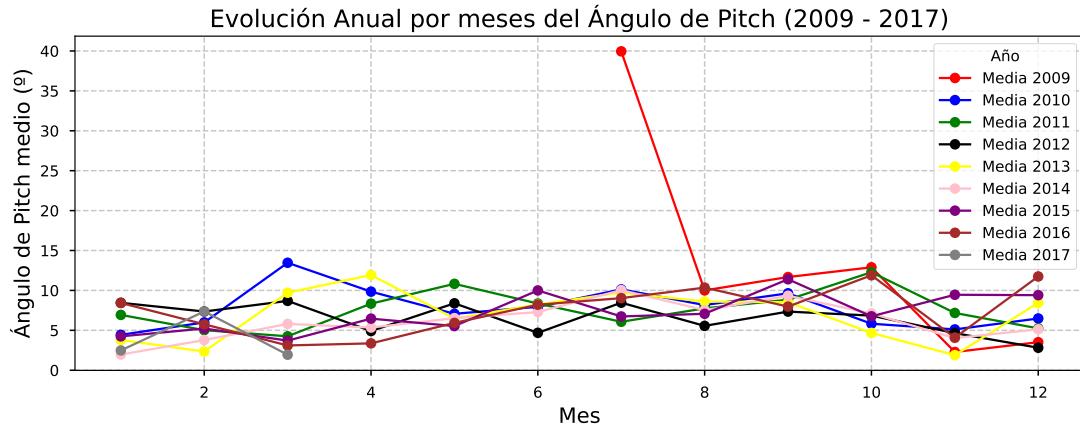


Figura 4.9: Ángulo de *pitch* medio asociado a la turbina eólica en relación con cada mes.

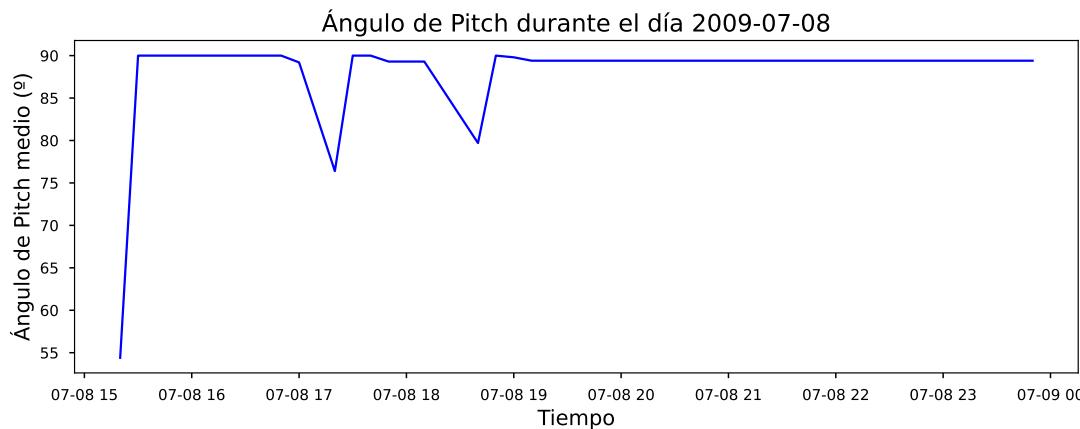


Figura 4.10: Ángulo de *pitch* medio asociado a la turbina eólica durante un día (9 horas).

el funcionamiento de la turbina eólica para prevenir fallos ante velocidades elevadas de viento.

Y, por último, en la Figura 4.10 se refleja la variación del ángulo de *pitch* durante el primer día capturado en el conjunto de datos (9 horas) de la turbina eólica asociada. El ángulo de *pitch* oscila entre 75° y 90°, indicando que la turbina eólica se encuentra en un estado de parada o arranque. Con respecto al esquema de la curva de potencia presentado en la anterior Figura 4.5, corresponde a la *Región 1* (estado de arranque) o a la *Región 4* (estado apagado), respectivamente.

4.2. ANÁLISIS DE LOS DATOS

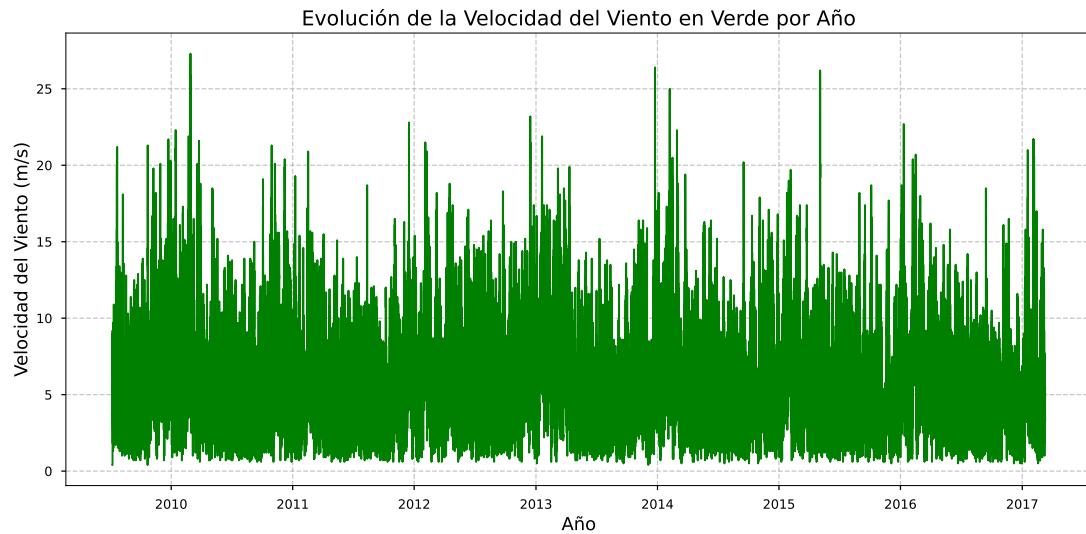


Figura 4.11: Velocidad media del viento de la turbina eólica durante 8 años.

4.2.2. Velocidad del Viento

La producción de energía eólica de una turbina está intrínsecamente ligada a la velocidad del viento, como se ha detallado en el modelo físico. La potencia generada por una turbina eólica depende directamente de la densidad del viento, su dirección y su intensidad.

La Figura 4.11 dibuja la evolución de la velocidad del viento de la turbina a lo largo de los ocho años del conjunto de datos tratado. Se comprueba que la velocidad oscila entre 5 y 10 m/s, situación ubicada en la *Región 2* considerada en la Figura 4.5. Esta estimación media de la velocidad del viento puede resultar crucial para detectar posibles anomalías en la turbina. Por ejemplo, si en ciertos momentos la máquina intenta producir más energía eólica cuando la velocidad no supera la nominal, podría indicar una sobrecarga.

4.2. ANÁLISIS DE LOS DATOS

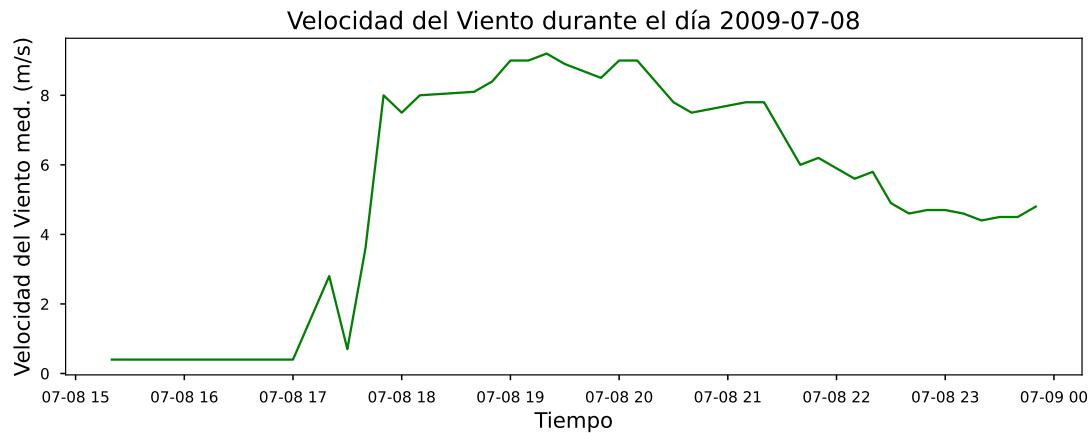


Figura 4.13: Velocidad media del viento asociado a la turbina eólica durante un día (9 horas).

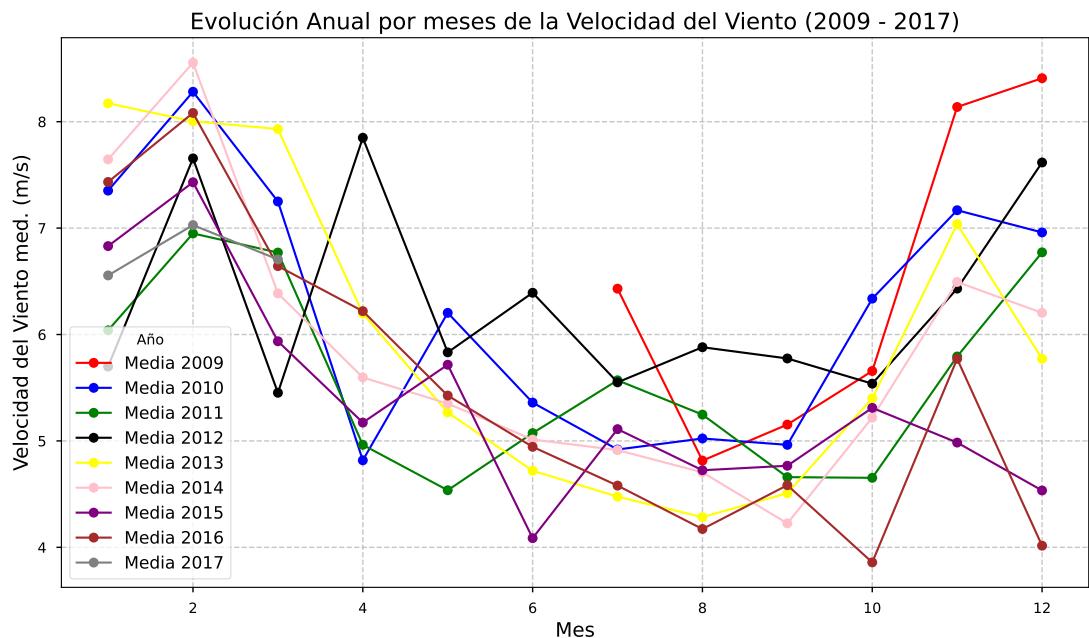


Figura 4.12: Velocidad media del viento de la turbina eólica en relación con cada mes.

Luego, si observamos la Figura 4.12, que representa la velocidad del viento media en relación con los meses del año, se puede apreciar que existe una disminución en la intensidad del viento durante los meses más cálidos (de junio a septiembre). Enfocar la detección de anomalías en estos meses podría ser crucial, ya que la turbina podría estar sobreactuando para generar más energía eólica en momentos de velocidades de viento insuficientes.

4.2. ANÁLISIS DE LOS DATOS

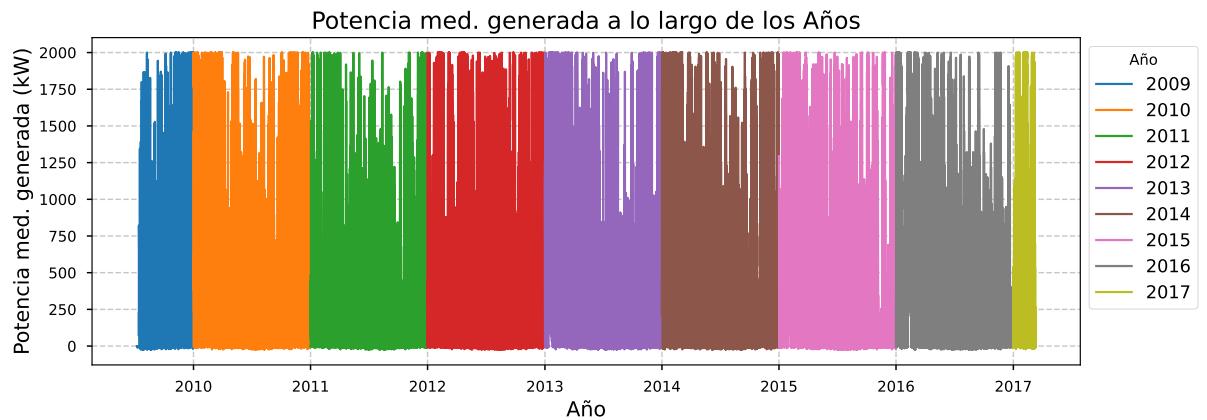


Figura 4.14: Potencia media generada por la turbina eólica durante 8 años.

La Figura 4.13 representa la evolución de la velocidad del viento en el primer día (9 horas) capturado en el conjunto de datos. Desde las 15:00 hasta las 17:00, la velocidad del viento es prácticamente nula, y desde las 18:00 hasta las 21:00 se mantiene constante en $7,5m/s$ hasta el final del día, cuando experimenta una caída hasta los $4m/s$.

4.2.3. Potencia

La potencia es la variable más crucial en la detección de anomalías de una turbina eólica, ya que sirve como base para el etiquetado del conjunto de entrenamiento y evalúa el rendimiento de la turbina.

La Figura 4.14 muestra la potencia media generada por una turbina eólica en los ocho años que abarca el conjunto de datos. Se observa que la potencia media oscila entre 1000 y 2000 kW , pero no se evidencia una variación significativa a lo largo de los años.

4.2. ANÁLISIS DE LOS DATOS

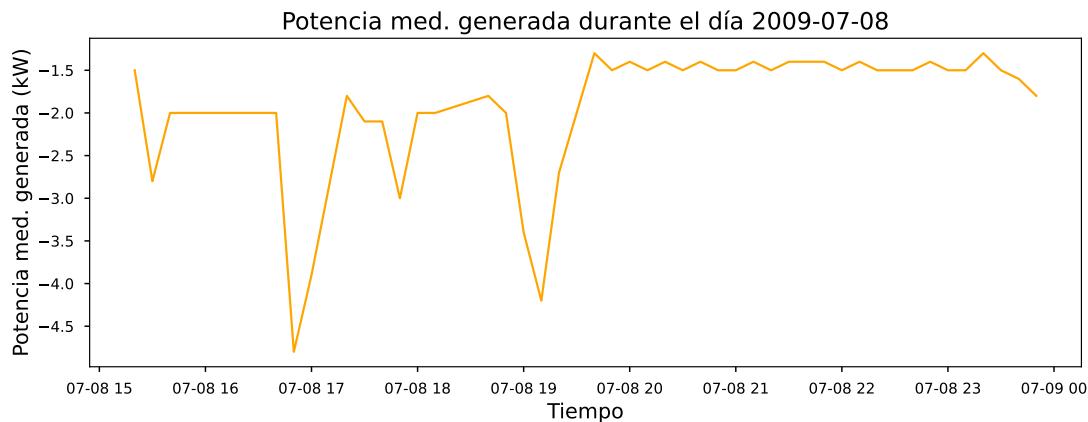


Figura 4.16: Potencia media generada por la turbina eólica durante un día (9 horas).

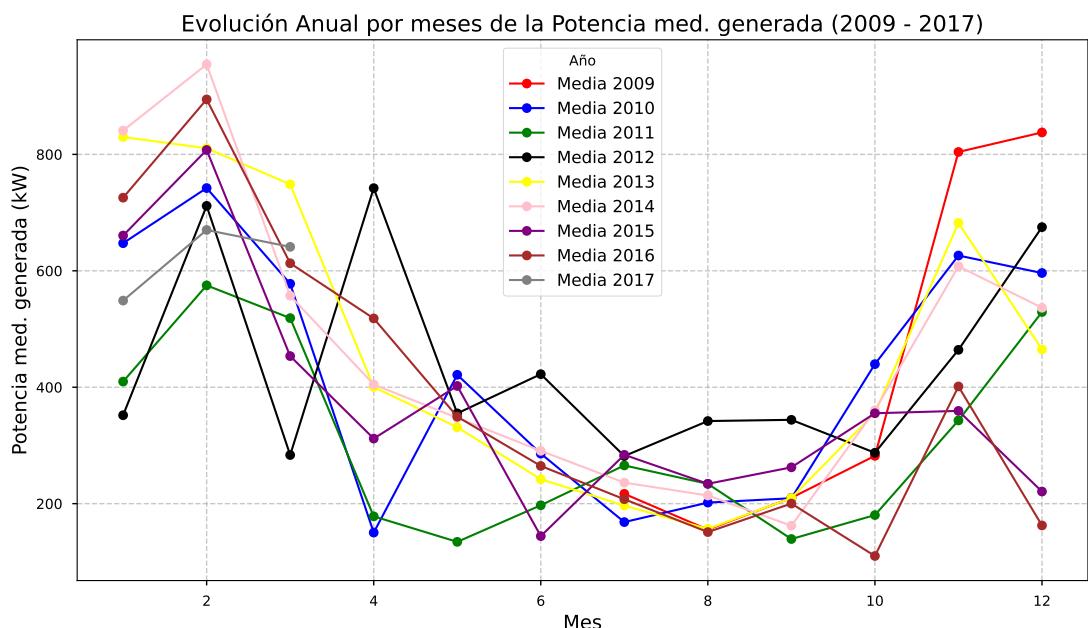


Figura 4.15: Potencia media generada por la turbina eólica en relación con cada mes.

De igual manera que se ha hecho con la velocidad del viento, con la Figura 4.15 contigua, que representa la potencia media generada en relación con cada mes del año, se verifica una tendencia concordante con la evolución de la velocidad del viento en los meses de julio a septiembre, como se constata en la Figura 4.12, donde la velocidad del viento es menor que en otros meses, resultando en una menor generación de energía eólica.

Al representar la evolución de la potencia media generada por la turbina eólica

4.2. ANÁLISIS DE LOS DATOS

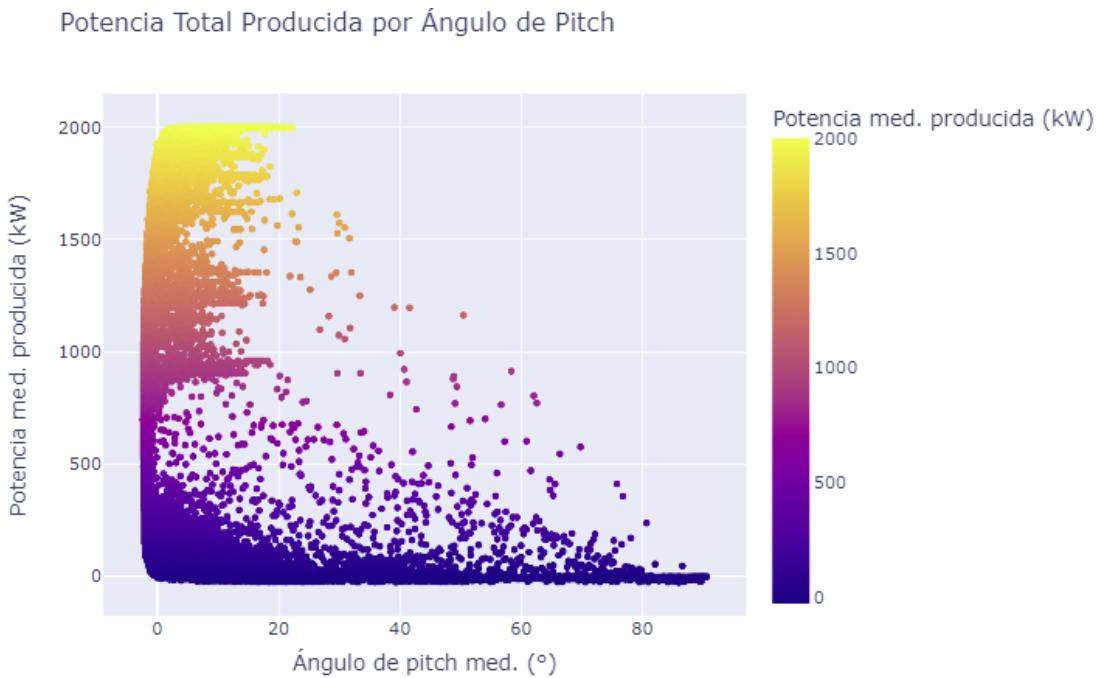


Figura 4.17: Curva de la potencia media generada en función del ángulo de *pitch*.

durante un día (9 horas), mostrada en la Figura 4.16 subsiguiente, se ve con relativa claridad que los valores de potencia generada son negativos cuando la fuerza del viento no es suficiente para cubrir la potencia requerida para el funcionamiento de la turbina eólica.

A continuación, se utilizarán varias gráficas para describir la relación de la curva de potencia de la turbina eólica en cuestión con respecto a la velocidad del viento y el ángulo de *pitch*. La Figura 4.17 visualiza la relación de la potencia media generada por la turbina en función del ángulo de *pitch* medio de las palas. Se coteja que el ángulo de *pitch* no funciona como un indicador directo para maximizar la potencia generada, pero es útil para entender que, en un ángulo cercano a los 90°, la potencia generada es mínima.

Con las tres figuras que siguen, Figuras 4.18, 4.19 y 4.20, se describe el comportamiento de la turbina eólica en términos de potencia media, máxima y mínima, respectivamente, producida en relación con la velocidad del viento. La primera, Figura 4.18, muestra el comportamiento teórico normal de una turbina eólica, siguiendo el esquema de las regiones presentadas en la Figura 4.5. Desde 0 hasta 5 m/s,

4.2. ANÁLISIS DE LOS DATOS

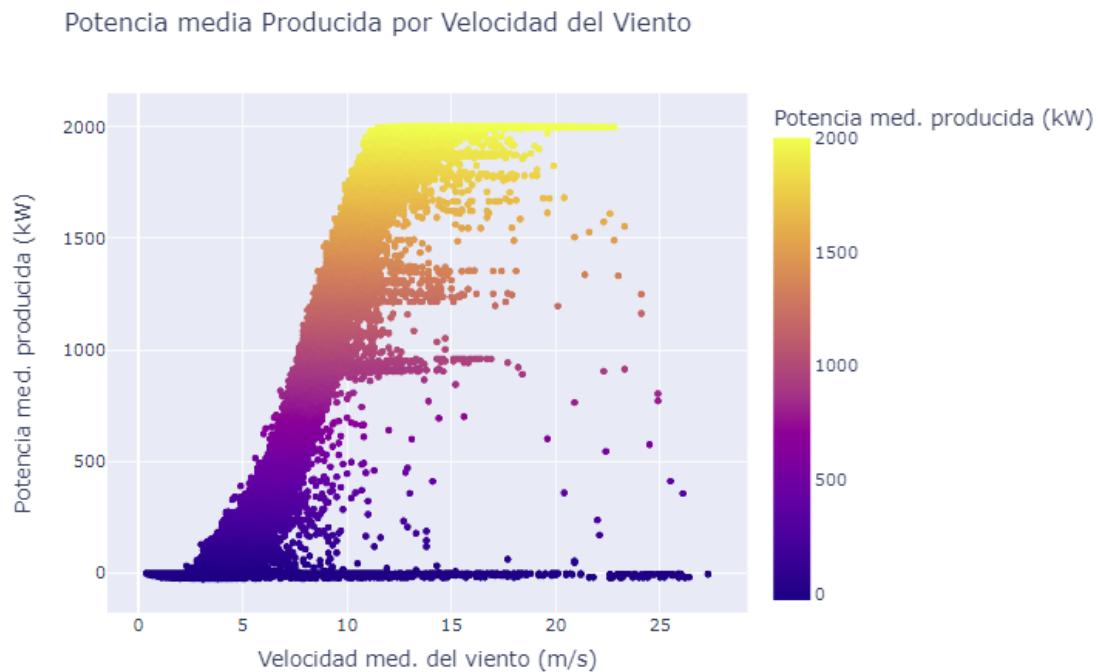


Figura 4.18: Curva de la potencia media generada en función de la velocidad media del viento.

prácticamente no se produce energía eólica. Cuando la velocidad del viento supera los 5 m/s , la turbina comienza a funcionar hasta alcanzar un valor nominal de $12,5\text{ m/s}$, momento en el cual se maximiza la producción de energía eólica. Ráfagas intensas de viento pueden resultar en la parada de la máquina para prevenir fallos.

La segunda Figura 4.19 representa la potencia máxima generada en relación con la velocidad media del viento. La gráfica resume un comportamiento que puede interpretarse como anómalo, ya que sobrecargar la turbina para que produzca grandes cantidades de energía sin la suficiente intensidad de viento puede dar lugar a fallos mecánicos y a un desgaste prematuro de sus componentes.

En el caso opuesto, en la tercera Figura 4.20, se observa un aumento de los puntos de la gráfica en las rachas de viento que varían entre $12,5$ y 20 m/s , lo cual puede indicar anomalías en el funcionamiento de la máquina al no acercarse a la potencia máxima teórica generada, que ronda los 2000 kW .

4.2. ANÁLISIS DE LOS DATOS

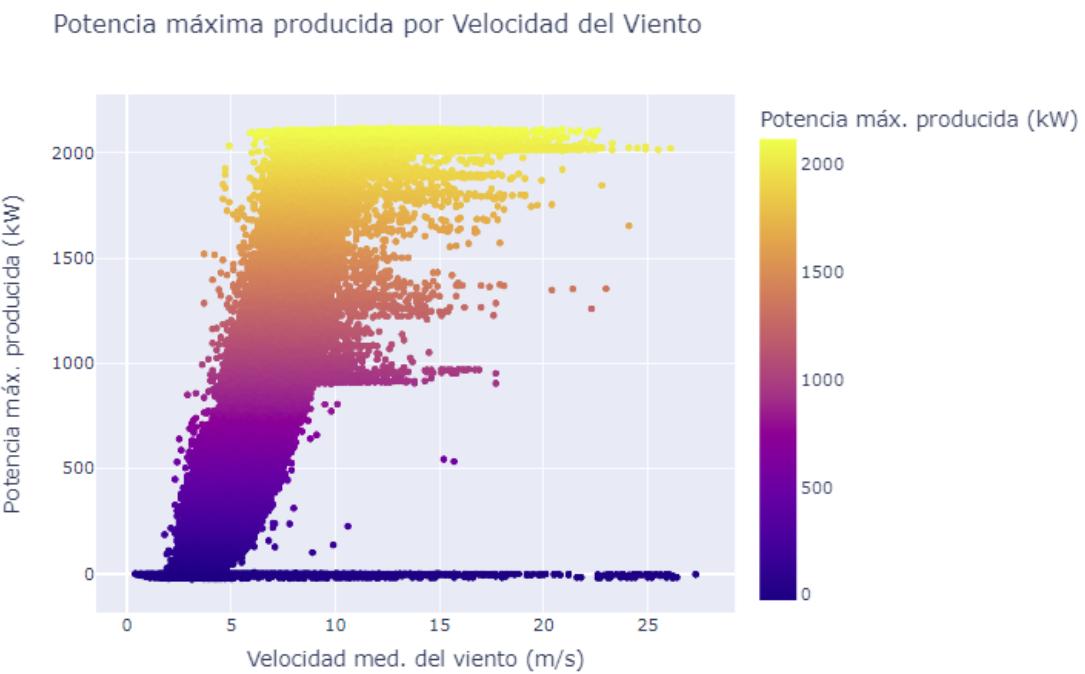


Figura 4.19: Curva de la potencia máxima generada en función de la velocidad media del viento.

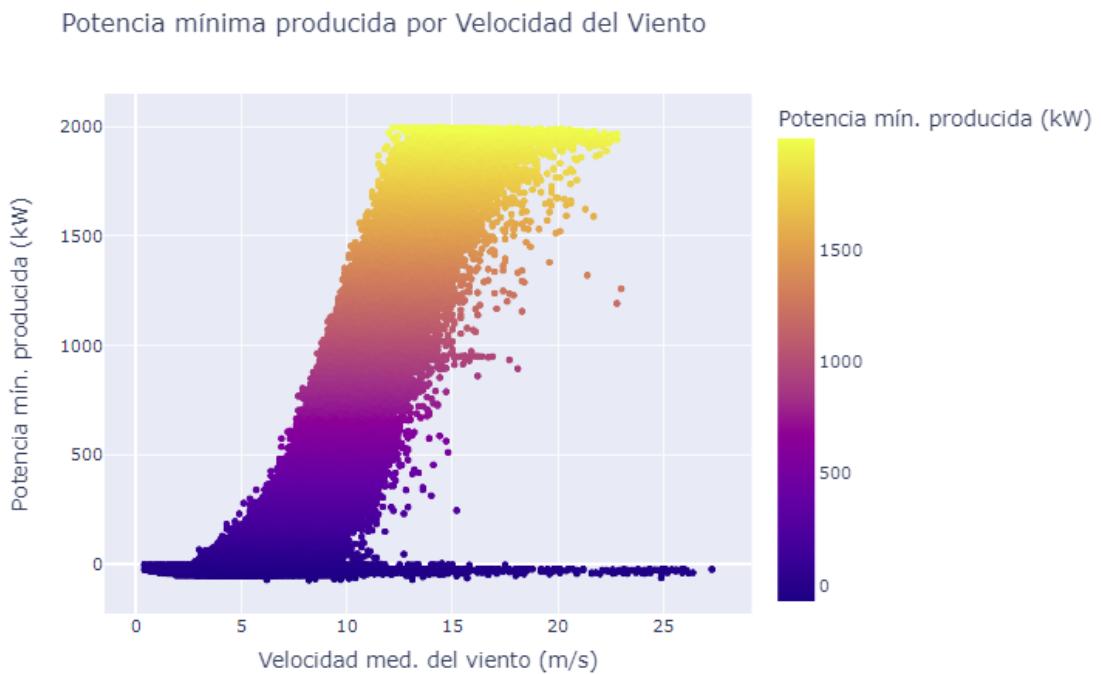


Figura 4.20: Curva de la potencia mínima generada en función de la velocidad media del viento.

4.2. ANÁLISIS DE LOS DATOS

4.2.4. Correlación entre las variables

Se ha empleado una técnica de visualización, conocida como *mapa de calor de correlación*⁴, o ”*correlation heatmap*”, para analizar las relaciones presentes en el conjunto de datos entre las variables de potencia media generada, velocidad del viento y ángulo de *pitch*. El estudio de la correlación entre variables desempeña un papel fundamental en la formulación del modelo, detallado en el próximo [Capítulo 5](#).

Implementación y desarrollo de la arquitectura *U-Net + LSTM*.

La Figura 4.21 presenta el mapa de calor que denota la correlación entre las cinco variables más relevantes del conjunto de datos: ángulo de *pitch*, velocidad media del viento, potencia media generada, potencia mínima generada y potencia máxima generada. En este mapa de calor, las variables que mantienen una relación más estrecha entre sí se representan en color rojo, con valores cercanos a 1. Mirándolo con atención, se constata que la velocidad del viento está directamente relacionada con las tres medidas de potencia generadas; sin embargo, no se evidencia una correlación directa entre el ángulo de *pitch* y las demás variables, ya que este ángulo de las palas se utiliza para controlar el comportamiento de la turbina y prevenir posibles errores.

Las conclusiones extraídas en este apartado proporcionan el contexto esencial para la siguiente fase de la investigación, la cual se centra en la implementación de la arquitectura seleccionada, el detallado proceso de etiquetado de datos y la configuración precisa de los parámetros más críticos. Estos hallazgos son fundamentales para comprender y llevar a cabo la ejecución efectiva de la arquitectura, asegurando así una configuración óptima de los hiperparámetros esenciales.

⁴**Mapa de calor de correlación:** es una tabla que muestra los coeficientes de correlación entre varias variables. Cada celda de la matriz contiene el coeficiente de correlación entre la variable en la fila y la variable en la columna correspondiente. Este tipo de matriz es especialmente útil para analizar la asociación entre múltiples variables simultáneamente.

4.2. ANÁLISIS DE LOS DATOS

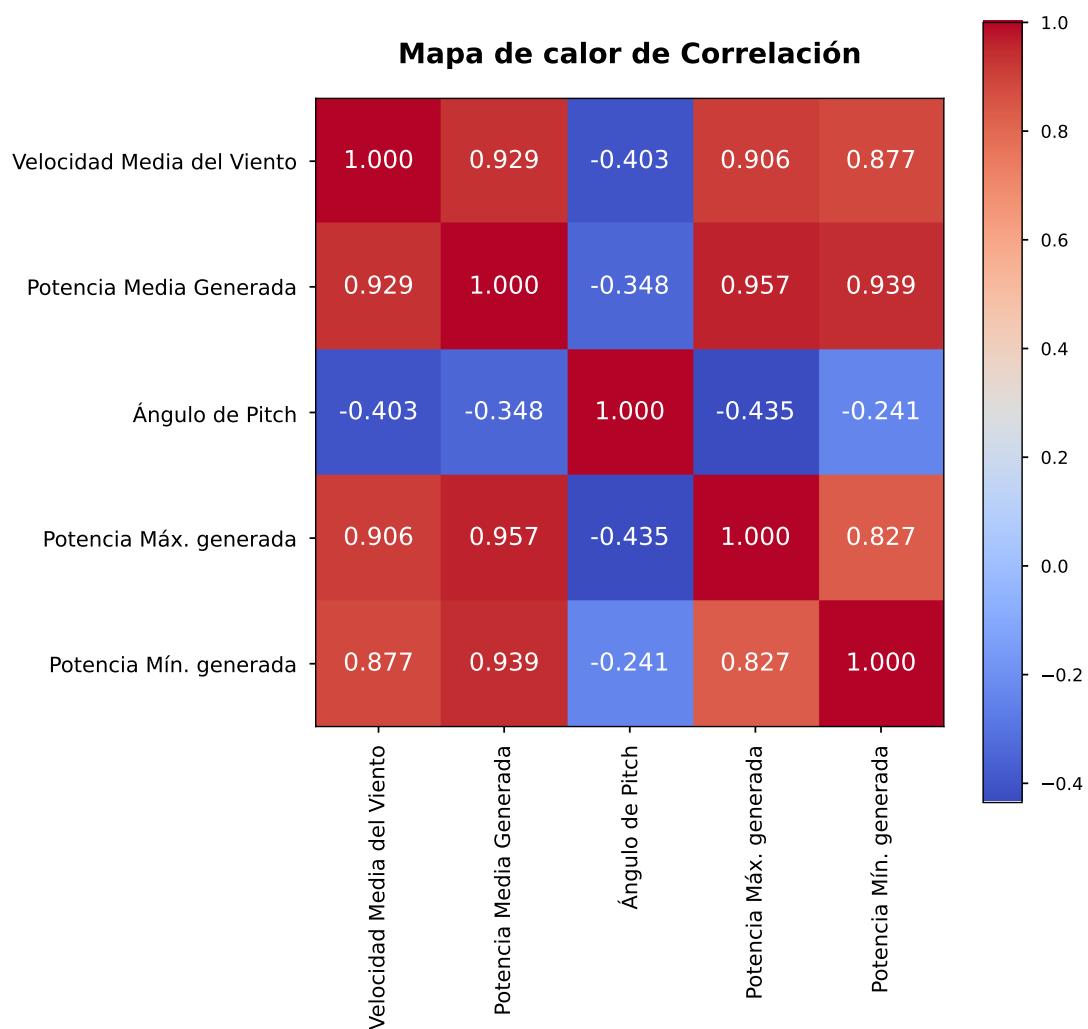


Figura 4.21: Matriz de correlación de las 5 variables más importantes del *dataset*.

Capítulo 5

Implementación y desarrollo de la arquitectura *U-Net + LSTM*

Este capítulo comienza examinando en detalle el proceso de preprocesamiento del conjunto de datos proporcionado por *CénitS*, destinado a la conversión de las series temporales de datos en imágenes y su posterior etiquetado en el apartado 5.1 Preprocesado de los datos.

El siguiente apartado 5.2 Sistema de autoetiquetado. Arquitectura *U-Net* describe la arquitectura de red neuronal convolucional *U-Net* empleada por Cambero Rojas (2023) en un trabajo fin de grado anterior, la cual conforma un sistema de autoetiquetado de las señales temporales de datos convertidas a imágenes sintéticas.

Además, en el apartado 5.3 Estudio de modelos *baseline*, se describe el estudio de dos modelos iniciales que sirven de punto de partida para alcanzar el objetivo propuesto de crear un sistema de predicción temprana de anomalías. El primer modelo es una técnica de regresión de *Machine Learning*, mientras que el segundo es una red neuronal recurrente *LSTM* en diferido. Ambos modelos operan sobre una serie temporal de datos generada por la arquitectura *U-Net* detallada en el apartado anterior. A pesar de su utilización inicial, se descartan estas opciones debido a su limitada capacidad para capturar la complejidad de los datos.

En lugar de ello, en el apartado 5.4 Arquitectura *U-Net + LSTM* propuesta se

5.1. PREPROCESADO DE LOS DATOS

profundiza en una arquitectura más compleja que combina la estructura *U-Net* con capas neuronales *LSTM* en un mismo modelo, junto con su configuración óptima. El enfoque se centra en abordar de manera eficiente la tarea de desarrollar un modelo de predicción temprana de anomalías en las turbinas eólicas.

A la vista de los resultados prometedores obtenidos por el modelo *U-Net + LSTM*, se plantea un nuevo modelo constituido por unas capas de redes neuronales especiales llamadas *ConvLSTM* en el apartado [5.5 Arquitectura *ConvLSTM*](#).

5.1. Preprocesado de los datos

En este apartado se explican dos técnicas de preprocesado del conjunto de datos: la conversión de las series temporales de datos a imágenes, que es una forma necesaria de representar los datos para el entrenamiento de un modelo convolucional, y el proceso de etiquetado del conjunto de datos.

5.1.1. Conversión de las series temporales de datos a imágenes

Como se ha explicado en el [Capítulo 2. Antecedentes](#), trabajar con una RNC *U-Net* y con series temporales de datos requiere convertir las series temporales en imágenes. En su artículo científico *"Wind turbine fault detection and classification by means of image texture analysis"* ("Detección de fallos en turbinas eólicas y clasificación mediante el análisis de texturas de imagen"), Ruiz y cols. (2018) describen el proceso de conversión de los datos de señales capturadas por sensores de las turbinas eólicas, recopilados por el sistema SCADA, para detectar anomalías mediante clasificación. Los datos de los sensores se almacenan como series temporales de vibraciones y se convierten posteriormente en imágenes. La conversión de la serie temporal de datos a imágenes de Ruiz y cols. (2018) se representa en la Figura 5.1. Primeramente se obtiene un conjunto de datos discretos a lo largo del tiempo determinado por marcas de tiempo o *timestamps*. Este proceso incluye un método de partición de la serie temporal de una señal en muestras sucesivas con N valores discretos, utilizando una técnica de

5.1. PREPROCESADO DE LOS DATOS

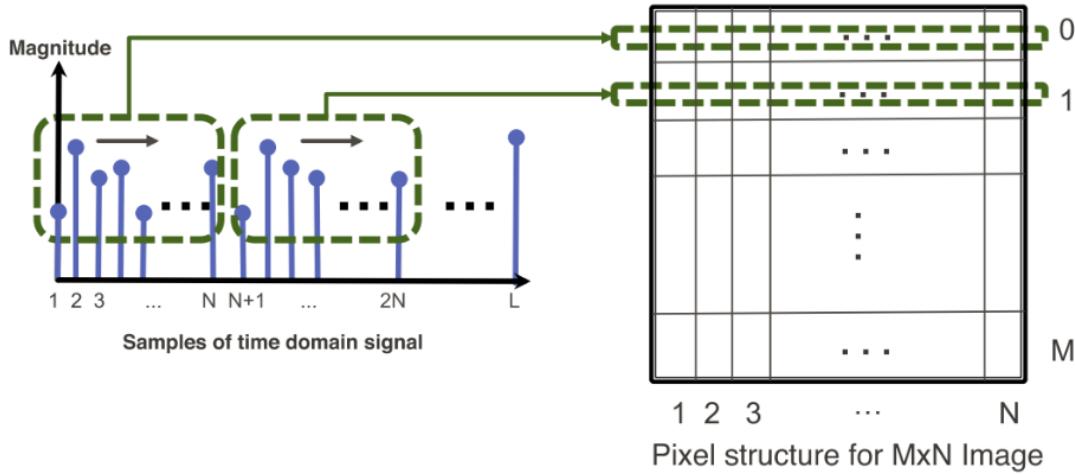


Figura 5.1: Conversión de una señal a una imagen de tamaño $M \times N$. Fuente: [Ruiz y cols. \(2018\)](#)

ventana de tiempo deslizante. La ventana debe tener un tamaño fijo. Cada muestra de tamaño N ocupa una fila de una imagen representada por una matriz. Por lo tanto, si la ventana deslizante tiene un tamaño de N muestras y elegimos M ventanas, se forma una imagen $N \times M$ que representa la serie temporal adecuada a $M \times N$ timestamps, correspondiente a M ventanas de tiempo.

La selección del tamaño de la ventana, o lo que es lo mismo, la selección de la longitud de tiempo de cada muestra, es un aspecto clave a la hora de predecir anomalías. Cada imagen debe abarcar un período de tiempo lo suficientemente amplio como para capturar patrones relacionados con anomalías o estados de rendimiento diferentes de la turbina eólica. De esta manera, la red convolucional, que se explicará más adelante en este mismo capítulo, puede aprender a reconocer patrones anómalos al procesar la información completa relacionada con una anomalía disponible en su respectiva imagen.

En el presente trabajo, el proceso de conversión de las señales vibratorias a imágenes está basado en el enfoque propuesto por [Ruiz y cols. \(2018\)](#). Aún así, hay que añadir que una de las contribuciones novedosas de nuestro estudio implica el solapamiento de la ventana de tiempo deslizante entre muestras consecutivas en la serie temporal para así mantener un recuerdo de las señales de momentos anteriores.

5.1. PREPROCESADO DE LOS DATOS

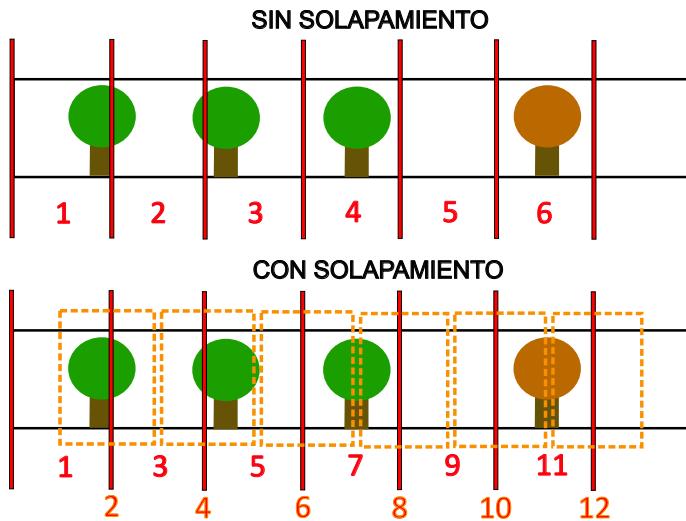


Figura 5.2: Partición con y sin solapamiento mediante la ventana de tiempo deslizante.

Esta técnica mejorada ofrece dos ventajas esenciales en nuestro ámbito de estudio. En primer lugar, un aumento en el tamaño del conjunto de datos: se generan más imágenes para nuestro conjunto de datos. Uno de los desafíos que enfrenta el aprendizaje profundo es la escasez de datos limpios, sin ruido. La técnica impacta positivamente en el entrenamiento del modelo y en la posterior inferencia debido a su capacidad de generalización mejorada al haber sido expuesto a una mayor cantidad de datos.

La segunda ventaja radica en la mejora de la predicción temprana de anomalías. En la Figura 5.2 que sigue se aclara el concepto con un ejemplo de vídeo plasmando una serie temporal de imágenes. Partimos de la hipótesis de predecir con la mayor antelación posible la detección de un árbol otoñal, entre una multitud de árboles verdes. Una predicción es más temprana cuanto menos información del pasado tengamos que proporcionar al modelo para que reconozca el patrón que predice la aparición del árbol otoñal en el futuro. En la parte superior de la figura se observa una partición mediante la ventana deslizante de tiempo sin solapamiento, es decir, la información contenida en cada imagen es única. En esta representación, un modelo de red neuronal predictivo será capaz de predecir un árbol otoñal cuando procese la imagen numérica 6 (en rojo). En la mitad inferior, se observa una partición de la serie temporal mediante

5.1. PREPROCESADO DE LOS DATOS

el solapamiento de la ventana deslizante. Por cada dos imágenes del conjunto de datos reales, se añade una imagen intermedia que contiene parte de la información redundante, como se refleja en la imagen 2 (en naranja), que contiene una porción de información relativa a las imágenes 1 y 3 (en rojo). La novedad de esta técnica radica en que la red puede aprender a predecir un árbol otoñal en el momento en que procesa la imagen 10 (en naranja), la cual contiene una porción del árbol otoñal, aunque no en su totalidad.

Aplicado a la predicción de anomalías en turbinas eólicas, esta técnica facilita una predicción más temprana de una anomalía, ya que requerirá menos información del pasado relacionada con dicha anomalía para anticipar su ocurrencia en el futuro.

En relación al tamaño de las imágenes, [Delgado Muñoz \(2022\)](#), en su trabajo de Fin de Grado titulado “Aplicación de técnicas de clasificación mediante red neuronal *U-Net* a datos provenientes de turbinas eólicas”, concluye que las imágenes de dimensiones 8×8 , que contienen un total de 64 *timesteps*, cada uno de 10 minutos, equivalente a una ventana de tiempo de 11 horas, son suficientes para que la imagen pueda contener la información completa de una posible anomalía.

En el caso de estudio que nos ocupa, las variables físicas que más influyen en el rendimiento de una turbina eólica son la potencia activa media, la velocidad del viento y el ángulo de *pitch* (véase el apartado [4.2 Modelo físico de una turbina eólica](#) para conocer el proceso de selección de las variables). Siguiendo el procedimiento de conversión, cada ventana de tiempo deslizante generará 3 imágenes, correspondientes a cada una de las 3 características (*features* en inglés) del conjunto de datos.

La Figura [5.3](#) refleja una ventana de tiempo de 64 valores discretos de la serie temporal asociada a la potencia activa media, convertida en una imagen en escala de grises de tamaño 8×8 píxeles.

5.1.2. Proceso de etiquetado del conjunto de imágenes

En la sección [4.2 Modelo físico de una turbina eólica](#), se describe la importancia de la curva de potencia generada por una turbina eólica en el proceso de etiquetado. Según

5.1. PREPROCESADO DE LOS DATOS

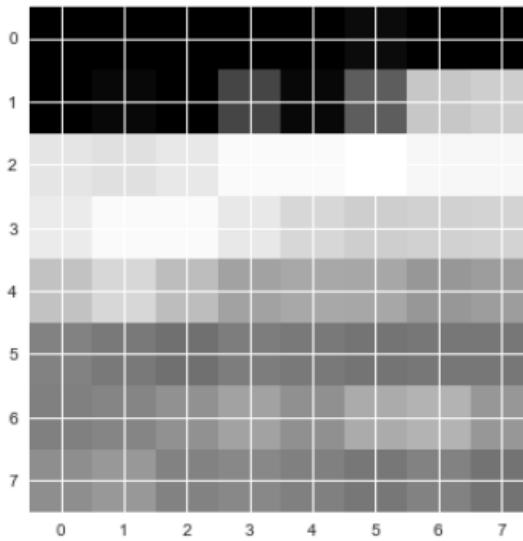


Figura 5.3: Ventana de tiempo de 64 *timesteps* de una serie temporal de potencia activa media convertida a imagen

estudios previos realizados en *CenitS/COMPUTAEX*, enmarcados en un primer trabajo de Sánchez-Fernández y cols. (2023) dentro del proyecto de investigación *Anemoi*, la curva de potencia se puede dividir en 6 clases, que califican el estado de una turbina eólica, como se describe en la Tabla 5.1 y se refleja en la Figura 5.4. Cada imagen tiene asociada la etiqueta que presenta una mayor moda¹ en sus 64 muestras discretas o píxeles. En la misma tabla se presenta también el recuento de imágenes asociadas a cada etiqueta de las 25 turbinas del parque eólico monitorizado, utilizando un tamaño de ventana de tiempo de 64; es decir, de 8×8 . Se constata que la mayor parte de las imágenes del conjunto de datos presentan un comportamiento de óptimo rendimiento (un total de 678.981 imágenes). Esta observación será clave para el posterior modelado y para comprender cómo afecta el desequilibrio de clases (etiquetas) en el proceso de entrenamiento.

¹**Moda:** en estadística, la moda es el valor que aparece con mayor frecuencia en un conjunto de datos

5.1. PREPROCESADO DE LOS DATOS

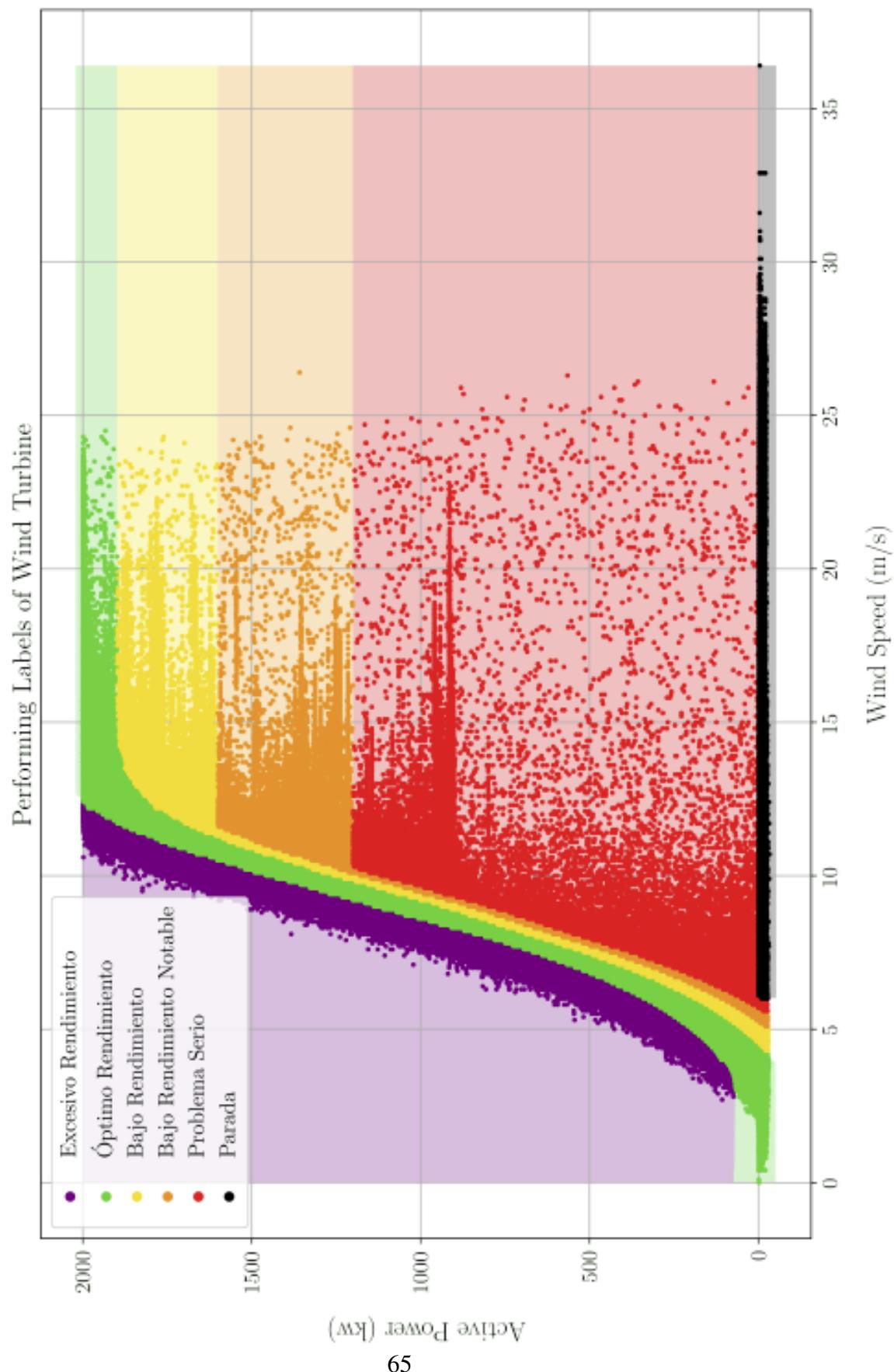


Figura 5.4: Clasificación de la curva de potencia media activa de una turbina eólica.

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

Etiqueta	Descripción	Nº imágenes
0 - Excesivo Rendimiento	Sobreesfuerzo realizado por la turbina.	57424
1 - Óptimo Rendimiento	Funcionamiento normal de la turbina.	678981
2 - Bajo Rendimiento	Comportamiento poco óptimo de la turbina. Se produce una cantidad de energía ligeramente por debajo de la nominal.	44104
3 - Bajo Rendimiento Notable	Comportamiento no óptimo, insuficiencia energética.	3532
4 - Problema Serio	Mal comportamiento de la turbina.	2157
5 - Parada	Estado de parada de la turbina. Nula generación de energía debido a una avería o altas velocidades de viento.	5157

Tabla 5.1: Descripción de las seis etiquetas con el número de imágenes asociado.

5.2. Sistema de autoetiquetado. Arquitectura *U-Net*

Se ha comenzado partiendo del modelo desarrollado por [Cambero Rojas \(2023\)](#) en su trabajo fin de grado, también enmarcado en el proyecto *Anemoi*, en el que se propone un modelo de red neuronal convolucional *U-Net* para el diagnóstico de fallos en turbinas eólicas. Este modelo se encarga de realizar una segmentación semántica de las señales que han sido convertidas en imágenes con el fin de detectar clases anómalas. Y la función de este modelo en el presente trabajo consiste en etiquetar, en una primera instancia, las imágenes sintéticas que representan señales, almacenando todas estas etiquetas (clases) para formar un conjunto de datos intermedio que constituye una serie temporal de clases.

5.2.1. Por qué la arquitectura *U-Net*

Las redes convolucionales, también conocidas como *convnets*, han dominado durante mucho tiempo el ámbito del reconocimiento visual. Sin embargo, su efectividad ha estado históricamente limitada por el tamaño del conjunto de entrenamiento y la escala de la red. El avance en el desarrollo de *hardware* permitió a Krizhevsky diseñar *AlexNet*², una red de 8 capas con millones de parámetros, entrenada con un gran conjunto de datos supervisados.

A partir de *AlexNet*, se comenzaron a entrenar redes de gran envergadura para tareas de clasificación, donde el objetivo principal era asignar una etiqueta de clase a una imagen. En el campo de la medicina, especialmente en el procesamiento de imágenes biomédicas, la salida deseada debía incluir información de localización, es decir, asignar una clase a cada píxel. Este tipo de problemas se conoce como segmentación semántica.

En Ciresan y cols. (2012), se propone un prototipo de *convnet* capaz de predecir la clase a la que pertenece cada píxel mediante un *patch*, una región de píxeles circundantes que sirve como entrada al modelo. Este enfoque permite localizar patrones observables en regiones locales y ampliar el conjunto de datos, ya que el modelo opera con regiones reducidas de una imagen en lugar de la imagen completa. Sin embargo, este modelo presenta algunos inconvenientes notables:

- La velocidad de procesamiento de la red queda afectada debido al gran número de *patches*.
- Existe información redundante debido a la superposición de píxeles entre los *patches*.
- Se presenta un conflicto entre el uso del contexto (tamaño del *patch*) y la precisión de la localización.

²*AlexNet*: pionero modelo de red neuronal convolucional diseñado por Alex Krizhevsky en 2012. Ganador del desafío *ImageNet*, destaca por su arquitectura profunda, compuesta por ocho capas, que revolucionó el campo de la visión por computadora y el aprendizaje profundo.

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

El uso de *patches* grandes permite que la red considere más contexto para predecir la salida, pero la aplicación de la operación de *max-pooling* (véase el [Apéndice A](#) para ver el significado de la operación de *max-pooling*) para reducir su dimensionalidad resulta en la pérdida de detalles. Es decir, la red se vuelve invariante a pequeñas traslaciones espaciales de las características debido al submuestreo agresivo³. Esto es útil en tareas donde la precisión de la localización no es crítica.

Por otro lado, si se utilizan *patches* pequeños, se mejora la precisión de la localización pero se reduce el contexto empleado por la red para realizar predicciones.

Desde entonces, han surgido modelos que buscan abordar estos inconvenientes, buscando un equilibrio entre una buena localización y el uso eficiente del contexto. La arquitectura *U-Net* es una propuesta de solución a este deseado equilibrio.

5.2.2. Estructura de la arquitectura *U-Net*

Ronneberger y cols. (2015), en su investigación ”*U-net: Convolutional networks for biomedical image segmentation*” (”*U-net: redes convolucionales para segmentación de imágenes biomédicas*”), desarrolla una arquitectura en forma de U basada en la red completamente convolucional (*fully convolutional network*) presentada por Ciresan y cols. (2012). Propone una modificación de esta arquitectura convolucional con el objetivo de operar eficazmente con pocos datos de entrenamiento y producir segmentaciones más precisas en imágenes biomédicas.

La idea principal es reemplazar algunas operaciones de *pooling* por *upsampling*⁴, aumentando así la resolución de salida, como se observa en la Figura 5.5. Para lograr la localización, la alta resolución de las características provenientes de la fase de contracción, o *contracting path* en inglés, que corresponde a la primera mitad izquierda del modelo reflejado en la Figura 5.5, se combina con la salida de la operación de

³**Submuestreo agresivo:** situación en la que se reduce significativamente la resolución de entrada, lo que resulta en la pérdida de detalles de características en el proceso. Esta estrategia puede ser beneficiosa cuando se busca capturar características de alto nivel al mismo tiempo que se reduce la carga computacional.

⁴**Upsampling:** proceso utilizado en el procesamiento de imágenes y en el aprendizaje profundo para aumentar la resolución espacial de una imagen.

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

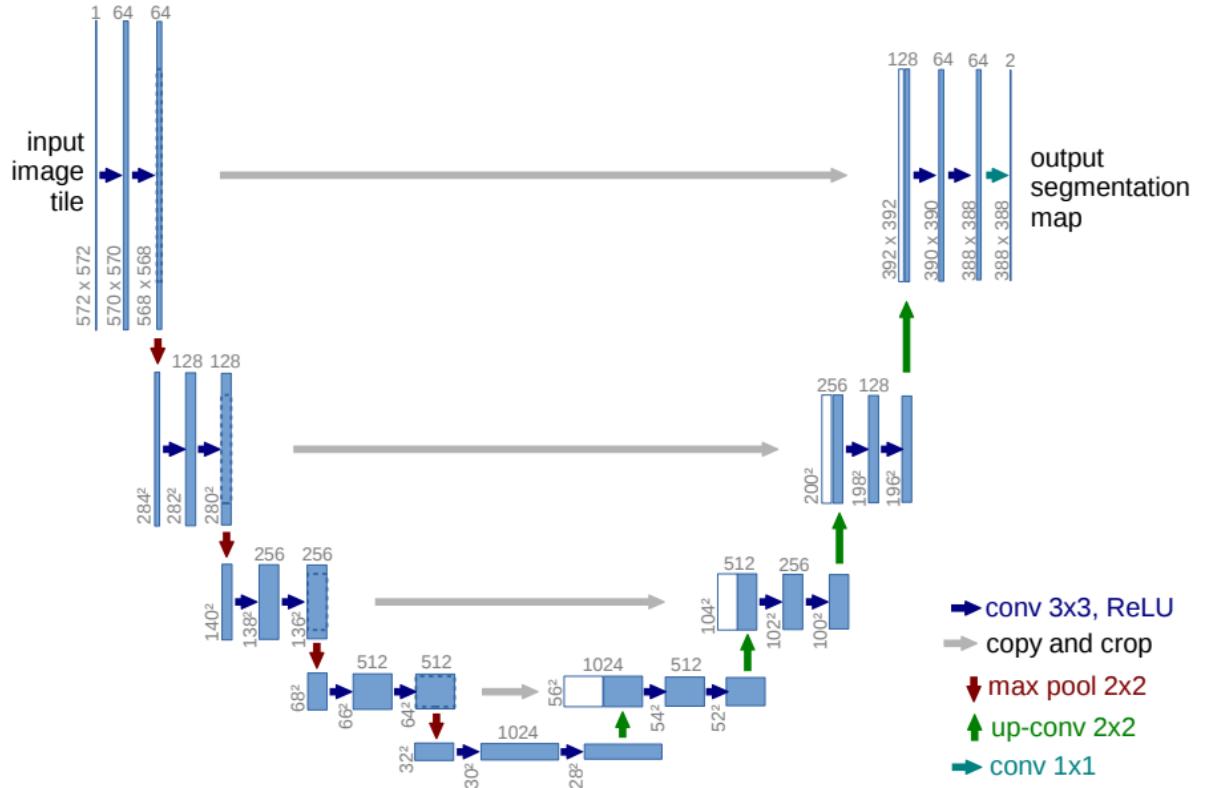


Figura 5.5: Arquitectura *U-Net*. Fuente: [Ronneberger y cols. \(2015\)](#)

upsampling. Una capa convolucional sucesiva puede aprender a producir una salida más precisa basada en esta información. Esta conexión entre las capas de contracción y las capas de *upsampling* se representa con las flechas grises en la Figura 5.5.

Una modificación importante presentada por [Ronneberger y cols. \(2015\)](#) se produce en la parte de *upsampling*: se generan mapas de características tan profundos que permiten a la red propagar información de contexto a capas de resolución más alta (cada píxel de la capa de segmentación de salida tiene suficiente información de contexto obtenida en la fase de contracción). Como resultado, la fase de expansión, o *expansive path* en inglés, que corresponde a la segunda mitad derecha del modelo reflejado en la Figura 5.5, es simétrica con la fase de contracción, estableciendo una forma de U.

La Figura 5.5 exhibe la arquitectura *U-Net* propuesta por [Ronneberger y cols. \(2015\)](#), resumida en dos componentes: un codificador (*encoder*) y un decodificador

(*decoder*), que se pueden resumir como:

1. **Codificador:** constituye la fase de contracción, o *contracting path*. Está formado por una secuencia de pares de capas convolucionales y *max-pooling* (véase el [Apéndice A](#) para obtener detalles sobre ambas operaciones) para extraer las características relevantes de las imágenes generadas a partir de las series temporales de datos.
2. **Decodificador:** también conocido como *expanding path*, o fase de expansión, y se encarga de reconstruir una imagen de salida segmentada mediante el conjunto de características extraídas del codificador. Cada etapa de la fase de expansión consiste en una operación convolucional transpuesta (véase el [Apéndice A](#) para conocer la operación de convolución transpuesta), que reduce el número de canales y aumenta la dimensión espacial, una concatenación con su correspondiente mapa de características de la fase de contracción y dos capas convolucionales adicionales.

En la operación de concatenación, representada en la Figura 5.5 por la flecha gris horizontal, la salida de la convolución transpuesta 2×2 se concatena con la característica correspondiente de la fase de contracción. La concatenación se realiza a lo largo de la dimensión de los canales. Por ejemplo, si la característica de *upsampling* tiene, digamos, 128 canales, y la característica de la fase de contracción tiene 256 canales, la concatenación se realizará para obtener una característica combinada de 384 canales ($128 + 256$).

La idea de las capas convolucionales detrás de la operación de concatenación es afinar las características en una mayor resolución en esa etapa de expansión. Gracias a la previa concatenación, se buscan detalles que podrían haberse perdido en la contracción, manteniendo al mismo tiempo una resolución de salida alta. La arquitectura finaliza con una convolución 1×1 , que actúa como clasificador para la tarea de segmentación de imágenes biomédicas presentada por [Ronneberger y cols. \(2015\)](#).

5.2.3. Arquitectura *U-Net* previa

Como se ha descrito al inicio del apartado actual ([5.2 Sistema de autoetiquetado. Arquitectura U-Net](#)), esta primera fase de la red busca automatizar el proceso de etiquetado de las imágenes convertidas que representan señales, conformando así un nuevo conjunto de datos que consiste en una secuencia temporal de etiquetas. A medida que llegan nuevas series temporales de datos procedentes del parque eólico, los datos se clasifican mediante la arquitectura *U-Net* presente. Esta fase de clasificación es esencial para la identificación y extracción de características relevantes de los datos.

Se parte de lo realizado por [Cambero Rojas \(2023\)](#) en su Trabajo Fin de Grado “Ajuste del diseño de una red neuronal *U-Net* para la predicción de comportamientos anómalos en turbinas eólicas”. En él, se estudia diferentes vías para conseguir un modelo óptimo que minimice el sobreajuste⁵ que hasta el momento se había obtenido en trabajos previos, sin menoscabo de los buenos resultados a la hora de predecir el rendimiento asociado a una imagen sintética.

En el estudio de [Cambero Rojas \(2023\)](#) se describe el diseño del modelo y su configuración, ideal para nuestro caso de estudio. La Figura [5.6](#) visualiza la arquitectura *U-Net* propuesta. Esta red cuenta con 4 niveles de profundidad. El codificador está compuesto por operaciones de convolución 2D con una ventana, o *kernel*, de tamaño 3×3 y operaciones de *max-pooling*. A medida que los datos avanzan por el codificador, se reduce el tamaño de los mapas de características y se aumenta progresivamente la profundidad hasta alcanzar 512 mapas de características en la última capa. La disminución de la dimensión de los mapas de características se logra mediante operaciones de *max-pooling*. La primera operación de *max-pooling* reduce a la mitad la altura y la anchura, la segunda operación reduce la altura a la mitad, y la tercera operación reduce a la mitad la anchura, generando un mapa de características de tamaño $4 \times 4 \times 512$.

En el decodificador, a medida que se avanza en las capas, se concatenan

⁵**Sobreajuste:** ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y no generaliza adecuadamente con nuevos datos no vistos previamente por el modelo. Es uno de los problemas más comunes en los modelos de clasificación.

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

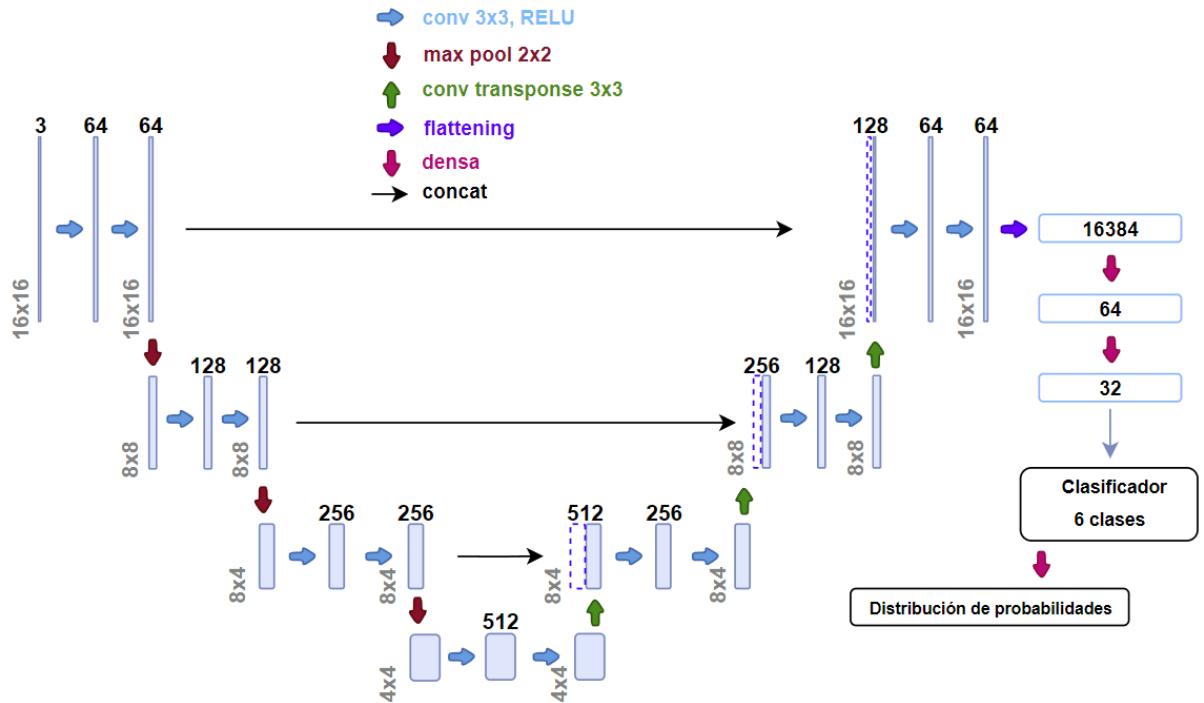


Figura 5.6: Arquitectura U-Net utilizada como punto de partida. Fuente: Cambero Rojas (2023)

los resultados obtenidos de la convolución correspondiente en el codificador con el resultado de la convolución transpuesta 2D. La convolución transpuesta busca aumentar la resolución de los mapas de características con el fin último de generar una imagen segmentada que pueda contener información identificativa sobre el estado de rendimiento de la turbina eólica. En la parte final del decodificador se genera un mapa de características manteniendo la resolución inicial de 16×16 con una profundidad de 64. Se aplica una capa de aplanamiento⁶ o *flattening* para convertir los mapas de características en un vector de números que pueda ser procesado por 2 capas densas de 64 y 32 neuronas respectivamente. Finalmente, el resultado intermedio sirve de entrada a una capa densa con 6 neuronas y una función de activación *softmax*, la cual es un clasificador que genera un vector de probabilidades de pertenencia a cada clase.

Algunos aspectos esenciales que contribuyen al buen rendimiento de esta

⁶**Aplanamiento:** o *flattening*, es el proceso de transformar un tensor multidimensional en un vector unidimensional, generalmente realizado en la capa de entrada de una red completamente densa o previo a una capa densa.

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

arquitectura propuesta por [Cambero Rojas \(2023\)](#) y los hiperparámetros asociados son los siguientes:

- El tensor de entrada de la *U-Net* tiene un tamaño de $16 \times 16 \times 3$, siendo 3 las características más influyentes en el rendimiento de una turbina eólica: la potencia activa media, la velocidad del viento y el ángulo de pitch (véase el apartado [4.2 Modelo físico de una turbina eólica](#) para conocer el proceso de selección de estas 3 características).
- Para prevenir el sobreajuste (*overfitting*) de la red, se incluyen capas *dropout* con un valor de 0,4 entre las capas convolucionales, las convoluciones transpuestas y la capa densa.
- En el proceso de entrenamiento se utiliza el optimizador *Adam* (véase el [Apéndice A](#) para entrar en detalle en el proceso de entrenamiento de una red neuronal). Este optimizador, a diferencia de otros relacionados con el descenso del gradiente, ajusta automáticamente la tasa de aprendizaje individualmente para cada parámetro de la red, lo que conduce a un entrenamiento más estable y rápido en la convergencia hacia un mínimo de la función de pérdida.
- La función de pérdida (*loss*) empleada es la entropía categórica cruzada, o *categorical crossentropy*, esencial para problemas de clasificación multiclas. Mide la discrepancia entre las distribuciones de probabilidad predichas por el modelo y la distribución real, y se expresa mediante la Ecuación [5.1](#):

$$\text{Categorical Crossentropy} = - \sum_{c=1}^C y_{\text{true},c} \log(y_{\text{pred},c}) \quad (5.1)$$

- Durante el entrenamiento empleamos las métricas de exactitud y precisión (véase el [Apéndice A](#) para obtener una descripción más detallada de ambas métricas).

La arquitectura *U-Net* descrita, utilizada para el diagnóstico de anomalías en una turbina eólica, ofrece resultados con una exactitud y precisión del 93,2% y 93,8%,

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

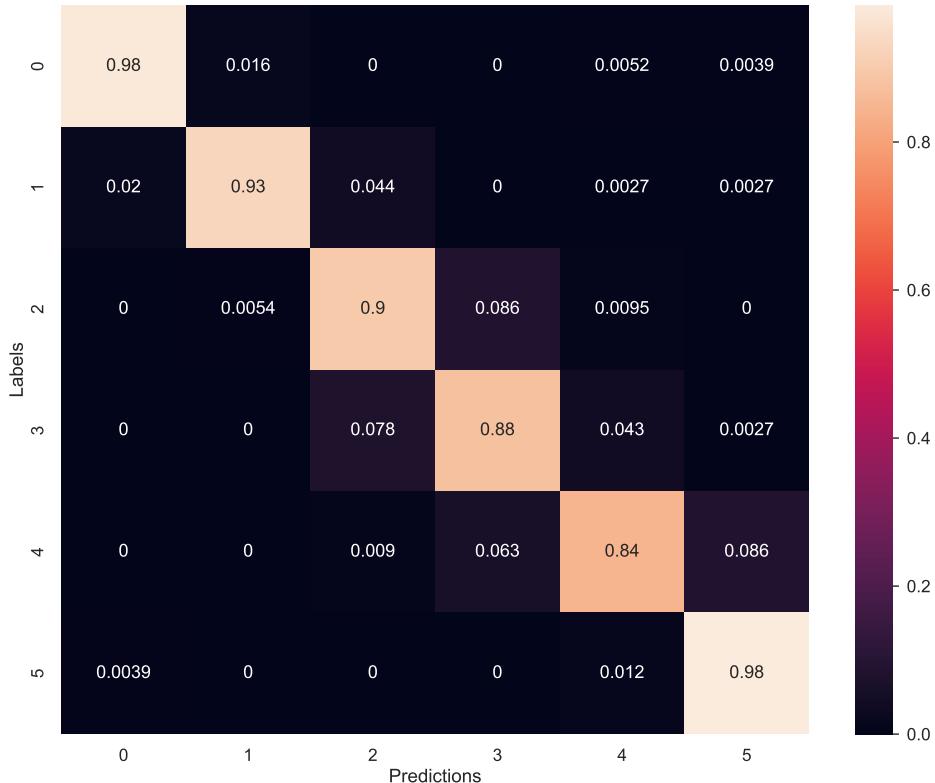


Figura 5.7: Matriz de confusión del modelo *U-Net* tomado como punto de partida.
Fuente: [Cambero Rojas \(2023\)](#)

respectivamente. La Figura 5.7 nos muestra la *matriz de confusión* asociada al modelo descrito. Se puede observar que el diagnóstico de las diferentes clases de rendimiento de la turbina eólica se aproximan en su diagonal entre el 90 % y el 95 %.

Por último, es importante quedar reflejada la estructura completa de la red neuronal *U-Net* diseñada por [Cambero Rojas \(2023\)](#), que sirve de punto de partida para los modelos diseñados posteriormente. La Figura 5.8 la representa de manera precisa.

5.2. SISTEMA DE AUTOETIQUETADO. ARQUITECTURA U-NET

Model: "UNet"				
Layer (type)	Output Shape	Param #	Connected to	
input_1 (InputLayer)	[(None, 16, 16, 3)]	0	[]	
EncodeConv1_1 (Conv2D)	(None, 16, 16, 64)	1792	['input_1[0][0]']	
dropout (Dropout)	(None, 16, 16, 64)	0	['EncodeConv1_1[0][0]']	
EncodeConv1_2 (Conv2D)	(None, 16, 16, 64)	36928	['dropout[0][0]']	
EncodeAveragePool1 (MaxPooling2D)	(None, 8, 8, 64)	0	['EncodeConv1_2[0][0]']	
BridgeConv_1 (Conv2D)	(None, 8, 8, 128)	73856	['EncodeAveragePool1[0][0]']	
dropout_1 (Dropout)	(None, 8, 8, 128)	0	['BridgeConv_1[0][0]']	
BridgeConv_2 (Conv2D)	(None, 8, 8, 128)	147584	['dropout_1[0][0]']	
EncodeAveragePool2 (MaxPooling2D)	(None, 8, 4, 128)	0	['BridgeConv_2[0][0]']	
BridgeConv_3 (Conv2D)	(None, 8, 4, 256)	295168	['EncodeAveragePool2[0][0]']	
dropout_2 (Dropout)	(None, 8, 4, 256)	0	['BridgeConv_3[0][0]']	
BridgeConv_4 (Conv2D)	(None, 8, 4, 256)	590080	['dropout_2[0][0]']	
EncodeAveragePool3 (MaxPooling2D)	(None, 4, 4, 256)	0	['BridgeConv_4[0][0]']	
BridgeConv_5 (Conv2D)	(None, 4, 4, 512)	1180160	['EncodeAveragePool3[0][0]']	
dropout_3 (Dropout)	(None, 4, 4, 512)	0	['BridgeConv_5[0][0]']	
BridgeConv_6 (Conv2D)	(None, 4, 4, 512)	2359808	['dropout_3[0][0]']	
DecodeUp3 (Conv2DTranspose)	(None, 8, 4, 256)	1179984	['BridgeConv_6[0][0]']	
Concat3 (Concatenate)	(None, 8, 4, 512)	0	['DecodeUp3[0][0]', 'BridgeConv_4[0][0]']	
DecodeConv3_1 (Conv2D)	(None, 8, 4, 256)	1179984	['Concat3[0][0]']	
dropout_4 (Dropout)	(None, 8, 4, 256)	0	['DecodeConv3_1[0][0]']	
DecodeConv3_2 (Conv2D)	(None, 8, 4, 256)	590080	['dropout_4[0][0]']	
DecodeUp2 (Conv2DTranspose)	(None, 8, 8, 128)	295040	['DecodeConv3_2[0][0]']	
Concat2 (Concatenate)	(None, 8, 8, 256)	0	['DecodeUp2[0][0]', 'BridgeConv_2[0][0]']	
DecodeConv2_1 (Conv2D)	(None, 8, 8, 128)	295040	['Concat2[0][0]']	
dropout_5 (Dropout)	(None, 8, 8, 128)	0	['DecodeConv2_1[0][0]']	
DecodeConv2_2 (Conv2D)	(None, 8, 8, 128)	147584	['dropout_5[0][0]']	
DecodeUp1 (Conv2DTranspose)	(None, 16, 16, 64)	73792	['DecodeConv2_2[0][0]']	
Concat1 (Concatenate)	(None, 16, 16, 128)	0	['DecodeUp1[0][0]', 'EncodeConv1_2[0][0]']	
DecodeConv1_1 (Conv2D)	(None, 16, 16, 64)	73792	['Concat1[0][0]']	
dropout_6 (Dropout)	(None, 16, 16, 64)	0	['DecodeConv1_1[0][0]']	
DecodeConv1_2 (Conv2D)	(None, 16, 16, 64)	36928	['dropout_6[0][0]']	
Flattening (Flatten)	(None, 16384)	0	['DecodeConv1_2[0][0]']	
dense (Dense)	(None, 64)	1048640	['Flattening[0][0]']	
dropout_7 (Dropout)	(None, 64)	0	['dense[0][0]']	
dense_1 (Dense)	(None, 32)	2080	['dropout_7[0][0]']	
dropout_8 (Dropout)	(None, 32)	0	['dense_1[0][0]']	
Classifier (Dense)	(None, 6)	198	['dropout_8[0][0]']	

Total params: 9608358 (36.65 MB)
Trainable params: 9608358 (36.65 MB)
Non-trainable params: 0 (0.00 Byte)

Figura 5.8: Arquitectura por capas U-Net tomada como punto de partida. Fuente: Cambero Rojas (2023)

5.3. MODELADO BASELINE INICIAL

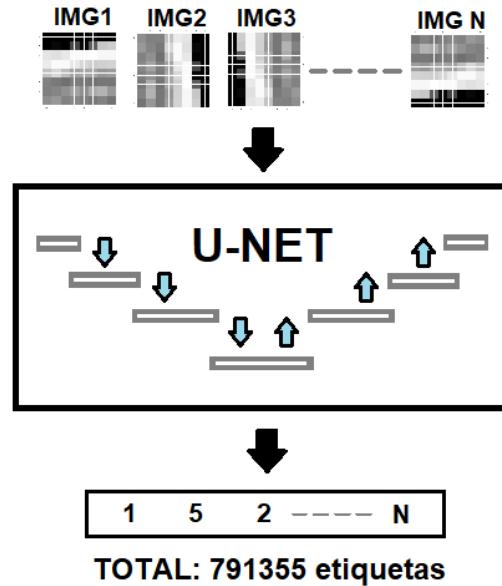


Figura 5.9: Proceso de autoetiquetado realizado por medio de la *U-Net*.

5.3. Modelado *baseline* inicial

En un proyecto de modelado mediante Deep Learning es crucial comenzar con un primer modelo inicial simple como punto de referencia, o *baseline*⁷. Para ello, se parte de la arquitectura *U-Net* descrita en la sección 5.2.3 Arquitectura *U-Net* previa.

5.3.1. Generación de la serie temporal de etiquetas

Como se ha explicado en las secciones previas, la arquitectura *U-Net* cumple la función de generar un conjunto de datos intermedio que representa una serie temporal de etiquetas asociadas a cada imagen. La Figura 5.9 refleja la generación de un total de 791.355 etiquetas del conjunto de imágenes.

La Tabla 5.2 presenta una comparativa entre el recuento de etiquetas reales y el recuento de etiquetas predichas por el modelo para cada imagen, recopilando el número de etiquetas asociadas a las 25 turbinas del parque eólico monitorizado entre 2009 y

⁷**Baseline:** en el contexto de la investigación y desarrollo de modelos, un modelo ”*baseline*” se refiere a un punto de referencia inicial o estándar utilizado para comparar y evaluar el rendimiento de modelos más complejos.

5.3. MODELADO BASELINE INICIAL

Etiqueta	Reales	Generadas
0 - Excesivo Rendimiento	57424	93226
1 - Óptimo Rendimiento	678981	609051
2 - Bajo Rendimiento	44104	72229
3 - Bajo Rendimiento Notable	3532	6602
4 - Problema Serio	2157	3517
5 - Parada	5157	6730

Tabla 5.2: Comparación entre las etiquetas reales de las imágenes y las generadas por el modelo *U-Net* para las 25 turbinas eólicas.

2017. El nuevo conjunto de datos contiene 25 series temporales de etiquetas, una por cada turbina monitorizada. Las turbinas están instaladas bajo condiciones diferentes. Entrenar un modelo con varias series temporales puede dar lugar a un resultado pésimo debido a que cada turbina opera bajo condiciones de terreno y orientación de las palas diferentes. Esto, sumado a los límites de capacidad de cómputo del *hardware*, llevó a decidir trabajar únicamente con la serie temporal de la turbina más representativa para el modelado.

5.3.2. División del conjunto de datos

En el [Apéndice B](#) se presenta el análisis mediante el cual se ha elegido la serie temporal de la turbina eólica identificada por el ID 2 (aerogenerador *WT2*). En resumen, la turbina eólica *WT2* presenta una mayor distribución de etiquetas anómalas en el año 2016, el cual se utilizará para la fase de prueba. La Tabla 5.3 refleja el número de muestras de la serie temporal de la turbina eólica *WT2* que tiene cada etiqueta en los años 2016 y 2017.

Se observa una mayor presencia de las etiquetas 2, 3, 4, 5 en el año 2016, motivo por el que se ha tomado la decisión de particionar el *dataset* para los dos modelos *baseline* en dos conjuntos:

1. **Conjunto de datos de entrenamiento:** recoge los pasos de tiempo de la serie temporal de la *WT2* desde 2009 hasta 2015, inclusive.

5.3. MODELADO BASELINE INICIAL

Etiqueta	2016	2017
0 - Excesivo Rendimiento	471	271
1 - Óptimo Rendimiento	4908	794
2 - Bajo Rendimiento	85	1
3 - Bajo Rendimiento Notable	15	0
4 - Problema Serio	58	1
5 - Parada	172	7

Tabla 5.3: Distribución de las etiquetas en la serie temporal de la turbina eólica WT2 en los años 2016 y 2017.

2. **Conjunto de datos de prueba:** constituye los pasos de tiempo de la serie temporal de etiquetas de la WT2 en 2016.

Sobre la serie temporal de etiquetas generada de la turbina WT2, se evaluarán dos modelos iniciales para determinar si son capaces de abordar el problema de detección de anomalías mediante un enfoque básico; lo que ayudará a determinar si se requiere la implementación de modelos más complejos para una solución efectiva.

5.3.3. Modelo de regresión con WEKA

Un modelo de regresión es un algoritmo estadístico que permite analizar la relación entre dos o más variables, considerando una de ellas como dependiente del resto. Su objetivo es determinar cómo las variables independientes afectan a una variable dependiente. El ejemplo más simple de regresión es la regresión lineal.

En un modelo de regresión lineal, se establece la correlación entre la variable independiente y las variables dependientes como se define en la Ecuación 5.2:

$$Y = \beta_0 + \beta_1 X_1 + \dots + \beta_m X_m + \varepsilon \quad (5.2)$$

donde:

- Y es la variable dependiente o variable de respuesta.
- X_1, X_2, \dots, X_m son las variables independientes.

5.3. MODELADO BASELINE INICIAL

- $\beta_0, \beta_1, \beta_2, \dots, \beta_m$ son los parámetros del modelo, miden la influencia que las variables independientes tienen sobre la variable dependiente.

Existen otros modelos de regresión que son capaces de capturar correlaciones más complejas entre los datos de entrada, como la regresión múltiple, exponencial, entre otros. El modelo regresivo *baseline* que se presenta en este trabajo se ha desarrollado utilizando el *software* de código abierto *WEKA*⁸.

El primer modelo regresivo creado con *WEKA* se configura de tal forma que prediga la siguiente etiqueta futura posterior a la serie temporal de etiquetas de la *WT2*.

Los aspectos más importantes de la configuración de proceso de entrenamiento en la regresión de *WEKA* son los siguientes:

- Número de unidades de tiempo a predecir: 1.
- *Lag lengths*: se refiere al número de pasos de tiempo anteriores que se utilizan como entradas para predecir el valor en el siguiente paso de tiempo. Dado que la serie temporal está compuesta por valores discretos que fluctúan en lugar de presentar continuidad, deberíamos considerar las etiquetas como una variable independiente del tiempo, ya que no dependen de n valores del pasado, sino de la serie completa. Sin embargo, en el modelo de regresión lineal de *WEKA*, es necesario indicar un rango de pasos pasados como las variables independientes sobre las que se ajustará la curva para predecir el siguiente valor. En concreto, se establecen 12 pasos pasados.
- Se eligen las métricas *MAE* y *RMSE* para evaluar al modelo durante la fase de entrenamiento y la fase de prueba (véase el [Apéndice A](#) para conocer en las expresiones matemáticas de las métricas).

⁸**WEKA**: es una plataforma de *software* de código abierto que ofrece un conjunto de algoritmos de *Machine Learning* diseñados para abordar problemas de minería de datos del mundo real

5.3. MODELADO BASELINE INICIAL

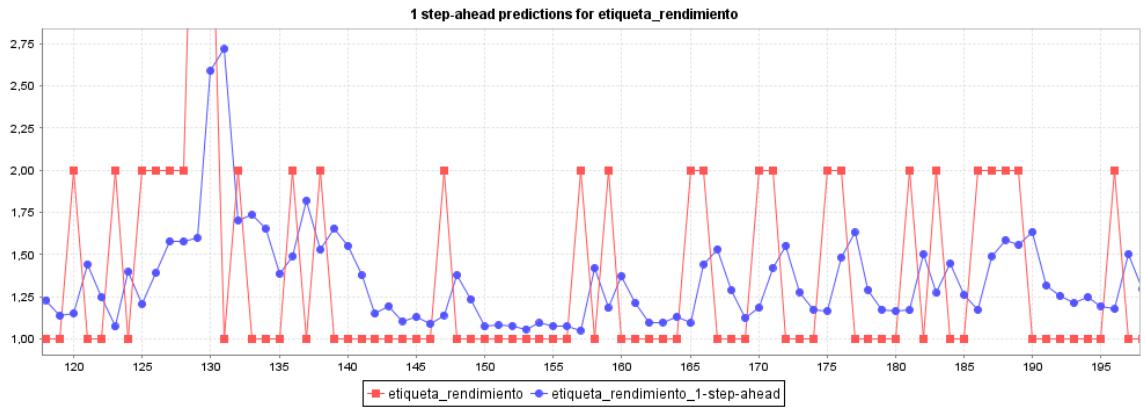


Figura 5.10: Comportamiento del modelo del modelo de regresión con WEKA en un intervalo de tiempo concreto del conjunto de datos del año 2016.

Resultado

Los resultados de las métricas obtenidas con el conjunto de entrenamiento muestran un *MAE* de 0,313 y un *RMSE* de 0,566. Por otro lado, las métricas correspondientes a la fase de prueba son un *MAE* de 0,31 y un *RMSE* de 0,57.

Es crucial analizar qué ocurre con la predicción de un valor futuro a lo largo del conjunto de prueba. En la Figura 5.10 se presenta el comportamiento del modelo regresivo en el conjunto de prueba. La línea roja corresponde a las etiquetas originales del conjunto de prueba, mientras que la linea azul son las predicciones del modelo. Se visualiza un patrón en la tendencia creciente y decreciente de las etiquetas predichas retrasado en un paso de tiempo. Entre las muestras 145 – 150 se observa que la curva de predicción comienza a ascender una vez la entrada del modelo regresivo contenía el pico intermedio con etiqueta real 2. Este patrón es observable a lo largo del conjunto de prueba. El modelo predice con un retraso de un paso futuro.

5.3.4. Modelos de regresión con red neuronal *LSTM*

5.3.4.1. Red neuronal *LSTM* no stateful

Se construye una primera aproximación de regresión basado en red neuronal con un modelo de red neuronal recurrente con capas *LSTM* (el Apéndice A incluye todos los conceptos que ayudan a comprender el funcionamiento de una red neuronal, entre el

5.3. MODELADO BASELINE INICIAL

que se encuentra el de capa *LSTM*), porque las capas *LSTM* almacenan una memoria interna que se actualiza a medida que se procesan los elementos de una secuencia de datos. La ventaja que ofrecen estas capas es su capacidad de aprender patrones temporales más complejos que el del modelo de regresión clásico de *ML*.

Debido a la técnica de solapamiento explicada en el apartado [5.1.1 Conversión de las series temporales de datos a imágenes](#), y como se muestra en la Figura [5.2](#) del mismo apartado, la secuencia de la figura sin solapamiento enmarca un periodo de tiempo de $(6 \times 11 \text{ horas}) = 66 \text{ horas}$ por imagen. De forma generalizada, se recoge con la expresión $T_{\text{sec}} = (11h \times \text{long}_{\text{sec}})$.

Con solapamiento, la secuencia de 6 imágenes contiene un periodo de tiempo de 38,5 horas; expresado convenientemente: $T_{\text{solap}} = \left(\frac{11 \text{ horas} \times \text{long}_{\text{sec}}}{2} \right) + \frac{11 \text{ horas}}{2}$.

Al trabajar con series temporales de datos, es crucial contextualizar cuál es el periodo de tiempo que abarca una secuencia de imágenes, dado que una sola imagen recoge datos procedentes de un intervalo de tiempo de 11 horas. La serie temporal completa de etiquetas de la turbina se divide en secuencias de 65 *timestamps*. Aplicando la fórmula anterior, una secuencia de 65 pasos de tiempo equivale a 15,25 días.

Los aspectos mas importantes del modelo LSTM *baseline* creado son los siguientes (véase el [Apéndice A](#) para profundizar en los diferentes hiperparámetros):

- Solapamiento entre secuencias: la serie temporal de etiquetas del conjunto de entrenamiento se fragmenta en secuencias con un solapamiento de 10 pasos de tiempo. Por ejemplo, si la primera secuencia es [1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 3...], la segunda secuencia sería [2, 2, 3...].
- Número de periodos (épocas): se establecen 15 *epochs* en el proceso de entrenamiento.
- Tamaño del lote (*batch size*): se refiere al número de muestras de datos que se utilizan en cada iteración durante el entrenamiento de un modelo. Por defecto, se usará 64.

5.3. MODELADO BASELINE INICIAL

- *Dropout*: Semeniuta y cols. (2016) estudian el impacto del hiperparámetro de *dropout* en la memoria interna de las capas *LSTM*. Determinan que un valor de *dropout* de 0,25 es crucial en el proceso de entrenamiento de una red neuronal recurrente para evitar el sobreajuste.
- La secuencia de etiquetas se normaliza mediante el algoritmo *MinMaxScaler*, proporcionado por la biblioteca *scikit-learn*⁹. La mayoría de las técnicas de predicción ofrecen mejores resultados cuando operan con datos normalizados. Las unidades *LSTM* realizan transformaciones no lineales, como la función *sigmoide* y la función *tangente hiperbólica (tanh)* para manejar la memoria interna. Cuando los datos de entrada tienen valores grandes, ambas funciones pueden causar una pérdida de información durante el procesamiento de una secuencia.
- Una opción que presentan las capas *LSTM* es mantener su memoria interna. La opcionalidad se determina mediante el parámetro *stateful*. La documentación oficial del marco de trabajo *Keras* explica que si *stateful* toma un valor *True*, el último estado para cada muestra de índice *i* de un *batch* será usado como estado inicial para la muestra de índice *i* del siguiente *batch*. En el modelo creado que nos ocupa, se establece *stateful* a *False* porque consideramos que la longitud de secuencia de 15 días es suficiente para predecir el siguiente valor.
- Se elige la métrica *MAE* para evaluar la evolución de las etapas en el entrenamiento.

La Figura 5.11 recoge la arquitectura completa *LSTM* no *stateful*, o carente de memoria interna, junto con el número de parámetros entrenables de cada capa.

Resultado

Durante la fase de prueba, se pronostican varios valores futuros basados en una secuencia de etiquetas con el propósito de analizar la tendencia de los valores

⁹*Scikit-learn*: biblioteca *Python* de aprendizaje automático que proporciona una amplia gama de algoritmos de análisis de datos y de modelado

5.3. MODELADO BASELINE INICIAL

```

Model: "LSTM No stateful baseline"

Layer (type)          Output Shape         Param #
=====
input_2 (InputLayer)   [(None, 65, 1)]      0
lstm_2 (LSTM)          (None, 65, 32)       4352
batch_normalization_1 (BatchNormalization) (None, 65, 32)   128
lstm_3 (LSTM)          (None, 64)            24832
dropout_1 (Dropout)    (None, 64)            0
dense_2 (Dense)        (None, 32)           2080
dense_3 (Dense)        (None, 1)             33
=====
Total params: 31425 (122.75 KB)
Trainable params: 31361 (122.50 KB)
Non-trainable params: 64 (256.00 Byte)

```

Figura 5.11: Arquitectura *LSTM no stateful baseline* inicialmente propuesta.

pronosticados. Utilizando una secuencia previa de 65 etiquetas del conjunto de prueba, se anticipan los próximos 10 valores futuros. Como el modelo ha sido entrenado para predecir el siguiente valor de la secuencia de entrada, para pronosticar una secuencia de valores se añade la última etiqueta pronosticada al final de la secuencia de entrada y se elimina la primera etiqueta de forma iterativa, hasta predecir las siguientes 10 etiquetas futuras.

En la Figura 5.12, la línea gráfica roja representa la secuencia real de etiquetas de la WT2; la línea gráfica azul muestra varias secuencias pronosticadas. Recordemos que la mayoría de las etiquetas en la secuencia de entrenamiento tienen el valor 1, etiqueta asociada al rendimiento óptimo de la turbina eólica. Atendiendo a la visualización obtenida con la Figura 5.12, se comprueba que el modelo nunca podrá predecir con precisión valores de etiquetas distintos a 1 debido al sobreajuste del entrenamiento. Como se ha mencionado anteriormente, es importante analizar la tendencia ascendente o descendente de las secuencias pronosticadas para verificar si hay una correlación con las reales. Se concluye que las secuencias pronosticadas en azul no muestran una

5.3. MODELADO BASELINE INICIAL

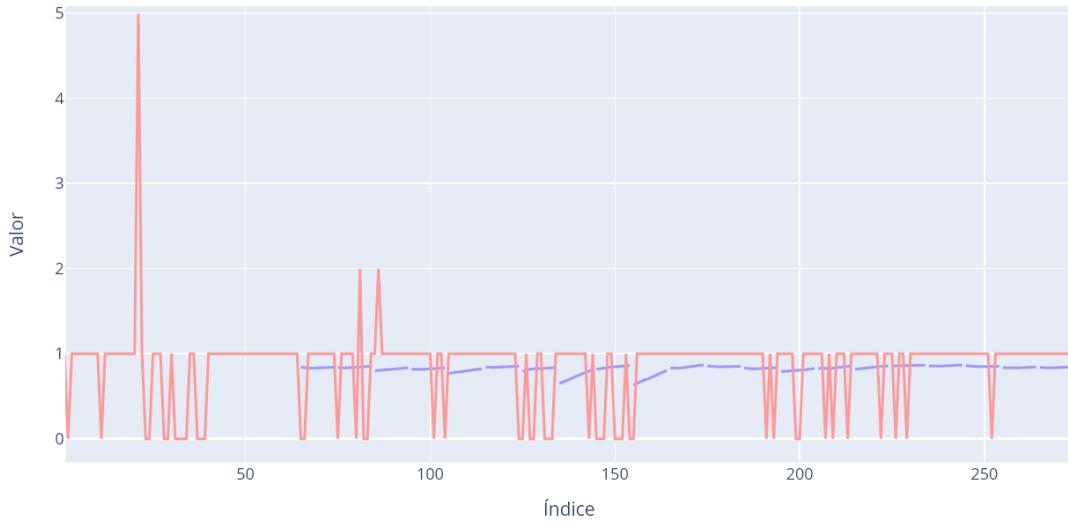


Figura 5.12: Predicción de secuencias de 10 valores futuros con el modelo *LSTM* no *stateful baseline* sobre el conjunto de datos del año 2016.

tendencia clara respecto a las reales; esta aproximación *baseline* no *stateful* arroja un resultado deficiente.

El modelo de regresión *baseline* experimentado con *WEKA* se entrena con la secuencia completa de entrenamiento de una sola vez y devuelve un resultado mejor que el de la red neuronal *LSTM* no *stateful*, este último presenta un comportamiento cercano a lo aleatorio. Por ello, el siguiente paso fue probar el modelo *LSTM stateful* en modo diferido.

5.3.4.2. Red neuronal *LSTM stateful* en diferido

En vistas de los resultados anteriores, se diseña un modelo de red neuronal recurrente (RNN) con las capas *LSTM* y arquitectura similar al *baseline* previo, pero con en el parámetro *stateful* activado; esto es, manteniendo la memoria interna de procesamientos anteriores. En lugar de indicar a la red el número de pasos del pasado con los que se desea entrenar el modelo para predecir el siguiente valor de la secuencia, se define *stateful* como *True*, y se establece el tamaño de lote de secuencias (*batch size*) a 1, con secuencias de muestras de longitud 65, para que la red neuronal se entrene con toda la secuencia de entrada de forma ordenada, inicializando el estado interno de las

5.3. MODELADO BASELINE INICIAL

capas *LSTM* en base al estado interno resultante del procesamiento del *batch* anterior. De esta manera, la red neuronal será autónoma para modificar su memoria interna a medida que procesa la secuencia completa de entrenamiento, ajustando sus pesos internos para encontrar dependencias temporales a largo plazo que no dependan de un número fijo de pasos de tiempo pasados.

A modo de ejemplo, imagine que mi serie temporal del conjunto de entrenamiento tiene las siguientes 8 etiquetas $[1, 1, 1, 1, 0, 1, 5, 2]$, y la longitud de las secuencias con las que voy a entrenar el modelo es de tamaño 4. En primer lugar, el modelo se entrenaría con la primera secuencia $[1, 1, 1, 1]$, almacenando un estado en su memoria interna. A continuación, ajustaría sus pesos procesando la segunda secuencia $[0, 1, 5, 2]$ a partir del estado almacenado en su memoria interna al procesar la primera secuencia.

A modo de resumen, los aspectos más destacables del modelo *stateful* creado con respecto al anterior son los siguientes:

- Tamaño del lote (*batch size*): se asigna 1 al valor de número de secuencias por lote. Para mantener el orden de la serie temporal de entrenamiento, cada lote debe tener una sola secuencia; de esta forma, cada secuencia se procesa con la información almacenada en el estado interno de las secuencias del pasado.
- Parámetro *stateful*: se activa con *True* para preservar la memoria interna de las capas *LSTM* entre lotes sucesivos.
- Número de *epochs*: se establecen 5 etapas en el proceso de entrenamiento. En cada etapa, se procesa la serie temporal completa de la *WT2*. No es necesario repetir el entrenamiento porque el modelo se ajustaría a los datos.

Resultado

La fase de prueba es similar al modelo *no stateful* anterior. Dada una secuencia de tamaño 65 del conjunto de prueba, se predicen los siguientes 10 valores futuros con el fin de monitorear la tendencia de la predicción.

El inconveniente que presenta este modelo es que, para predecir uno o varios valores futuros, es imprescindible mantener consistente la memoria interna de las

5.3. MODELADO BASELINE INICIAL

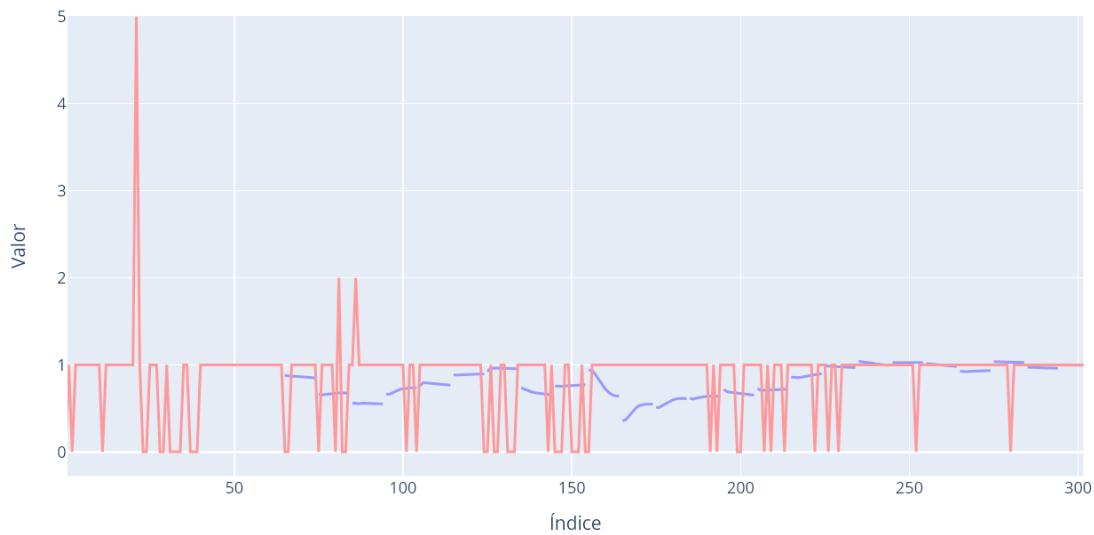


Figura 5.13: Predicción de secuencias de 10 valores futuros con el modelo *LSTM stateful baseline* sobre el conjunto de datos del año 2016.

unidades *LSTM*. Si se desea predecir una etiqueta futura a partir de la segunda secuencia del conjunto de prueba, se necesita volver a entrenar el modelo con la primera secuencia del conjunto de prueba incrustada como la última secuencia del conjunto de entrenamiento. En otras palabras, a medida que llegan nuevas etiquetas de la *WT2* del parque eólico, es necesario entrenar el modelo con toda la serie temporal del pasado. A este tipo de modelo se le denomina “*LSTM* en diferido”.

El coste necesario para entrenar el modelo a medida que avanza el tiempo es muy elevado, lo que dificulta su implementación en producción. Por este motivo, en vez de alimentar el modelo con toda la secuencia de entrenamiento de una sola vez, se decide implementar un tipo de prueba llamada *walk-forward validation* (validación hacia adelante). Este mecanismo permite reentrenar el modelo con la secuencia de datos de prueba que se ha utilizado en la iteración previa. Cuando el modelo predice 10 pasos futuros, los datos reales de esos pasos futuros están disponibles para el modelo, de modo que se pueden utilizar como base para predecir los siguientes valores.

Por otro lado, al analizar la gráfica de predicciones del conjunto de prueba en la Figura 5.13, se puede observar cómo la línea azul correspondiente a las predicciones muestra un menor sobreajuste o aplanamiento en comparación con el modelo *no*

stateful, lo que indica que mantener la memoria interna entre lotes es exitoso. Sin embargo, la tendencia de la predicción sigue sin ser coherente con los valores reales.

5.3.5. Consecuencias del modelado *baseline*

En definitiva, los tres modelos *baseline* experimentados inicialmente, el modelo de regresión con *WEKA* y los dos realizados con red neuronal, *LSTM no stateful* y *LSTM stateful*, no son suficientes para predecir con precisión los valores futuros de la serie temporal de etiquetas de la turbina eólica *WT2*. Uno de los mayores problemas que presentan los tres modelos viene de trabajar con una secuencia de datos escalares de una dimensión, los cuales han sido generados con la arquitectura *U-Net* descrita en el apartado [5.1 Sistema de autoetiquetado. Arquitectura *U-Net*.](#) Dado que los modelos descritos hasta ahora son insuficientes, se exploran modelos más complejos que trabajen con una estructura de datos más rica, como las señales temporales convertidas a imágenes. Estos nuevos modelos, descritos en los siguientes apartados [5.4](#) y [5.5](#), utilizan como punto de partida el parámetro *stateful* en las unidades *LSTM* con idea de mantener memoria de situaciones previas.

5.4. Modelado con la arquitectura *U-Net + LSTM*

Esta arquitectura se entrena para predecir la etiqueta asociada al comportamiento de la turbina eólica *WT2* en un plazo de 24 horas. La longitud de la secuencia de imágenes con la que se entrena el modelo es de 65 imágenes, al igual que la empleada en los modelos *baseline*.

5.4. MODELADO CON LA ARQUITECTURA *U-NET + LSTM*

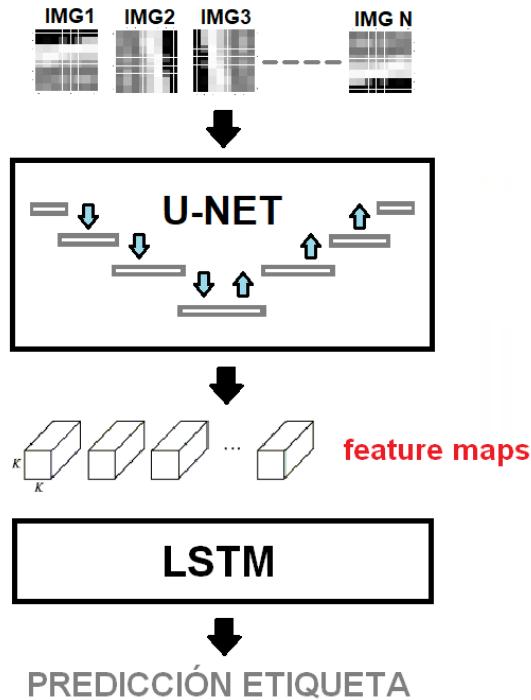


Figura 5.14: Esquema de la arquitectura *U-Net + LSTM* propuesta.

La arquitectura propuesta en este apartado consta de dos partes diferenciadas:

- la primera de ellas es la *U-Net* presentada en el trabajo de [Cambero Rojas \(2023\)](#), en la que, según se ha descrito en el apartado [5.2 Sistema de autoetiquetado. Arquitectura *U-Net*](#), se propone un modelo para el diagnóstico de fallos en turbinas eólicas mediante una red convolucional con forma de U;
- la segunda parte de la red está constituida por Redes Neuronales Recurrentes (*RNN*) (véase el [Apéndice A](#) para más detalles sobre las *RNN*), que aportan un mecanismo de atención a la serie temporal de datos, incrementando la capacidad del modelo para aprender patrones temporales a más largo plazo. Esta parte se basa en la investigación de [Orozco y cols. \(2021\)](#), donde emplean un mecanismo de atención mediante *LSTM* después de la fase convolucional que captura patrones visuales de las imágenes.

La Figura 5.14 esquematiza la arquitectura completa propuesta, que incluye la parte convolucional de la *U-Net* y la parte recurrente con las capas *LSTM*. En este punto, es

5.4. MODELADO CON LA ARQUITECTURA *U-NET + LSTM*

importante dejar patente que la **innovación** presentada en este trabajo radica en la **unificación de ambas estrategias, *U-Net* y *LSTM*.**

En primer lugar, la arquitectura *U-Net* se encarga de realizar una segmentación semántica de las señales, convertidas en imágenes, para detectar clases anómalas. Su función es producir una secuencia de mapas de características (*feature maps* en el esquema de la Figura 5.14). Estos mapas de características se corresponden a cada imagen procesada de la secuencia de entrada. El resultado se pasa luego a una fase de capas *LSTM* que aplican un mecanismo de atención sobre la secuencia de mapas de características, como si de un vídeo se tratase, con el objetivo de aprender a detectar patrones temporales sobre los patrones espaciales aprendidos de las imágenes (véase el Anexo B.2 Estructura por capas del modelo *U-Net + LSTM* propuesto para visualizar la estructura completa de la arquitectura).

El proceso de ajustar los parámetros de la *U-Net* a medida que se procesan todas las secuencias de imágenes requiere de un costo computacional muy alto, lo cual es inviable debido a las dimensiones del trabajo. La solución que se plantea es una técnica común en *DL* denominada *transfer learning*, transferir el aprendizaje. Esta técnica consiste en reutilizar modelos previamente entrenados con un conjunto de datos y transferir las características aprendidas a un nuevo modelo.

5.4. MODELADO CON LA ARQUITECTURA *U-NET + LSTM*

```

Capas transferidas de la arquitectura UNET
=====
Layer (type)
=====
input_1
EncodeConv1_1
dropout
EncodeConv1_2
EncodeAveragePool1
BridgeConv_1
dropout_1
BridgeConv_2
EncodeAveragePool2
BridgeConv_3
dropout_2
BridgeConv_4
EncodeAveragePool3
BridgeConv_5
dropout_3
BridgeConv_6
DecodeUp3
Concat3
DecodeConv3_1
dropout_4
DecodeConv3_2
DecodeUp2
Concat2
DecodeConv2_1
dropout_5
DecodeConv2_2
DecodeUp1
Concat1
DecodeConv1_1
dropout_6
DecodeConv1_2
=====

Total params: 10698369 (40.81 MB)
Trainable params: 2140865 (8.17 MB)
Non-trainable params: 8557504 (32.64 MB)
=====
```

Figura 5.15: Conjunto de capas transferidas desde el modelo *U-Net* previo de Cambero Rojas (2023) a la arquitectura *U-Net + LSTM* propuesta.

Una de las ventajas que ofrecen las redes neuronales convolucionales es que las primeras capas aprenden abstracciones más generales de las imágenes de entrada (detección de bordes, contornos, etc.) y, por tanto, se pueden volver a usar en un modelo diferente. En este trabajo, se aplica *transfer learning* desde todas las capas anteriores a la capa de aplanamiento de la arquitectura *U-Net*.

Dentro de las técnicas de *transfer learning*, existen dos formas de reutilizar

5.4. MODELADO CON LA ARQUITECTURA U-NET + LSTM

un modelo: extracción de características, o *feature extraction*¹⁰, y *fine-tuning*¹¹. El desarrollo realizado aplica la extracción de características y se establecen los pesos de esas capas como no entrenables. En la Figura 5.15 se visualizan las capas que se han transferido a la arquitectura propuesta en este trabajo. El número de parámetros no entrenables de la arquitectura propuesta es de 8.5 millones, mientras que los 2.1 millones de parámetros entrenables corresponden a la segunda fase *LSTM*.

La segunda fase de la red *LSTM* consta de tres capas *LSTM* con 32, 32 y 64 neuronas respectivamente. En ellas, se configura el parámetro *stateful* a *True*. Entre estas capas, se utiliza una capa de normalización llamada *BatchNormalization* en el entorno de desarrollo *Keras*. Esta capa es útil para prevenir el fenómeno conocido como *"gradiente explosivo"*, fenómeno que ocurre cuando los datos de entrada de una capa toman valores grandes: durante el algoritmo de retropropagación, los gradientes grandes se propagan hacia atrás, causando un crecimiento exponencial en los gradientes de las primeras capas, lo que dificulta la convergencia del modelo.

La salida de las capas *LSTM* se envía a un módulo de capas densamente conectadas *Dense* de *Keras* y termina con una última capa densa con una neurona para producir la predicción de la etiqueta en las próximas 24 horas. En la Figura 5.16 se muestra la estructura del modelo asociada a la segunda fase *LSTM*.

5.4.1. Configuración de los hiperparámetros

Un hiperparámetro es un parámetro externo al modelo que no se ajusta durante el proceso de entrenamiento de la red neuronal; lo ajusta el arquitecto del modelo con el objetivo de construir un modelo eficiente que generalice bien con nuevos datos no vistos durante la etapa de entrenamiento y devuelva buenos resultados (véase el Anexo A.3.2 Hiperparámetros si desea conocer más acerca de los mismos).

¹⁰ **Feature extraction:** en el contexto de *transfer learning* se refiere a tomar un modelo pre-entrenado en un conjunto de datos grande y general, y utilizar sus capas convolucionales, o características aprendidas, para extraer representaciones de alto nivel de nuevas imágenes.

¹¹ **Fine-tuning:** implica tomar un modelo pre-entrenado y ajustar sus parámetros específicamente para una nueva tarea o conjunto de datos; esto es, implica descongelar algunas de las capas superiores del modelo pre-entrenado y volver a entrenar el modelo en el nuevo conjunto de datos.

5.4. MODELADO CON LA ARQUITECTURA U-NET + LSTM

lstm_3 (LSTM)	(1, 65, 32)	2101376	['time_distributed_55[0][0]']
batch_normalization_2 (BatchNormalization)	(1, 65, 32)	128	['lstm_3[0][0]']
lstm_4 (LSTM)	(1, 65, 32)	8320	['batch_normalization_2[0][0]']
batch_normalization_3 (BatchNormalization)	(1, 65, 32)	128	['lstm_4[0][0]']
lstm_5 (LSTM)	(1, 64)	24832	['batch_normalization_3[0][0]']
dense_2 (Dense)	(1, 64)	4160	['lstm_5[0][0]']
dropout_16 (Dropout)	(1, 64)	0	['dense_2[0][0]']
dense_3 (Dense)	(1, 32)	2080	['dropout_16[0][0]']
dropout_17 (Dropout)	(1, 32)	0	['dense_3[0][0]']
classifier (Dense)	(1, 1)	33	['dropout_17[0][0]']
<hr/>			
Total params:	10698497 (40.81 MB)		
Trainable params:	2140929 (8.17 MB)		
Non-trainable params:	8557568 (32.64 MB)		

Figura 5.16: Estructura en capas de la segunda fase *LSTM* del modelo *U-Net + LSTM* propuesto.

La selección de hiperparámetros es crucial para el éxito del entrenamiento de una red neuronal profunda. La búsqueda y selección de los valores óptimos para los diferentes hiperparámetros se considera un *arte* para el ingeniero de modelos. La selección aleatoria de los hiperparámetros puede llevar mucho tiempo en converger y proporcionar buenos resultados debido al entrenamiento de redes neuronales con una gran cantidad de parámetros entrenables para cada conjunto de hiperparámetros seleccionados, lo que supone un alto consumo de tiempo.

La relación de hiperparámetros de configuración son los siguientes:

- **Batch Size:** del mismo modo que en el modelo *baseline* diseñado en el apartado 5.3.4.2 Red neuronal *LSTM stateful* en diferido, se establece el parámetro *stateful* a *True*, con un *batch size* de 1, porque se considera el mejor enfoque para resolver este problema. Es necesario capturar un amplio contexto en la serie temporal de imágenes para predecir el futuro de manera precisa. Por ello, el tamaño del lote (*batch size*) se mantiene en 1, lo que

garantiza que cada secuencia temporal se procese de manera independiente, permitiendo así la captura de un contexto extenso en las imágenes y mejorando la capacidad predictiva del modelo. Esta configuración es especialmente relevante en aplicaciones donde la correlación temporal entre las imágenes es crítica para la precisión de las predicciones, como es el caso en el diagnóstico de anomalías en series temporales de imágenes.

- **Número de epochs:** una ventaja que ofrece el *batch size* de 1 es que en cada etapa del proceso de entrenamiento se procesa la serie temporal completa una sola vez en orden temporal. Dado que el comportamiento óptimo de la turbina eólica (etiqueta con valor 1) es el estado más frecuente, la selección del número de etapas es un factor clave en el entrenamiento del modelo. Un número alto de *epochs* puede desembocar en un sobreajuste. Por ejemplo, al comienzo de la segunda etapa, se procesa el principio de la serie temporal con unos parámetros que se han ajustado tras haber procesado toda la serie temporal en la etapa anterior (en cierto modo se usa información del futuro para ajustar de nuevo los parámetros en una posición del pasado). Un número de etapas pequeño puede dar lugar a un subajuste, o *underfitting*, donde el modelo es demasiado simple para capturar la estructura subyacente de los datos.

Por tanto, se seleccionan los números de *epochs* 1, 2 y 4 para comparar los resultados. Una práctica recomendable en la selección de un hiperparámetro es evaluar cómo se comporta el modelo con un valor extremo. También se opta por entrenar el modelo con 10 *epochs*.

- **Optimizador:** en el trabajo de Cambero Rojas (2023), se emplea el optimizador *Madam* para la arquitectura *U-Net*. En el modelo propuesto, la fase convolucional no es entrenable, por lo que se elige por defecto el optimizador *RMSProp*. Este optimizador se basa en el algoritmo de descenso del gradiente estocástico (*SGD*) y adapta la tasa de aprendizaje para cada parámetro en función del historial de gradientes pasados para cada parámetro. *RMSProp* es

conocido por acelerar la convergencia del modelo cuando se trabaja con grandes volúmenes de datos.

- **Función de pérdida**, o *loss*: se emplea el error cuadrático medio *MSE* como función de pérdida. La función da más peso a las grandes discrepancias entre las predicciones y los valores reales, lo cual es adecuado en muchos contextos de problemas. Por ejemplo, si la etiqueta real es 5 y la predicha es 1, la diferencia es significativa, lo que indica que el modelo necesita ajustar sus pesos más significativamente para mejorar la precisión de la predicción.
- **Dropout**: se establece un valor de *dropout* recurrente a 0,25 para garantizar la eficacia del modelo durante de entrenamiento y mitigar el sobreajuste causado por el comportamiento mayormente óptimo de la turbina eólica (etiquetas a 1).

5.5. Modelado con la arquitectura *ConvLSTM*

A la vista de los resultados devueltos por el modelo *U-Net + LSTM* del apartado 5.4 anterior, y que serán expuestos en el apartado ([6.3 Resultados de la arquitectura *U-Net + LSTM*](#)) del [Capítulo 6. Resultados](#), buscando su mejora surge la propuesta de un modelo de red neuronal constituido por un tipo de capas que combinan las operaciones de convolución de las *Conv2D* y la memoria interna de las *LSTM*, ambas capas expuestas en modelos anteriores, denominado *ConvLSTM*. La idea es investigar si esta nueva aproximación del modelo proporciona una mayor precisión en la predicción de anomalías en comparación con el modelo *U-Net + LSTM*. La combinación de capas *ConvLSTM* integran las operaciones convolucionales sustituyendo las operaciones de matrices de las unidades de memoria interna de las capas *LSTM* (véase el apartado [A.4.3 RN recurrentes](#) para conocer en detalle el funcionamiento interno de las redes neuronales recurrentes).

El objetivo es investigar si construir la arquitectura *ConvLSTM* a partir del modelo *LSTM stateful* en diferido (apartado [5.3.4.2](#)) presenta una mejora en la predicción de anomalías.

5.5. MODELADO CON LA ARQUITECTURA CONVLSTM

Model: "ConvLSTM_simple_Regression"		
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[1, 65, 16, 16, 3]	0
conv_lstm2d (ConvLSTM2D)	(1, 65, 16, 16, 32)	13568
batch_normalization (Batch Normalization)	(1, 65, 16, 16, 32)	128
conv_lstm2d_1 (ConvLSTM2D)	(1, 65, 16, 16, 64)	73984
batch_normalization_1 (BatchNormalization)	(1, 65, 16, 16, 64)	256
conv_lstm2d_2 (ConvLSTM2D)	(1, 16, 16, 64)	98560
batch_normalization_2 (BatchNormalization)	(1, 16, 16, 64)	256
flatten (Flatten)	(1, 16384)	0
dense (Dense)	(1, 64)	1048640
dropout (Dropout)	(1, 64)	0
dense_1 (Dense)	(1, 1)	65

Total params: 1235457 (4.71 MB)
 Trainable params: 1235137 (4.71 MB)
 Non-trainable params: 320 (1.25 KB)

Figura 5.17: Estructura en capas del modelo *ConvLSTM* propuesto

En la Figura 5.17 se presenta la estructura por capas del modelo diseñado *ConvLSTM*, en el que se reemplazan las capas *LSTM* del modelo base por capas *ConvLSTM*. Es importante destacar que el formato de los datos de entrada de este nuevo modelo consiste en un conjunto de secuencias de imágenes, de manera similar a la arquitectura *UNET + LSTM* descrita en el apartado 5.4 anterior.

En cuanto a la configuración de los hiperparámetros, se mantiene el enfoque aplicado en los modelos anteriores: con el parámetro *stateful* activado; el optimizador *RMSprop* seleccionado, de manera similar al del modelo *U-Net + LSTM*; un tamaño de *kernel* de 3×3 y una única etapa de entrenamiento, con el propósito de analizar el estado de convergencia al finalizar el proceso de entrenamiento con pocas *epochs*; y, por último, al igual que en las implementaciones anteriores, se opta por utilizar el error cuadrático medio, *MSE*, como función de análisis de la pérdida.

5.5. MODELADO CON LA ARQUITECTURA CONVLSTM

Con las dos arquitecturas propuestas ya diseñadas, se procede a analizar los resultados de las pruebas de validez efectuadas en el capítulo que sigue.

Capítulo 6

Resultados

En el presente capítulo se describen las pruebas completadas sobre las dos arquitecturas propuestas en el [Capítulo 5. Implementación y desarrollo de la arquitectura *U-Net + LSTM*](#), dada la insuficiente complejidad como para obtener una respuesta satisfactoria de los modelos más elementales diseñados en su apartado [5.3 Modelado *baseline* inicial](#).

El objetivo de este trabajo es explorar múltiples enfoques para abordar el problema de la predicción de anomalías en turbinas eólicas, utilizando redes neuronales sobre la serie temporal de datos provenientes de las señales de los sensores y convertida en imágenes. La implementación secuencial de distintos enfoques conduce a una serie de conclusiones que podrían ser fundamentales para una futura redefinición del problema.

6.1. Procedimiento de la fase de prueba

El procedimiento empleado para la fase de prueba de las arquitecturas *U-Net + LSTM* y *ConvLSTM* difiere del procedimiento llevado a cabo en la sección [5.3 Modelado *baseline* inicial](#).

En el estudio de modelos *baseline* se utiliza el proceso de prueba de validación hacia delante. Este método consiste en reentrenar el modelo con la secuencia de datos del conjunto de pruebas que se utilizó en la iteración previa para predecir la etiqueta

6.2. DEFINICIÓN DEL RESULTADO DESEADO

futura. De esta manera, se mantiene consistente el estado interno de las unidades *LSTM*. Además, en estos modelos se predice una secuencia de 10 valores futuros, añadiendo la última etiqueta pronosticada al final de la secuencia de entrada y eliminando la primera etiqueta de forma iterativa, hasta predecir las siguientes 10 etiquetas futuras.

En contraste, para las arquitecturas *U-Net + LSTM* y *ConvLSTM* se requiere un enfoque diferente. La compatibilidad entre la estructura de datos de la predicción y los datos de la secuencia es crucial. Mientras que en los modelos *baseline* la predicción numérica se puede integrar fácilmente en la secuencia de entrada, en estas otras arquitecturas esto no es posible debido a la naturaleza de las secuencias de imágenes. La predicción numérica de la etiqueta futura no se puede incluir al final de la secuencia de imágenes de entrada; por lo tanto, la evaluación de ambos modelos se realiza mediante la predicción de un único valor futuro (24 horas), dada la secuencia de imágenes previa del conjunto de prueba.

Las métricas empleadas para medir el rendimiento de ambos modelos son el error absoluto medio, *MAE*, y el error cuadrático medio, *MSE*, empleados como función de pérdida.

6.2. Definición del resultado deseado

Las gráficas de resultados que se adjuntan en los siguientes apartados persiguen el objetivo de este trabajo de identificar una predicción que anticipa el comportamiento de la serie temporal real.

Por defecto, los modelos tenderán a presentar un sobreajuste sobre el comportamiento óptimo de la turbina eólica y nunca alcanzarán a predecir con exactitud los valores de las etiquetas, debido a la gran disparidad en la distribución de las mismas en el conjunto de datos. Es común observar un comportamiento óptimo de las turbinas eólicas con valores de etiquetas igual a 1. Por ello, la fase de prueba se centra en analizar las tendencias presentes en las predicciones sobre el conjunto de prueba.

En todas las gráficas utilizadas se empleará el trazado en azul de las líneas para las

6.3. RESULTADOS DE LA ARQUITECTURA *U-NET + LSTM* PROPUESTA

predicciones, y el trazado en rojo para los valores utilizados en el entrenamiento del modelo.

Con ello, resultado se considera óptimo cuando las predicciones (trazado azul) se produzcan antes de los valores de estado de entrenamiento; es decir, se presente un desplazamiento a la izquierda sobre las abscisas de las gráficas del trazo azul respecto de los resultados recogidos por el trazo rojo. En esta situación ideal, el modelo debería ser capaz de predecir correctamente las 24 horas siguientes.

6.3. Resultados de la arquitectura *U-Net + LSTM* propuesta

La combinación del parámetro *stateful* a *True* y un *batch size* de 1 provoca que en cada etapa del proceso de entrenamiento se procese la serie temporal de imágenes completa en orden temporal. La elección del número de etapas es fundamental: un número grande de etapas puede conducir al sobreajuste porque el comportamiento más frecuente de la turbina eólica es el óptimo (valor de etiqueta 1); un número pequeño de etapas puede dar lugar a un subajuste y el modelo puede no capturar bien las dependencias a largo plazo.

A continuación, se describen los resultados recogidos del modelo entrenado en 1, 2, 4 y 10 etapas o *epochs*.

6.3. RESULTADOS DE LA ARQUITECTURA U-NET + LSTM PROPUESTA

Comparación de predicciones promediadas y secuencia real

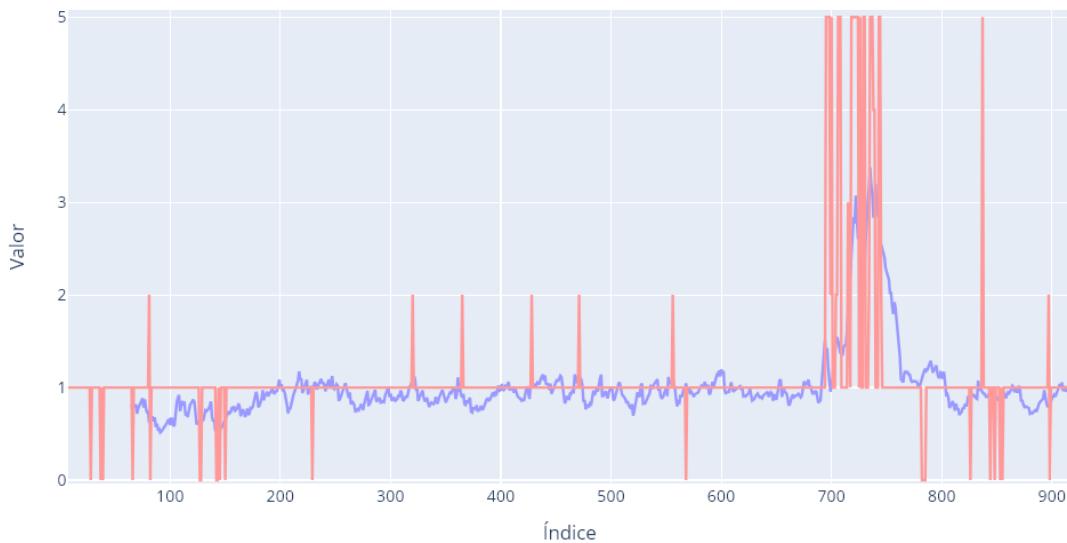


Figura 6.1: Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 1 epoch. *Valor*: identificador de etiqueta; *Índice*: secuencia temporal de las etiquetas.

1 Epoch

La fase de entrenamiento del modelo con 1 etapa devuelve los siguientes valores: $MSE = 0,4995$ y $MAE = 0,45$.

La Figura 6.1 muestra la comparación de las predicciones de etiquetas que realiza el modelo (trazado azul) y los valores de entrenamiento del conjunto de datos de prueba procedente del año 2016 (trazado rojo). En las ordenadas se indica como *valor* las etiquetas de estado; y en las abscisas el índice de la secuenciación de los estados a lo largo del tiempo. Atendiendo a la Figura 6.1, se observa que la predicción del modelo es inestable, ya que presenta picos crecientes y decrecientes en las predicciones (en azul). Es más, el modelo no es capaz de predecir con antelación las etiquetas con valor 5, estado de *Parada*; siendo este el caso más fácil de predecir, debido a la diferencia de valor que existe entre las etiquetas 5, del estado de *Parada*, con las etiquetas 1 de *Rendimiento Óptimo*, las más frecuentes de todo el procesamiento al ser el estado normal de los aerogeneradores. Esta situación puede distinguirse con suma claridad entre los índices 700 – 750 de la gráfica.

6.3. RESULTADOS DE LA ARQUITECTURA U-NET + LSTM PROPUESTA

Comparación de predicciones promediadas y secuencia real

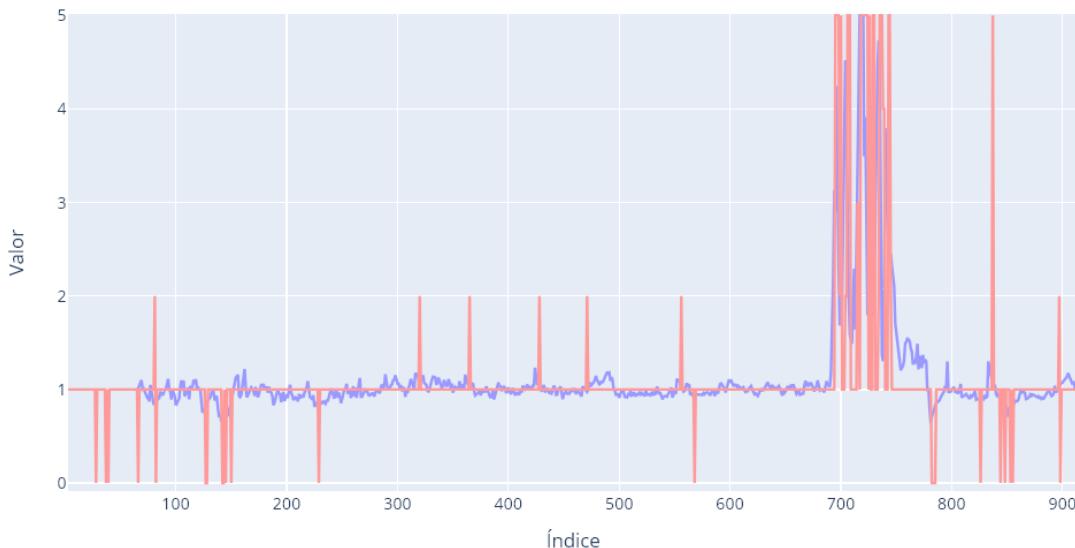


Figura 6.2: Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 2 epochs. *Valor*: identificador de etiqueta; *Índice*: secuencia temporal de las etiquetas.

2 Epochs

Tras dos etapas del proceso de entrenamiento, el modelo devuelve los siguientes valores: $MSE = 0,35$ y $MAE = 0,33$.

Se evidencia un cambio sustancial en las métricas respecto a las de 1 epoch: se reducen debido a que los parámetros del modelo se reajustan nuevamente con el conjunto completo de entrenamiento.

Con la Figura 6.2 volvemos a la representación comparativa entre las predicciones de etiquetas realizadas por el modelo (trazado azul) y los valores de las etiquetas en el conjunto de datos de prueba (trazado rojo). Ahora se aprecia una mayor estabilidad en las predicciones respecto al modelo entrenado con una sola epoch. Además, en algunas anomalías identificadas con la etiqueta 2, *Bajo Rendimiento*, se observa una tendencia creciente previa en las predicciones (en azul).

En relación al rango de etiquetas entre los índices 700 – 750, el modelo predice correctamente el conjunto de anomalías etiquetadas con el valor 5, *Parada*, con una precisión superior a los resultados del modelo de una sola epoch, como se puede

6.3. RESULTADOS DE LA ARQUITECTURA U-NET + LSTM PROPUESTA

Comparación de predicciones promediadas y secuencia real

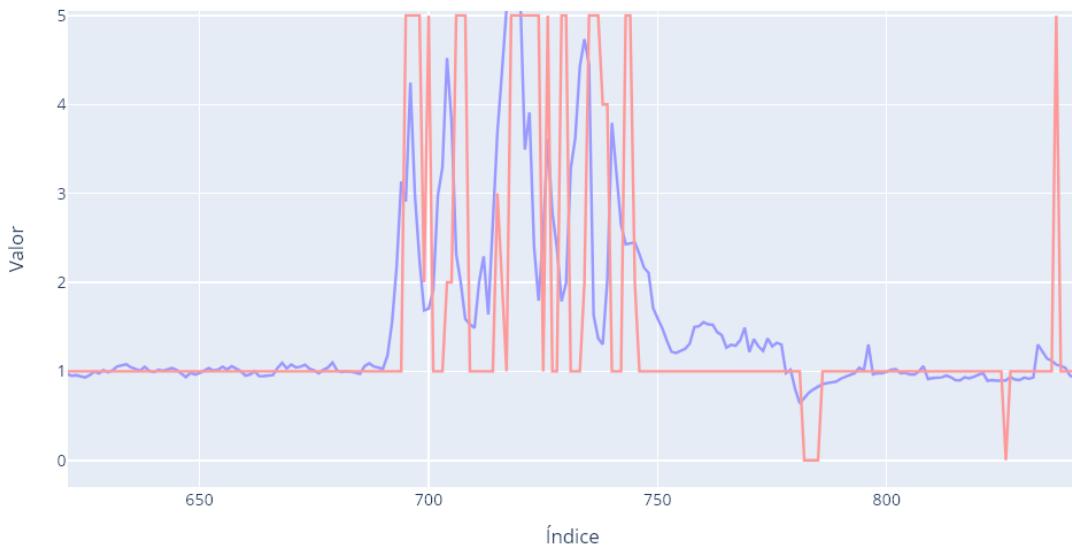


Figura 6.3: Rango ampliado (600 – 900) de predicciones de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 2 epochs. *Valor: identificador de etiqueta; Índice: secuencia temporal de las etiquetas.*

contrastar con mayor apreciación en la Figura 6.3 ampliando los trazados recogidos por la Figura 6.2 entre los rangos de índice 600 – 850.

Este modelo exhibe un comportamiento más próximo al deseado en comparación con el modelo de una sola epoch; no obstante, aún muestra ciertas tendencias que representan falsos positivos para las anomalías.

4 Epochs

El proceso de entrenamiento con 4 etapas arroja unas métricas: $MSE = 0,3096$ y $MAE = 0,2684$.

El modelo muestra métricas más ajustadas en comparación con su predecesor de 2 epochs. Es crucial analizar las predicciones realizadas en el conjunto de prueba para evaluar si las nuevas métricas reflejan un comportamiento de predicción más preciso en relación con la predicción de las anomalías (etiquetas 0, 2, 3, 4 y 5).

La Figura 6.4 refleja las predicciones realizadas sobre el conjunto de prueba del año 2016 (trazado en azul) respecto la secuencia real de etiquetas (trazado en rojo).

6.3. RESULTADOS DE LA ARQUITECTURA U-NET + LSTM PROPUESTA

Comparación de predicciones promediadas y secuencia real

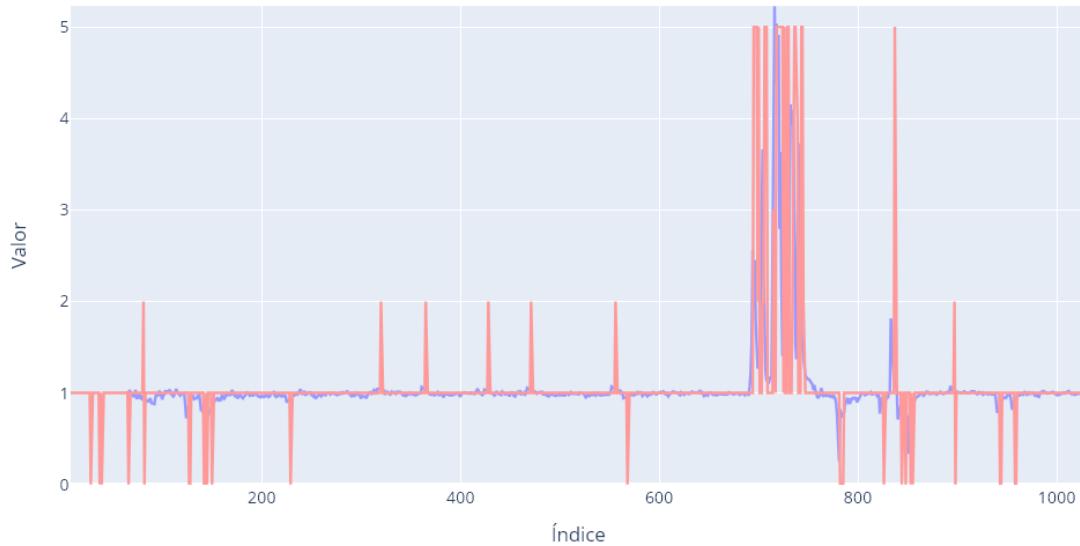


Figura 6.4: Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 4 epochs. *Valor: identificador de etiqueta; Índice: secuencia temporal de las etiquetas.*

Con 4 etapas de entrenamiento, las predicciones sufren un sobreajuste respecto las etiquetas con valor 1 de comportamiento óptimo. No se encuentran evidencias de que el modelo sea capaz de predecir las anomalías con total exactitud, pero se puede observar un comportamiento anómalo previo a la mayoría de las anomalías (2 y 0, *Bajo Rendimiento* y *Excesivo Rendimiento*, respectivamente) presentes en la Figura 6.5, que nos amplía la visión centrándose en las muestras entre los índices de secuencia 75 – 600 y valores de etiquetaje 0 – 2. En las primeras 150 etiquetas del conjunto de prueba, las predicciones trazadas en azul reflejan una caída previa a las etiquetas con valor 0, *Excesivo Rendimiento*, además de múltiples pequeños picos anteriores a las etiquetas 2, *Bajo Rendimiento*.

En la comparación del conjunto de anomalías de tipo 5, *Parada*, entre la Figura 6.6, ampliación con 4 epochs, y la Figura 6.3, ampliación con 2 epochs, se observa que el modelo predice las anomalías de tipo 5 de manera bastante similar. Sin embargo, con 4 epochs es capaz de predecir con una mayor precisión las anomalías posteriores de tipo 0, *Excesivo Rendimiento*.

6.3. RESULTADOS DE LA ARQUITECTURA U-NET + LSTM PROPUESTA

Comparación de predicciones promediadas y secuencia real

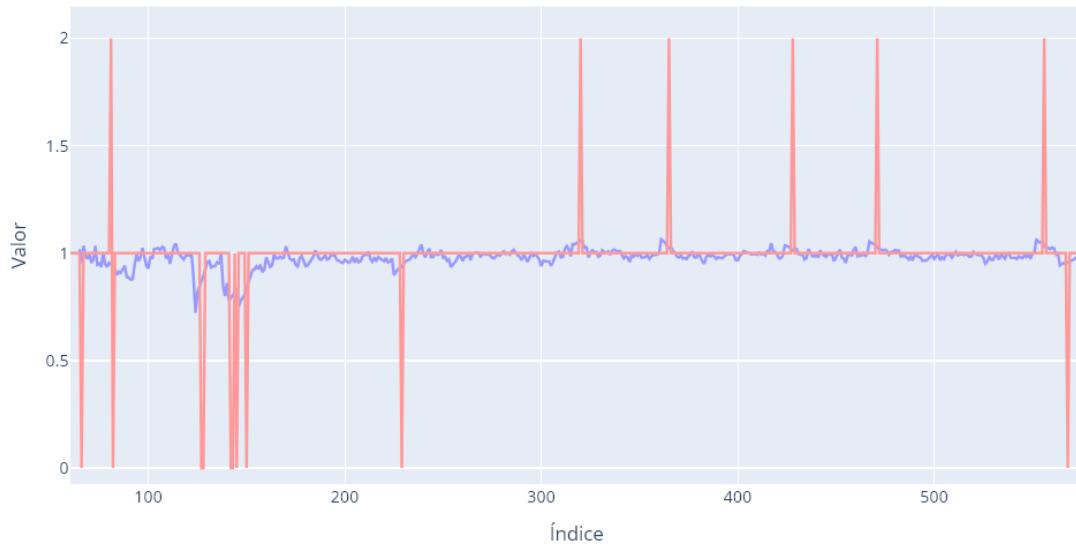


Figura 6.5: Rango ampliado (75 – 600, etiquetas 0 – 2) de predicciones de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 4 epochs. *Valor: identificador de etiqueta; Índice: secuencia temporal de las etiquetas.*

Comparación de predicciones promediadas y secuencia real

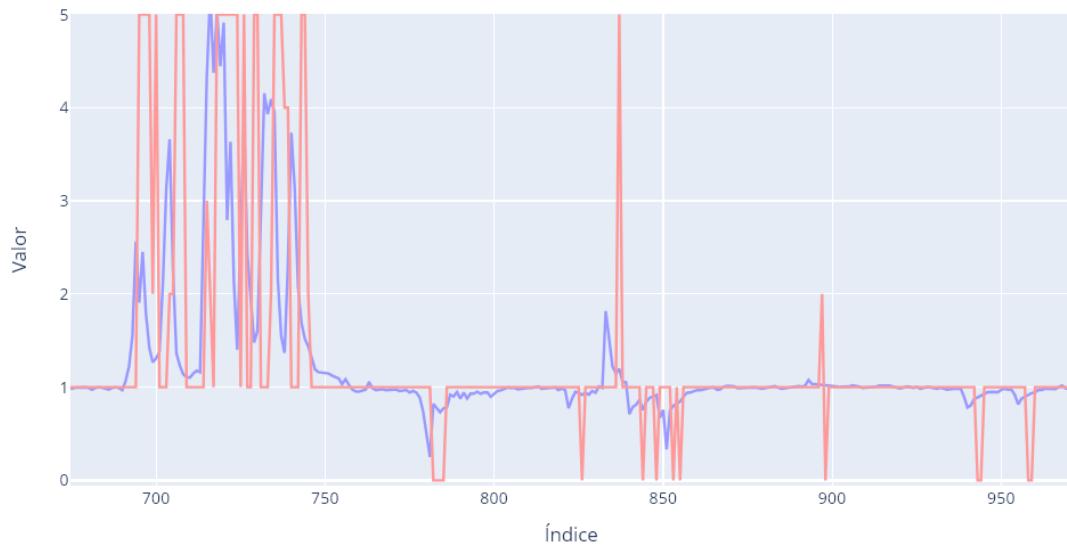


Figura 6.6: Rango ampliado (675 – 975) de predicciones de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 4 epochs. *Valor: identificador de etiqueta; Índice: secuencia temporal de las etiquetas.*

Este modelo presenta una mejora notable en las predicciones de las anomalías 5,

6.3. RESULTADOS DE LA ARQUITECTURA U-NET + LSTM PROPUESTA

Comparación de predicciones promediadas y secuencia real

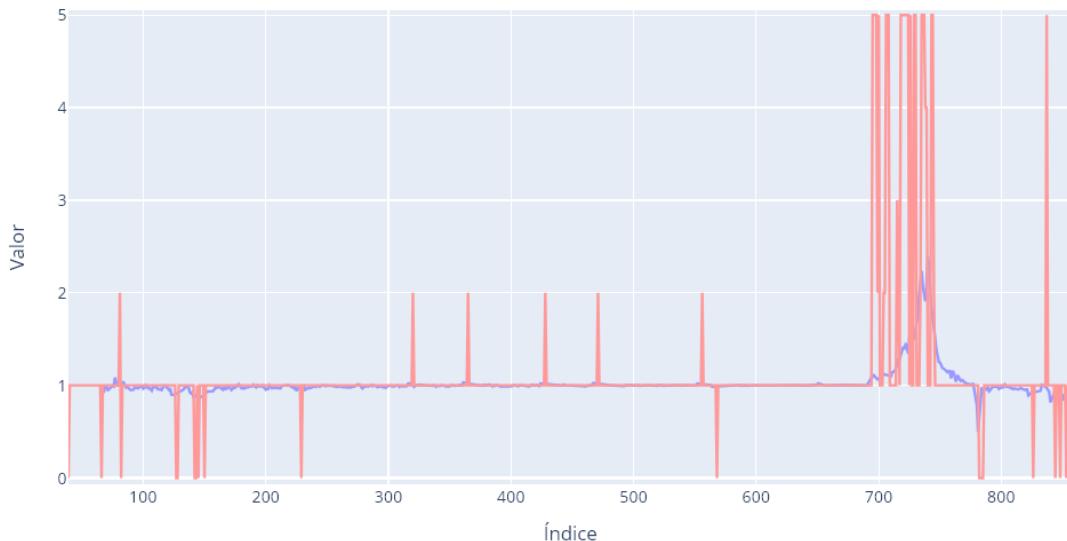


Figura 6.7: Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *U-Net + LSTM* y 10 epochs. *Valor: identificador de etiqueta; Índice: secuencia temporal de las etiquetas.*

Parada y 0, *Excesivo Rendimiento*. En cuanto a las de tipo 2, *Bajo Rendimiento*, el modelo genera en las predicciones un pequeño pico previo a dichas anomalías, aunque con una precisión ligeramente menor.

10 Epochs

El proceso de entrenamiento con 10 etapas devuelve unas métricas de: $MSE = 0,28$ y $MAE = 0,20$.

Del mismo modo que en el modelo de 4 epochs, a continuación se analizan las predicciones realizadas en el conjunto de prueba para concluir si un valor alto de epochs afecta al rendimiento del modelo en la predicción de anomalías.

En la Figura 6.7, las predicciones con las etiquetas 0, *Excesivo Rendimiento*, y 5, *Parada*, son menos precisas respecto al modelo de 4 epochs. Las predicciones, trazadas en azul, se ajustan considerablemente a las etiquetas con valor 1, *Óptimo Rendimiento*, lo cual es una señal negativa si el objetivo ideal es predecir comportamientos anómalos. La Figura 6.8 nos amplía la visualización del comportamiento predictivo entre las

6.3. RESULTADOS DE LA ARQUITECTURA *U-NET + LSTM* PROPUESTA

Comparación de predicciones promediadas y secuencia real

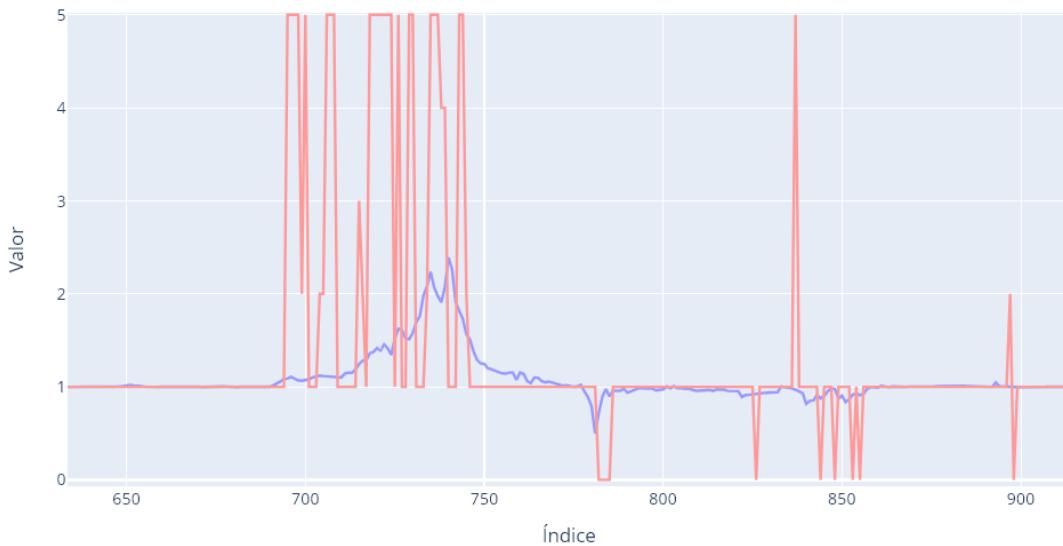


Figura 6.8: Rango ampliado de predicciones 700 - 950 sobre las etiquetas con valor 5 y 0.

predicciones 650 – 925, donde podemos observar un deterioro en las predicciones. trazadas en azul, del conjunto de anomalías con valor 5, *Parada*, respecto a los modelos *U-Net + LSTM* previamente experimentados.

La evolución de los resultados recopilados de la arquitectura *U-Net + LSTM* indica que el modelo entrenado con 4 *epochs* predice con mayor precisión las anomalías respecto al resto de modelos, centrándose el estudio en las anomalías de tipo 0, 5 y 2, *Excesivo Rendimiento*, *Parada* y *Bajo Rendimiento*, respectivamente. Sin embargo, la predicción queda lejos del resultado esperado. La elección de un número mayor de épocas como hiperparámetro terminaría resultando un modelo incapaz de predecir comportamientos anómalos, como se puede observar en los resultados obtenidos con el modelo de 10 *epochs*.

‘El carácter experimental de este trabajo motiva a seguir explorando la creación de un nuevo modelo, con el propósito de alcanzar una mayor precisión en la predicción de anomalías, superando así los resultados prometedores obtenidos por el modelo anteriormente presentado.’

6.4. RESULTADOS DE LA ARQUITECTURA CONVLSTM PROPUESTA

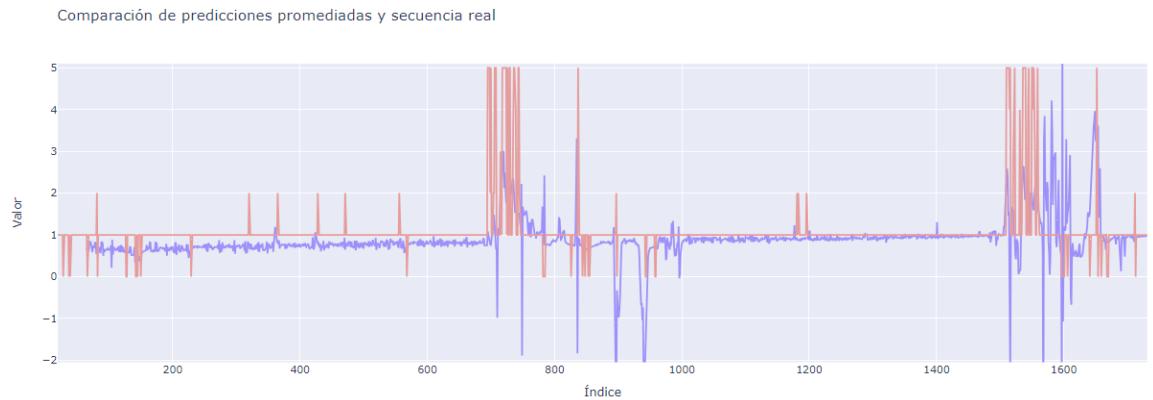


Figura 6.9: Predicción de etiquetas en 24 horas en el conjunto de prueba del año 2016. Modelo *ConvLSTM*. *Valor*: identificador de etiqueta; *Índice*: secuencia temporal de las etiquetas.

6.4. Resultados de la arquitectura *ConvLSTM* propuesta

A partir de los resultados imprecisos pero prometedores obtenidos con el modelo *U-Net + LSTM* de 4 etapas, se decide continuar la línea de investigación con la implementación y evaluación de un nuevo modelo utilizando las capas *ConvLSTM*, tal como se describe en la sección [5.5 Arquitectura *ConvLSTM* propuesta](#).

En la Figura 6.9 se presenta la gráfica que visualiza la predicción (trazada en azul) realizada por el modelo *ConvLSTM*, comparada con la secuencia real de etiquetas del conjunto de prueba (trazada en rojo), en un rango extenso de 2,000 predicciones del año 2016. Se observa que el modelo no predice adecuadamente el conjunto de anomalías de tipo 5, *Parada*, acumuladas entre los índices de secuenciación 700 – 800, así como entre los índices 1500 – 1600. Además, el modelo presenta un comportamiento irregular en determinados índices, específicamente en el rango 800 – 1,000. En definitiva, el modelo *ConvLSTM* muestra un resultado lejos de lo esperado en comparación con la arquitectura *U-Net + LSTM*.

6.4. RESULTADOS DE LA ARQUITECTURA CONVLSTM PROPUESTA

Capítulo 7

Conclusiones y trabajos futuros

En este capítulo se exponen tanto las conclusiones personales derivadas de la experiencia y reflexión del autor durante el desarrollo del proyecto de investigación experimental, como las conclusiones específicas del proyecto. Las conclusiones personales abarcan las percepciones individuales sobre el proceso de investigación, los aprendizajes obtenidos y las reflexiones personales sobre el tema estudiado.

Termina el capítulo con las propuestas de trabajos futuros que se estiman convenientes seguir abordando.

7.1. Conclusiones

Las conclusiones alcanzadas durante esta línea de investigación se dividen en dos secciones: la primera, las conclusiones relacionadas con el proyecto; la segunda, las conclusiones relativas a la visión personal del proceso seguido.

7.1.1. Conclusiones relacionadas con el proyecto

- El estudio inicial de modelos básicos demuestra que dichos modelos son incapaces de predecir de manera satisfactoria las anomalías de las turbinas eólicas en este trabajo. Estos modelos simples, aunque útiles como punto de partida, se revelan insuficientes para abordar la complejidad de las anomalías en

7.1. CONCLUSIONES

este contexto específico. La investigación y puesta a prueba de modelos de redes neuronales más complejos como la arquitectura *U-Net + LSTM*, que trabajen con estructuras de datos más ricas en información, suponen una línea a explorar en futuros trabajos.

- La arquitectura *ConvLSTM*, que presenta una estructura de capas similar a las *LSTM* en diferido, devuelve unos resultados lejos de los esperados. En cambio, el modelo *U-Net + LSTM* descrito en el presente proyecto puede servir como guía para el desarrollo de un sistema de predicción deseado en el campo de las turbinas eólicas. Dado los resultados proporcionados por este modelo más complejo, se sugiere que las investigaciones futuras sigan esta misma dirección de trabajo.
- Si bien se lograron completar los objetivos establecidos en este proyecto de investigación experimental, descritos en el apartado 1.2 **Objetivos**, y se extrajeron conclusiones beneficiosas para los futuros trabajos, es importante destacar que el desarrollo del sistema de predicción de anomalías en turbinas eólicas no alcanzó la etapa deseada. La predicción de anomalías en aerogeneradores es un desafío técnico y científico considerable, que requiere un análisis exhaustivo de múltiples variables y una comprensión profunda de los procesos involucrados.

7.1.2. Conclusiones personales

La realización del proyecto me ha proporcionado un amplio conocimiento en el campo de la inteligencia artificial. La lectura y estudio de las dos principales fuentes de formación, [Torres \(2020\)](#) y [Chollet \(2021\)](#), han despertado una curiosidad infinita y un deseo de adentrarme más a fondo en las redes neuronales. La comprensión del funcionamiento interno de las redes neuronales, incluyendo sus aspectos matemáticos, ha sido un factor crucial en la consecución de este trabajo.

Por otro lado, aunque los resultados no han cumplido completamente con las

7.2. TRABAJOS FUTUROS

expectativas y han podido ser algo frustrantes en momentos decisivos, enfrentarme al desafío de desarrollar un sistema de predicción de anomalías, objetivo de este trabajo y sin precedentes en publicaciones académicas hasta la fecha, mediante la metodología seguida en los trabajos fin de grado anteriores, englobados en el proyecto *Anemoi*, ha sido muy enriquecedor. Resolver los problemas planteados durante todo el proceso ha contribuido a mi desarrollo personal y profesional.

Este proceso formativo, junto con el aprendizaje y la resolución de errores, así como la inmersión en los algoritmos de aprendizaje profundo, representa un salto cualitativo en mi última etapa universitaria y será de gran utilidad en mi futuro laboral.

7.2. Trabajos futuros

En este apartado se presentan las posibles líneas futuras que se derivan de la investigación llevada a cabo en el proyecto.

Una primera vía de trabajo futuro es emplear las capas *ConvLSTM*, las cuales integran operaciones de convolución en las transformaciones de los estados internos de una capa *LSTM*. Estas capas están diseñadas para realizar predicciones espacio-temporales, lo que las hace ideales para aplicaciones sobre conjuntos de imágenes, donde cada una representa una porción de serie temporal. En el ámbito de la predicción de anomalías en turbinas eólicas, no se encuentran publicaciones académicas actuales que hagan uso de este tipo de capas. Además, dada la eficacia del sistema de detección de anomalías de la *U-Net* desarrollada en los trabajos fin de grado previos de [Delgado Muñoz \(2022\)](#) y de [Cambero Rojas \(2023\)](#), la inclusión de las capas *ConvLSTM* en esta arquitectura en forma de U podría ser una dirección prometedora.

Una segunda vía de continuación sería abordar modelos similares a los proporcionados en este proyecto mediante un enfoque de clasificación, donde cada secuencia de imágenes esté asociada a una clase futura a predecir. Estos modelos devolverían, mediante la función de activación *softmax* (característica de problemas de clasificación), la distribución de probabilidad de pertenencia a cada clase. Al analizar

7.2. TRABAJOS FUTUROS

manualmente la evolución de las probabilidades en una secuencia de salida, o al transmitir estas probabilidades a un segundo modelo, podría detectarse tendencias anómalas en la evolución de las probabilidades, lo que podría conducir a una arquitectura innovadora no vista hasta el momento.

Anexos

Apéndice A

Conceptos fundamentales del *Deep Learning*

En este apéndice se abordan conceptos fundamentales del *Deep Learning (DL)*. En primer lugar, se introduce la disciplina del *DL* y su origen en relación con la inteligencia artificial y el *Machine Learning (ML)*. Posteriormente, se explica qué es una neurona, los tipos de redes neuronales (RN), su propósito y funcionamiento interno. Se detalla el análisis de las redes neuronales implicadas en el presente TFG para comprender exhaustivamente la arquitectura *U-Net* con *LSTM* del [Capítulo 5](#). **Implementación y desarrollo de la arquitectura *U-Net + LSTM*.** Como se explorará más adelante, una de las fases más cruciales del modelado en *DL* es la fase de evaluación, donde se evalúa el rendimiento de los modelos de aprendizaje profundo mediante diversas métricas. Será en este anexo donde se expondrán estas métricas detalladamente.

Dos de las fuentes que más conocimiento me han proporcionado a lo largo de todo el proceso de este proyecto son [Torres \(2020\)](#) y [Chollet \(2021\)](#). Estas referencias, seleccionadas minuciosamente, han sido pilares fundamentales en la teórica y práctica de mi trabajo. La información contenida en estas fuentes ha influido de manera directa en la calidad del anexo.

A.1. INTRODUCCIÓN A DEEP LEARNING

A.1. Introducción a *Deep Learning*

El concepto de aprendizaje profundo o *Deep Learning* (*DL*) está arraigado a otros conceptos como *Machine Learning* (aprendizaje máquina) e Inteligencia Artificial (IA).

Durante la historia del ser humano, múltiples disciplinas han contribuido al desarrollo de la IA, desde la filosofía, con Aristóteles (384 - 322 a.C) o Thomas Hobbes (1588 - 1679), este último propuso un modelo de computación numérica que seguía el razonamiento “mientras pensamos, realizamos operaciones numéricas en silencio”. Las matemáticas han contribuido al avance de la IA por medio de la lógica, con George Boole (1815 - 1864) que definió la lógica proposicional, por la computación con Alan Turing (1912 - 1954) y la probabilidad con genios como Pierre Laplace (1749 - 1827) o Thomas Bayes (1702 - 1761). Otras disciplinas como la economía, psicología, lingüística y la neurociencia han aportado conocimiento a la IA. El descubrimiento en 1929 del electroencefalograma (desarrollado por Hans Berger) y las imágenes de resonancia magnética funcional fueron claves en entender cómo el cerebro humano creaba las inteligencias, como dijo Searle (1992): “Una colección de simples células puede llegar a generar razonamiento, acción, y conciencia. Los cerebros generan las inteligencias”.

La inteligencia artificial se puede definir como el esfuerzo de automatizar tareas intelectuales normalmente hechas por humanos.

Turing (2012) fue el primero en articular una primera visión de la inteligencia artificial de manera formal. Hasta la contribución de Turing, la IA se conocía como “IA simbólica”: consistía en un grupo de programadores que manejaban una inmensa cantidad de reglas explícitas para aplicar sobre un conjunto grande de conocimiento. Sin embargo, esta metodología no era capaz de averiguar reglas explícitas complejas para resolver problemas de reconocimiento de patrones, de clasificación, etc. Turing y John McCarthy fueron los pioneros de *ML*, exponiendo un cambio de paradigma de programación para resolver problemas. En vez de programar manualmente las reglas para que actúen sobre datos de entrada y generen respuestas, *ML* busca aprender estas

A.1. INTRODUCCIÓN A DEEP LEARNING

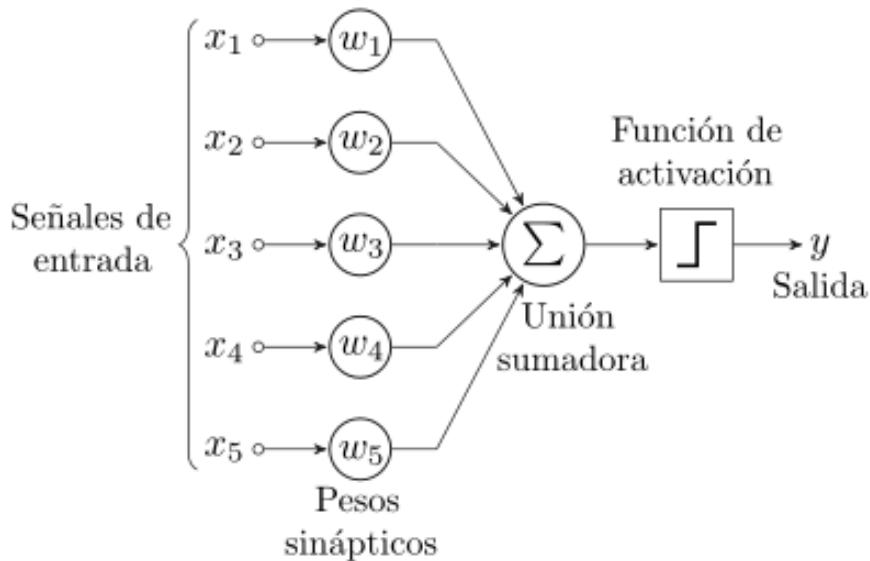


Figura A.1: Diagrama de un perceptrón con cinco señales de entrada. Fuente: [Wikipedia \(2023\)](#)

reglas complejas a partir de los datos de entrada y las salidas deseadas generadas.

Se puede definir *ML* como un subconjunto de la IA enfocado a enseñar a las máquinas para que aprendan de los datos y mejoren con la experiencia, sin instrucciones explícitas, usando algoritmos y modelos estocásticos para analizar e inferir patrones sobre los datos.

Un caso de algoritmo de *ML* es el perceptrón desarrollado por [Rosenblatt \(1958\)](#). El perceptrón es la definición formal más sencilla de una neurona. Es un algoritmo capaz de generar un criterio para seleccionar un subgrupo a partir de un grupo de componentes más grande. En la siguiente Figura A.1, se puede observar el diseño de un perceptrón con cinco señales de entrada.

- Las entradas x_i conforman el vector de entrada que será procesado por la neurona para producir una salida.
- Los pesos w_i constituyen el vector de pesos o parámetros de la neurona. Su función es ponderar los valores del vector de entrada.
- Unión sumadora: es el sumatorio del producto escalar de cada peso con su correspondiente valor de entrada.

A.1. INTRODUCCIÓN A DEEP LEARNING

- Función de activación: aplica transformaciones no lineales sobre el resultado del sumatorio de la neurona para producir una salida.

Esta es la arquitectura básica de una neurona artificial. A partir de ella, han surgido variantes de neuronas más completas que incluyen, por ejemplo, un nuevo elemento llamado sesgo (b) en la operación de suma ponderada. Este sesgo forma parte del conjunto de parámetros ajustables de la neurona.

Si un algoritmo se define con un perceptrón, este se clasifica como una RN con una sola capa con una sola neurona. En el caso de que la RN tenga múltiples capas de neuronas, se considera una RN profunda.

El *DL* puede ser definido como una técnica dentro del *ML*, en la cual la información se procesa a través de capas jerárquicas. Se trata del aprendizaje de múltiples representaciones de los datos de entrada, y la profundidad del modelo está determinada por el número de capas que lo conforman.

Para finalizar el aspecto introductorio, en la siguiente Figura A.2 se resume la taxonomía de las disciplinas de Inteligencia Artificial, *Machine Learning* y *Deep Learning*.

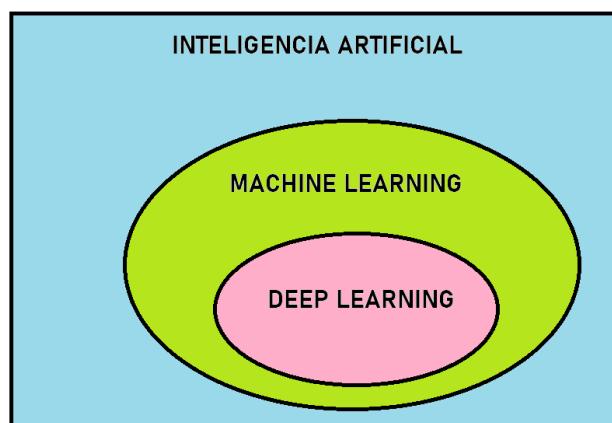


Figura A.2: Taxonomía de la IA, *DL* y *ML*.

A.2. Interpretación matemática y geométrica de una neurona

A continuación se presentan dos posibles interpretaciones que ayudan a comprender el propósito de una neurona artificial. La primera es la interpretación matemática de una neurona artificial, junto a ella, se introduce un ejemplo práctico para comprender el funcionamiento interno de una neurona y el proceso de entrenamiento de un modelo. La segunda interpretación aborda las transformaciones espaciales que sufre el espacio de características mediante la hipótesis del *manifold*. Antes de abordar ambas, es crucial definir los siguientes conceptos generales de *ML*:

- Una **muestra** o *ítem* es un dato del conjunto de datos de entrada.
- Una muestra presenta un conjunto de **características** o variables, también conocidas como *features*.
- La **etiqueta** o *label* es la información asociada a la muestra, su categoría. Es el resultado esperado que se utiliza para entrenar un modelo.
- Un **modelo** busca definir la relación entre las características y las etiquetas del conjunto de muestras. Esto se realiza por medio de dos fases, la **fase de entrenamiento** y la **fase de inferencia** o predicción. La primera consiste en crear la fase final del modelo ajustando sus parámetros, la segunda consiste en predecir las etiquetas mediante el modelo creado.
- El conjunto de datos inicial debe dividirse en tres conjuntos: el conjunto de **entrenamiento** (*training set*), que se utiliza para entrenar el modelo; el conjunto de **validación** (*validation set*), que se emplea para validar el rendimiento del modelo mediante métricas durante su entrenamiento; y el conjunto de **prueba** (*test set*), que se utiliza para evaluar el rendimiento del modelo en un conjunto nuevo de datos.

A.2.1. Interpretación matemática de una neurona

Esta interpretación es un resumen de la explicación proporcionada por [Torres \(2020\)](#) en su libro “*Python deep learning: Introducción práctica con Keras y TensorFlow 2*”. La relación más simple entre una característica y la etiqueta es una relación lineal, que se expresa de la siguiente forma:

$$y = wx + b \quad (\text{A.1})$$

- y es la etiqueta de una muestra de entrada.
- x es una característica de la muestra de entrada.
- w es el peso, es un parámetro que tendrá que ajustar el modelo para generar la salida deseada.
- b es el sesgo o *bias*, otro parámetro que debe ser aprendido por el modelo.

Si la muestra de entrada tiene varias características (x_1, x_2, x_3) , cada una de ellas tiene asociado un peso (w_1, w_2, w_3) :

$$y = w_1x_1 + w_2x_2 + w_3x_3 + b \quad (\text{A.2})$$

Esta notación puede expresarse de forma general de la siguiente forma:

$$y = \sum_i w_i x_i + b \quad (\text{A.3})$$

Supongamos que tenemos un conjunto de datos de entrada correspondiente a puntos en un plano. Una muestra del conjunto viene definida por dos coordenadas (x_1, x_2) y una etiqueta que clasifica la forma del punto como [cuadrado, triángulo]. Este problema es un caso de clasificación binaria y utilizaremos un perceptrón para encontrar una recta que separe linealmente los datos de entrada. Buscamos encontrar el modelo que, dado un punto con sus coordenadas, pueda predecir su etiqueta. La

A.2. INTERPRETACIÓN MATEMÁTICA Y GEOMÉTRICA DE UNA NEURONA

Figura A.3 ilustra la recta que queremos calcular para realizar dicha predicción para cualquier punto del plano.

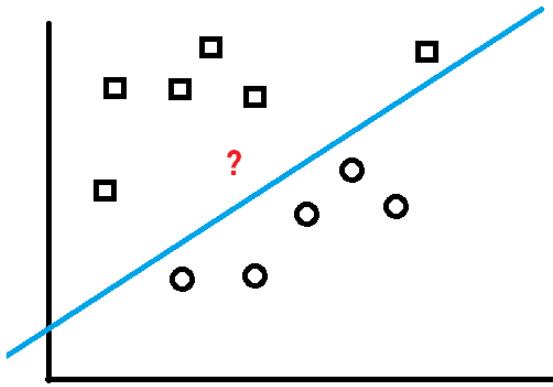


Figura A.3: Representación gráfica de los datos y la recta que los separa.

En este caso concreto, una neurona artificial valdría para resolver este problema de clasificación. La recta que separa linealmente los cuadrados y círculos puede definirse mediante la siguiente Ecuación A.4 de una recta:

$$y = (W_1 * X_1 + W_2 * X_2) + b \quad (\text{A.4})$$

Con estos cálculos, ya se puede construir una neurona artificial para clasificar un nuevo punto del plano. La neurona aplica el vector de pesos W sobre las características de entrada, le suma el sesgo b , y el resultado lo pasa a través de una función de activación, como se muestra en la Ecuación A.5. La función de activación es una función matemática aplicada a la salida de una neurona en una RN artificial. Su propósito principal es introducir no linealidades en el modelo, permitiendo a la red aprender patrones y relaciones más complejas en los datos.

$$\begin{aligned} z &= b + \sum_i x_i w_i \\ y &= \begin{cases} 1 & \text{si } z \geq 0 \\ 0 & \text{si } z < 0 \end{cases} \end{aligned} \quad (\text{A.5})$$

A.2. INTERPRETACIÓN MATEMÁTICA Y GEOMÉTRICA DE UNA NEURONA

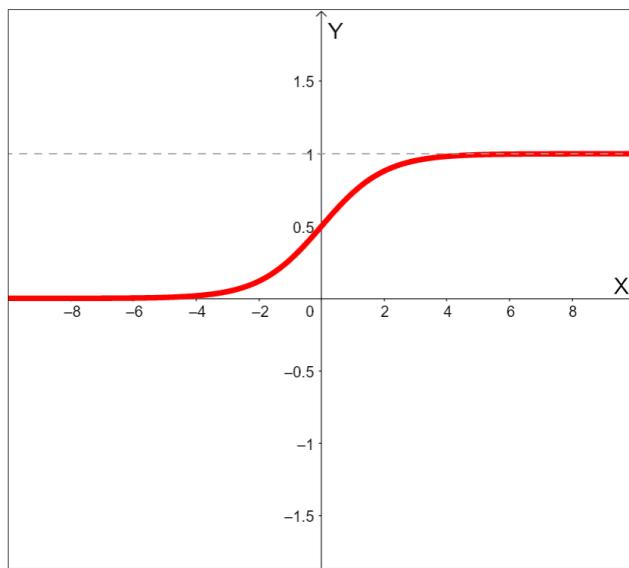


Figura A.4: Representación gráfica de la función *sigmoide*.

En el caso práctico de los puntos del plano, se aplica una función de activación llamada sigmoide que retorna un valor real de salida entre 0 y 1 para cualquier entrada. La Figura A.4 muestra el comportamiento de la función *sigmoide*.

El aprendizaje de una RN se lleva a cabo mediante un proceso iterativo que involucra todas las muestras del conjunto de datos de entrenamiento: se compara el valor de la etiqueta predicho por el modelo en cada iteración con el valor real. Respecto al error calculado por el modelo en cada iteración, se ajustan los valores de W y b con el objetivo de reducir dicho error a medida que se procesan las muestras.

A.2.2. Interpretación de una neurona artificial mediante la hipótesis del *manifold*

En la interpretación matemática previa, se ha mencionado el producto escalar entre las características de entrada y los pesos de la neurona, la suma de un sesgo y la aplicación de una función de activación. En la práctica, estas operaciones se llevan a cabo sobre un tipo especial de vectores llamados tensores. Sin entrar en detalles, un tensor es un objeto que generaliza el concepto de vector y matriz, y se define por los siguientes tres elementos:

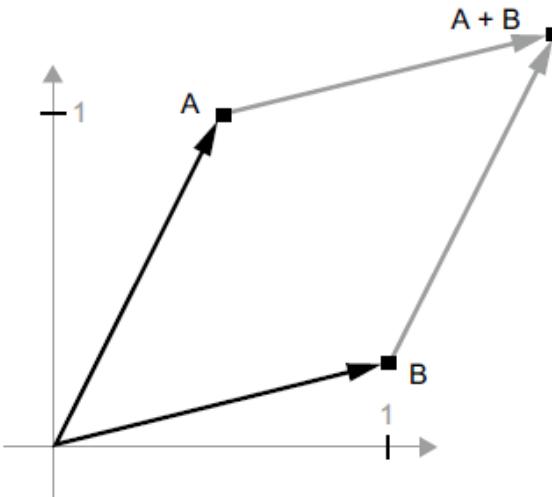


Figura A.5: Representación gráfica de la suma de dos vectores. Fuente: [Chollet \(2021\)](#)

- **Número de ejes:** un tensor que contiene un solo valor se le llama tensor *0D*. Si tiene un vector de elementos, es un tensor *1D*. Si tiene una matriz de elementos, es un tensor *2D*, y así sucesivamente. Por ejemplo, una imagen en blanco y negro se almacena en un tensor *2D*, en el cual cada posición de la matriz corresponde a un píxel de la imagen.
- **Forma:** se define por un vector de enteros que describe cuántas dimensiones tiene el tensor en cada eje. Este vector describe el tamaño de cada eje; es decir, cuántos elementos contiene cada eje.
- **Tipo de datos:** indica el tipo de datos que contiene el tensor, pudiendo ser *uint8*, *float32*, *float64*, entre otros.

Al igual que se pueden realizar operaciones entre vectores, matrices y números, también es posible llevar a cabo operaciones con tensores. En la Figura A.5 se observa la interpretación geométrica de la suma de dos vectores, cada uno con dos coordenadas que definen dos puntos en el espacio. La suma del vector *B* sobre el vector *A* implica la traslación del punto *A* a una distancia específica en el espacio.

Una operación de traslación similar a la suma de dos vectores se puede realizar con los objetos tensoriales. La Figura A.6 ilustra la traslación del tensor *K* de dos

A.2. INTERPRETACIÓN MATEMÁTICA Y GEOMÉTRICA DE UNA NEURONA

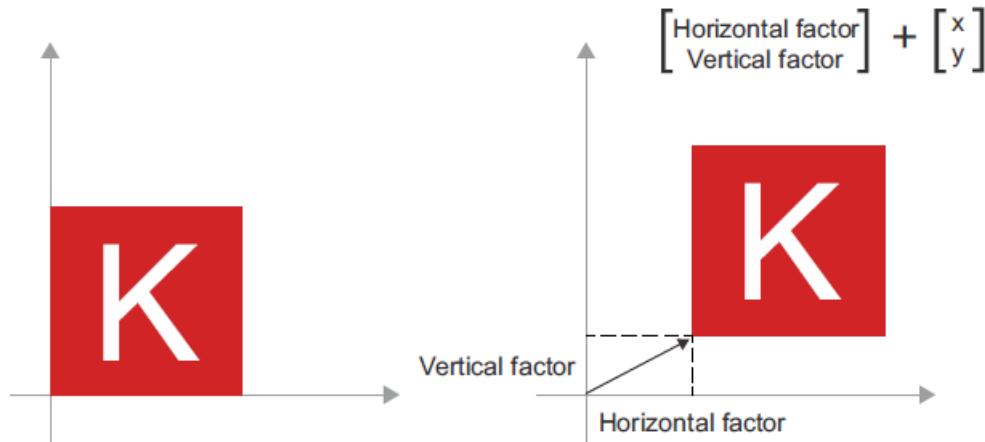


Figura A.6: Representación gráfica de la translación de un tensor K . Fuente: [Chollet \(2021\)](#)

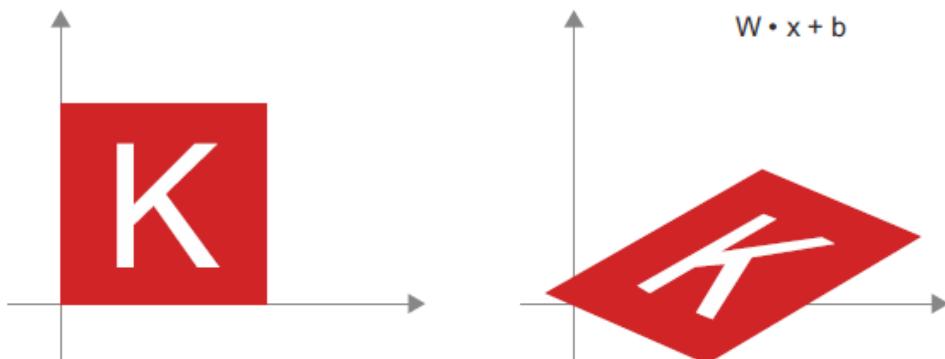


Figura A.7: Representación gráfica de la transformación afín. Fuente: [Chollet \(2021\)](#)

dimensiones mediante la operación de suma.

Existen diversas operaciones, como el producto escalar, la rotación o transformaciones no lineales, que de la misma manera pueden expresarse como operaciones de tensores. En la siguiente Figura A.7 se muestra la interpretación geométrica de las operaciones del producto escalar y suma como sigue la siguiente ecuación provista en la interpretación anterior: $y = wx + b$; A esta operación conjunta se le llama transformación afín o *affine transform*.

Un punto en un plano se define por dos características correspondientes a dos coordenadas. El desafío que enfrenta el *DL* radica en que los datos de entrada pueden

A.3. PROCESO ITERATIVO DE APRENDIZAJE DE UNA RED NEURONAL

contener miles de características, y cada una de ellas representa una dimensión, definiendo así un espacio de características enormemente dimensional. Podemos conceptualizar una RN como un conjunto de transformaciones geométricas complejas que se aplican sobre este espacio de alta dimensión. Estas transformaciones son parametrizadas por los pesos del modelo y se ajustan durante el entrenamiento. La interpretación geométrica de estas operaciones se basa en la hipótesis *manifold*, sugiriendo que los datos de alta dimensión residen en un *manifold* o variedad intrínseca subyacente. El propósito fundamental de estas transformaciones es remodelar y transformar este espacio de características para obtener representaciones más significativas y complejas de los datos de entrada, facilitando así la resolución efectiva de la tarea en cuestión.

A.3. Proceso iterativo de aprendizaje de una Red Neuronal

El proceso de aprendizaje de una RN es iterativo y se repite para todos los datos del conjunto de entrenamiento. Este proceso puede resumirse en tres fases:

1. **Fase de propagación hacia delante**, o *forward propagation*: en esta etapa, la red se expone a los datos de entrenamiento. Los datos cruzan la red, tras experimentando las transformaciones de cada capa de neuronas hasta alcanzar la capa de salida final, donde se obtiene un resultado de predicción. En este punto, se utiliza una función de pérdida para calcular el error cometido (*loss*) por la red entre la predicción realizada y el valor real deseado.
2. **Fase de propagación hacia atrás**, o *backpropagation*: los parámetros se ajustan para reducir la pérdida (*loss*) mediante la retropropagación (*backpropagation*) de la información del error calculado. Esta información se propaga (*forwardpropagation*) hacia las neuronas de las capas ocultas, y cada neurona recibe una fracción de la señal total del error, que representa la

A.3. PROCESO ITERATIVO DE APRENDIZAJE DE UNA RED NEURONAL

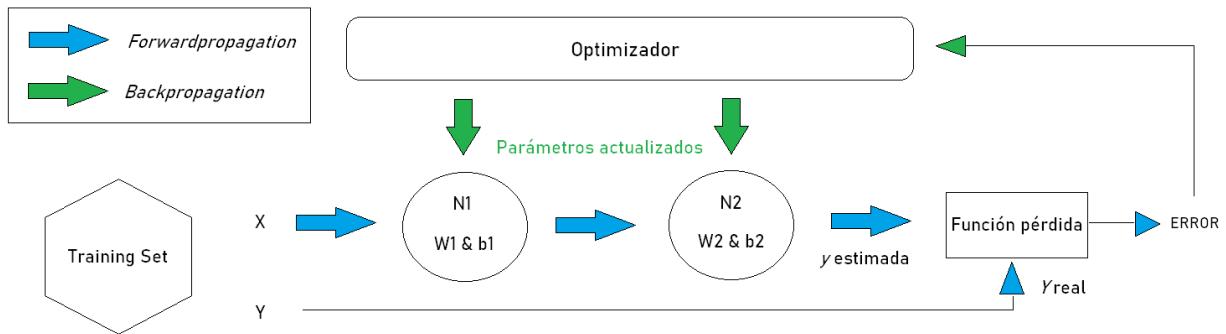


Figura A.8: Esquema del proceso de aprendizaje de una red neuronal.

contribución relativa de cada neurona a la salida final. Una vez retropropagado el error, se ajustan los pesos de cada neurona considerando la contribución del error cometido por la respectiva neurona. El optimizador es responsable de actualizar los valores de los pesos en pequeños incrementos, utilizando las derivadas de la función pérdida y la regla de la cadena.

En la Figura A.8 se ilustra las dos fases del proceso iterativo de aprendizaje de una neurona.

A.3.1. Optimizador y función pérdida

El optimizador es una parte fundamental del proceso de aprendizaje de una RN. Es un algoritmo que se centra en ajustar los parámetros de la red para reducir la *loss* calculada por la función de pérdida. El algoritmo fundamental sobre el que se basan la mayoría de los optimizadores es el *algoritmo de descenso del gradiente*.

La Figura A.9 representa una función convexa $y = x^2$. El mínimo de esta función se encuentra en $x = 0$, donde $y = 0$. La base del algoritmo descenso del gradiente es buscar un mínimo de la función perdida mediante la derivada donde el error sea pequeño. La derivada de una función es la pendiente de la función en un punto específico. Al igualar la derivada a 0, se encuentra el punto donde hay un mínimo (un punto con pendiente nula). En una función convexa, solo hay un punto mínimo global. Sin embargo, en una función no convexa, podemos tener puntos críticos, máximos locales o mínimos

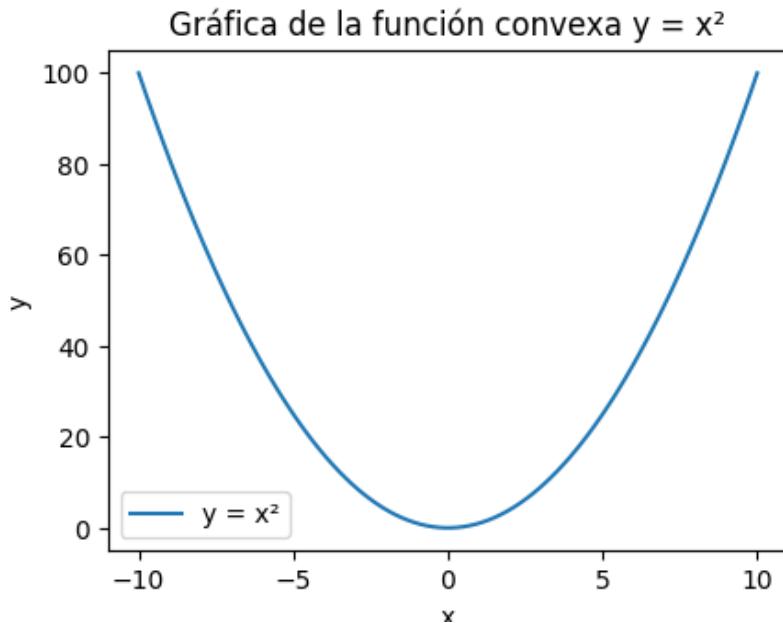


Figura A.9: Ejemplo de función convexa $y = x^2$.

locales, y se busca obtener el mínimo global o un mínimo que sea adecuado.

En el caso de una RN real, la función de pérdida puede depender de miles de parámetros. La Figura A.10 siguiente muestra la representación visual 3D de una función que depende de dos parámetros. Puede imaginar esta función como un terreno montañoso, donde el objetivo es avanzar hacia el punto con la menor altura (donde el error es mínimo).

La derivada de la función respecto a cada parámetro mide la mayor tasa de cambio en la altura del terreno con respecto a un cambio en ese parámetro. La derivada proporciona la dirección en la que debemos movernos para reducir el error de la función de pérdida.

Definimos entonces el gradiente como un vector que contiene las derivadas parciales de una función con respecto a todas sus variables. Este vector proporciona información sobre la dirección de la tasa de cambio más pronunciada de la función en un punto específico. El algoritmo de descenso del gradiente mide el gradiente local de la función de pérdida con respecto al vector de parámetros de la red y se desplaza en la dirección del gradiente descendente.

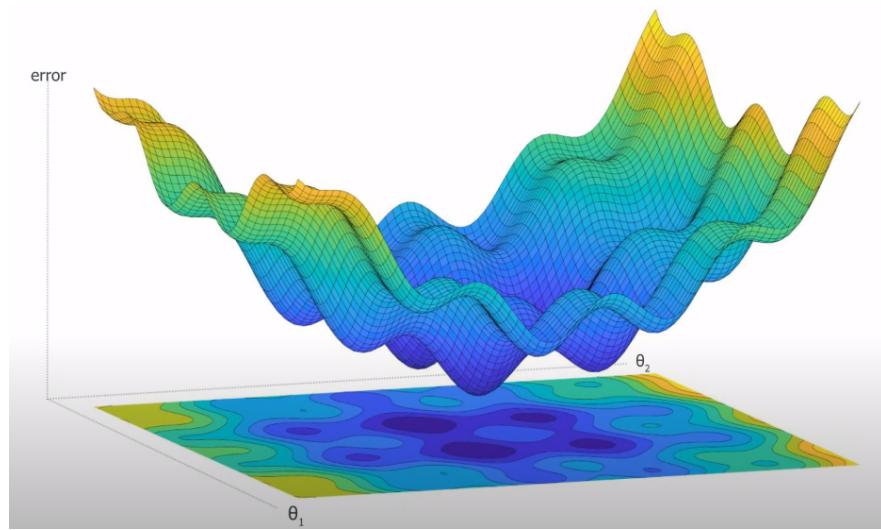


Figura A.10: Representación gráfica de una función pérdida que depende de dos parámetros de la red.

Se introduce el grado de aprendizaje (*learning rate*) como un hiperparámetro¹ que mide cuánto afecta el gradiente a la actualización de nuestros parámetros. En este contexto, se establece su valor mediante experimentación, siendo común utilizar valores en el rango de 0 a 1, donde un valor bajo puede llevar a una convergencia más lenta pero más precisa, y un valor alto puede acelerar la convergencia, aunque con el riesgo de oscilaciones o divergencia.

La fórmula para actualizar los parámetros tiene la siguiente estructura:

$$w_{i+1} = w_i - \alpha \times \nabla L(w) \quad (\text{A.6})$$

- w_{i+1} es el valor del peso que se actualiza para la siguiente iteración del proceso de entrenamiento, donde w_i representa el valor de un peso en la iteración actual.
- La tasa de aprendizaje, denominada por *alpha*, es un hiperparámetro que determina cuánto se ajustan los parámetros de la red durante cada iteración del proceso de entrenamiento.

¹**Hiperparámetro:** es un parámetro externo al modelo que se establece antes del proceso de entrenamiento y afecta la capacidad del modelo para aprender patrones a partir de los datos. Algunos ejemplos de hiperparámetros son la tasa de aprendizaje o el número de capas ocultas en una RN.

- $\nabla L(w)$ representa el gradiente de la función de pérdida con respecto al parámetro que se está ajustando en ese momento. Indica la dirección y magnitud de la mayor tasa de cambio de la función de pérdida en ese punto específico.

A.3.2. Hiperparámetros

Los parámetros de una red neuronal son los coeficientes aprendidos por el modelo, los cuales se obtienen de manera automática mediante el proceso de entrenamiento de la red neuronal. En contraste, los hiperparámetros son variables externas al modelo que no se ajustan durante el entrenamiento, sino que son seleccionados por el arquitecto del modelo y tienen un impacto directo en el rendimiento del proceso de entrenamiento.

Los hiperparámetros se organizan en dos grupos:

- Hiperparámetros a nivel de estructura y topología de la red: algunos de ellos incluyen el número de capas, número de neuronas por capa, función de activación, inicialización de pesos, entre otros.
- Hiperparámetros a nivel de algoritmo de aprendizaje, como el optimizador, tasa de aprendizaje, número de etapas o épocas (*epochs*), tamaño del lote (*batch size*), *callbacks*, etc.

A continuación, se describen de manera introductoria los hiperparámetros más importantes a nivel del algoritmo de aprendizaje en el diseño de un modelo de *DL*:

- **Número de *epochs***: o número de etapas, representan las iteraciones del proceso de entrenamiento, es decir, el número de veces que los datos de entrenamiento son procesados por la red neuronal durante su entrenamiento.

Un número elevado de *epochs* puede conducir a un sobreajuste a los datos de entrenamiento, conocido como *overfitting*, y puede generar problemas de generalización.

Para abordar el sobreajuste del modelo, se emplean los *callbacks*: un *callback* es una herramienta comúnmente utilizada en programación; consiste en pasar

como argumento a una función otra función llamada *callback*. Cuando se realiza una operación en la primera función se llama al *callback* y se le pasa un parámetro asociado a dicha operación. En la librería *Keras*, existen *callbacks* que se emplean para detener el proceso de entrenamiento cuando los resultados en los datos de validación no mejoran en comparación con los resultados de entrenamiento, evitando así el sobreajuste.

Al aumentar el número de *epochs*, la tasa de aprendizaje debe ajustarse correctamente para evitar pasos muy grandes o muy pequeños. El objetivo es controlar el ratio de aprendizaje para que haya un decaimiento gradual a medida que avanza el entrenamiento.

Un valor menor al óptimo puede limitar el potencial del modelo, ya que podría no entrenarse lo suficiente para realizar predicciones precisas.

- **Batch size:** o tamaño de lote, es la cantidad de muestras del conjunto de entrenamiento que son procesadas por el modelo antes de ajustar sus parámetros mediante el optimizador.

La pregunta que surge es: ¿con qué frecuencia se ajustan los valores de los parámetros de la red? Se puede aplicar el ajuste muestra a muestra o una vez por todo el conjunto de datos de entrenamiento, es decir, una vez por *epoch*. La utilización de lotes de un tamaño determinado puede acelerar la ejecución del entrenamiento porque se realizan menos operaciones, además de que puede acelerarse mediante la paralelización con GPU.

- **Learning rate:** o tasa de aprendizaje, es una variable que determina la velocidad de convergencia de la red. Una tasa de aprendizaje alta puede provocar un gran salto en los valores de los parámetros internos de la red. Sin embargo, un valor pequeño puede ralentizar el entrenamiento del modelo al tardar en acercarse a un resultado óptimo.

La mejor solución es emplear una tasa de aprendizaje que se adapte al tiempo; al inicio del entrenamiento, en las primeras etapas, o *epochs*, el aprendizaje

debería ser más rápido con tasas de aprendizaje más grandes, y a medida que se avanza, se realizan ajustes cada vez más pequeños para facilitar la convergencia del modelo hacia el mínimo de la función de pérdida.

- **Optimizador:** es un algoritmo que se centra en ajustar los parámetros de la red para reducir la pérdida calculada por la función de pérdida. En el apartado anterior, se describe en detalle la función del optimizador. Algunos ejemplos de optimizadores son *RMSprop*, *Adam*, descenso del gradiente estocástico (*SGD*), entre otros.
- **Dropout:** el hiperparámetro *dropout* es una técnica utilizada para prevenir el sobreajuste en las redes neuronales. Consiste en aleatoriamente desactivar un porcentaje de las neuronas durante el entrenamiento. Existen diferentes formas de aplicar *dropout* en una red neuronal. Una de ellas es mediante la inclusión de una capa especial llamada *dropout*, que aplica una tasa de anulación a cada peso de cada neurona en cada paso de entrenamiento. Sin embargo, en el caso de las redes neuronales recurrentes (RNN), como las *LSTM*, donde se produce un desenrollado temporal para procesar secuencias de entrada, aplicar *dropout* directamente podría generar problemas. Esto se debe a que una capa *dropout* anularía pesos de manera diferente en cada instante de tiempo del desenrollado, lo que podría afectar negativamente al resultado. Para solventar este inconveniente, existe el parámetro de *dropout* recurrente en las capas *LSTM*. Este parámetro permite aplicar una máscara de *dropout* que sea consistente para cada paso de tiempo del desenrollado de la *LSTM*.

A.3.3. Métricas

Las métricas desempeñan un papel fundamental en el diseño de modelos de *DL*. Durante el entrenamiento, es crucial seleccionar un conjunto de métricas que evalúen el rendimiento del modelo en función de las salidas generadas y las esperadas. Además, una vez que el modelo ha sido entrenado, es necesario realizar una evaluación adicional

A.3. PROCESO ITERATIVO DE APRENDIZAJE DE UNA RED NEURONAL

utilizando un conjunto de datos denominado conjunto de prueba (*test set*), que es independiente del proceso de aprendizaje. De esta manera, se garantiza que el modelo ha logrado una buena generalización² con respecto a nuevos datos.

Existen diversas métricas que presentan un conjunto de ventajas e inconvenientes, dependiendo del dominio del problema. La tarea abordada en el presente TFG se caracteriza por ser de aprendizaje supervisado³. La matriz de confusión es una tabla específica que permite visualizar el rendimiento de un algoritmo en el aprendizaje supervisado. Cada fila de la matriz representa las instancias de una clase actual, mientras que cada columna representa las instancias de una clase predicha. La visualización de la matriz de confusión facilita la comprensión cuando un modelo está confundiendo dos clases. A continuación, se definen cuatro términos importantes: verdaderos positivos, verdaderos negativos, falsos positivos y falsos negativos:

- Verdadero positivo o *True Positive (TP)*: son los casos en los que el modelo predijo correctamente la clase positiva (la clase que estamos interesados en identificar) y la clase real también es positiva.
- Verdadero negativo o *True Negative (TN)*: aquellos casos en los que el modelo predijo correctamente la clase negativa y la clase real también es negativa.
- Falso positivo o *False Positive (FP)*: son los casos en los que el modelo predijo incorrectamente la clase positiva cuando la clase real es negativa.
- Falso negativo o *False Negative (FN)*: son los casos en los que el modelo predijo incorrectamente la clase negativa cuando la clase real es positiva.

²**Generalización:** en el contexto de *DL*, es la capacidad de un modelo de responder con éxito ante nuevos datos de entrada no vistos en su etapa de entrenamiento.

³**Aprendizaje supervisado:** un enfoque en el campo del aprendizaje automático donde un modelo es entrenado utilizando un conjunto de datos que contiene ejemplos de entrada junto con sus correspondientes etiquetas o resultados deseados.

En relación a este trabajo, en los modelos de clasificación se abordarán la exactitud, la precisión, la exhaustividad y el *valor F1*, según lo descrito en Terven y cols. (2023):

- **Exactitud (Accuracy)**: es una métrica común usada en tareas de clasificación y mide la relación que existe entre las muestras clasificadas correctamente con el número total de muestras. Matemáticamente, se representa como:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \quad (\text{A.7})$$

- **Precisión**: mide la precisión de las predicciones positivas. Se define como el número de predicciones positivas verdaderas dividido entre la suma de predicciones verdaderas positivas y predicciones falsas positivas. Puede representarse de la siguiente forma:

$$\text{Precision} = \frac{TP}{TP + FP}, \quad (\text{A.8})$$

- **Exhaustividad (Recall)**: también conocida como sensibilidad, mide la proporción entre los verdaderos positivos sobre el total de instancias positivas. Indica si el modelo es capaz de identificar la mayoría de instancias positivas. Matemáticamente, se representa como:

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TruePositives} + \text{FalseNegatives}} \quad (\text{A.9})$$

- **Valor F1 (F1-Score)**: combina la precisión y la exhaustividad para considerar la capacidad del modelo para predecir los positivos verdaderos y todas las instancias positivas del *dataset*. Se expresa como:

$$F1 = 2 * \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \quad (\text{A.10})$$

A.4. TIPOS DE REDES NEURONALES

En los modelos de regresión, las métricas que se utilizan son tres:

- **MAE** (error absoluto medio): mide la magnitud promedio de los errores entre los valores predichos y los valores reales. Esta métrica se representa de la siguiente forma:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i| \quad (\text{A.11})$$

- **MSE** (error cuadrático medio): mide el promedio de las diferencias al cuadrado entre los valores predichos y los valores reales. Se expresa como:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2 \quad (\text{A.12})$$

- **RMSE** (error de raíz cuadrada media): es la raíz cuadrada del *MSE*. Se utiliza para proporcionar el error en las mismas unidades que la variable objetivo. Matemáticamente, se representa como:

$$\text{RMSE} = \sqrt{\text{MSE}} \quad (\text{A.13})$$

A.4. Tipos de Redes Neuronales

Los perceptrones, como se explica en A.2.1 Interpretación matemática de una neurona artificial, pueden combinarse entre ellos para crear una multitud de arquitecturas RN con el objetivo de resolver tareas específicas. En este proyecto, se abordan tipos particulares de RN, tales como el Perceptrón Multicapa, las Redes Neuronales Convolucionales y las Redes Neuronales Recurrentes.

A.4.1. Perceptrón Multicapa

Cuando todas las neuronas en una capa de una red neuronal están conectadas a cada neurona en la capa anterior, se denomina capa completamente conectada o capa

A.4. TIPOS DE REDES NEURONALES

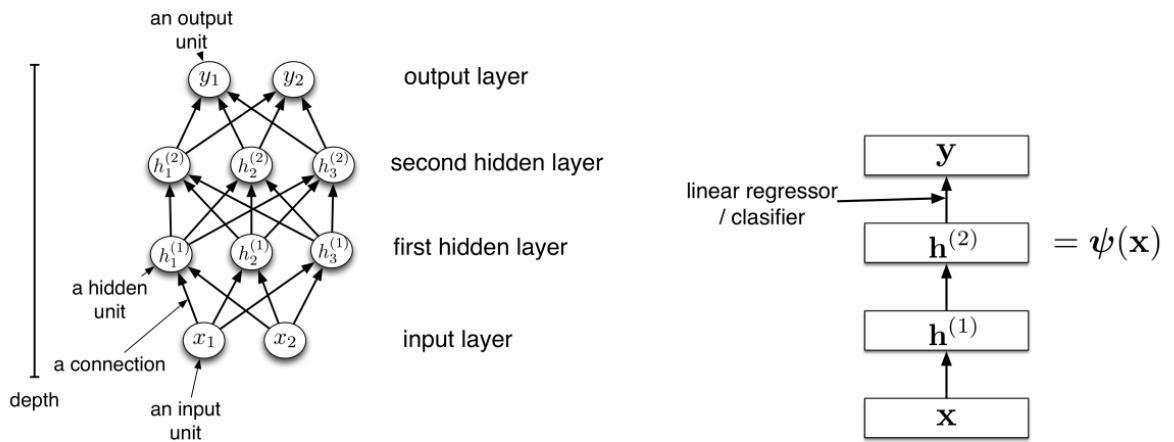


Figura A.11: Perceptrón Multicapa con una capa de entrada, dos capas ocultas y una capa de salida. Fuente: [Grosse \(2019\)](#)

densa (*fully connected neural network*). Una RN con capas densas se llama RN densa o RND.

Un Perceptrón Multicapa o *MLP (Multilayer Perceptron)* es una red neuronal que consta de una capa de entrada (*input layer*), una o más capas compuestas por perceptrones llamadas capas ocultas (*hidden layers*) y una capa final con varios perceptrones denominada capa de salida (*output layer*).

Las capas cercanas a la capa de entrada se denominan capas inferiores, mientras que las capas cercanas a la capa de salida se llaman capas superiores. Es común que las representaciones gráficas del *MLP* incluyan una neurona adicional que representa el sesgo de todas las neuronas en cada capa, excepto en la capa de salida.

La Figura A.11 refleja la representación gráfica de un Perceptrón Multicapa con dos capas ocultas (*hidden layers*). En la parte derecha, se presenta la misma red pero en una jerarquía de capas a un nivel superior.

A.4.2. RN Convolucionales

Las Redes Neuronales Convolucionales (RNC) o *CNN (Convolutional Neural Network* en inglés) son un tipo de red neuronal ampliamente utilizado en el campo

A.4. TIPOS DE REDES NEURONALES

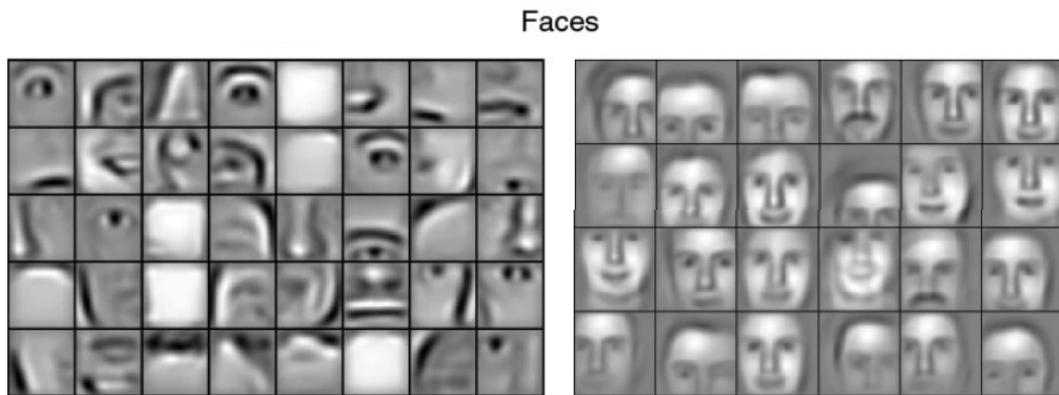


Figura A.12: A la izquierda, patrones generales aprendidos por una capa convolucional más general. A la derecha, patrones complejos de una capa convolucional más profunda. Fuente: [Lee y cols. \(2011\)](#)

de la visión por computadora. La diferencia principal con respecto a las Redes Neuronales Densas (RND) radica en que las RNC están diseñadas para trabajar con imágenes como entrada, permitiendo configurar propiedades específicas para reconocer elementos particulares en las mismas. Pueden ser conceptualizadas como clasificadores especializados en la detección de objetos en imágenes, como por ejemplo, rostros. Al analizar los componentes reconocidos en una imagen, como la nariz, los ojos o las orejas, la red puede predecir si hay un rostro presente en la misma. Por lo tanto, para identificar una cara, la red necesita comprender cómo se configuran los elementos, como los bordes y las formas.

Las capas de una RNC aprenden distintos niveles de abstracción, y a medida que se profundiza en la red, estas capas aprenden representaciones más complejas, aproximándose así a la solución del problema. En la Figura A.12 siguiente se puede observar como una primera capa convolucional aprende elementos generales de una cara, y una segunda capa convolucional aprende patrones compuestos y complejos de caras.

Para lograr su propósito, las RNC emplean cuatro operaciones fundamentales sobre las imágenes:

- **Operación de convolución:** la convolución es una operación clave en las Redes

A.4. TIPOS DE REDES NEURONALES

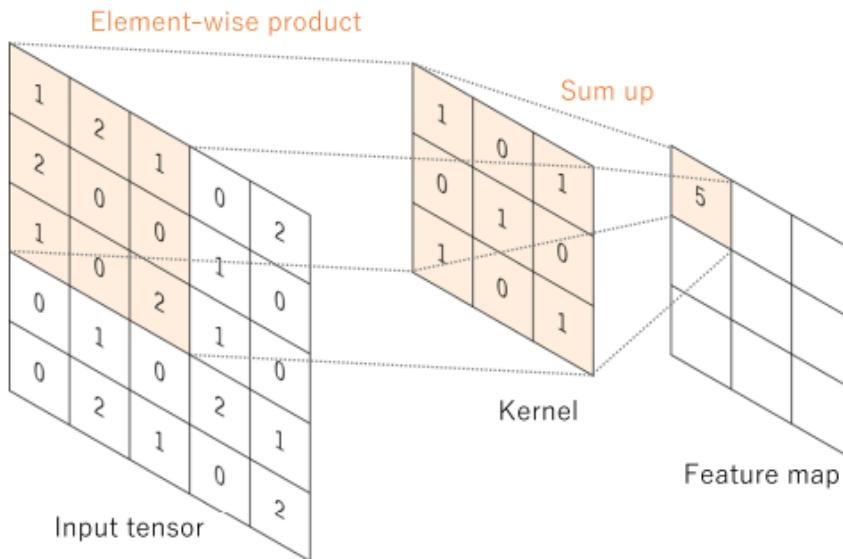


Figura A.13: Operación de convolución sobre un tensor de entrada 5x5 con un kernel 3x3. Fuente: Yamashita y cols. (2018)

Neuronales Convolucionales (RNC) que permite aprender patrones locales en una imagen. A diferencia de las capas densas, que captaron patrones globales, la convolución se centra en ventanas de dos dimensiones en la imagen. Una vez que la red ha aprendido un patrón, como el de una arista, puede reconocerlo en cualquier parte de la imagen. Además, las capas convolucionales pueden aprender jerarquías espaciales de patrones: las primeras capas detectan patrones generales, y las capas finales combinan estos patrones en estructuras más complejas.

En esta operación, se aplica una pequeña matriz de números, llamada *kernel*, a la entrada, que es una matriz de números o tensor. Se calcula el producto elemento por elemento entre cada elemento del *kernel* y el tensor de entrada en cada ubicación, y luego se suman para obtener el valor de salida en la posición correspondiente del tensor de salida, conocido como mapa de características. La convolución es fundamental para detectar patrones locales en datos como imágenes y es esencial en el proceso de extracción de características en redes neuronales convolucionales. La Figura A.13 muestra una representación gráfica de la convolución para una mejor comprensión visual.

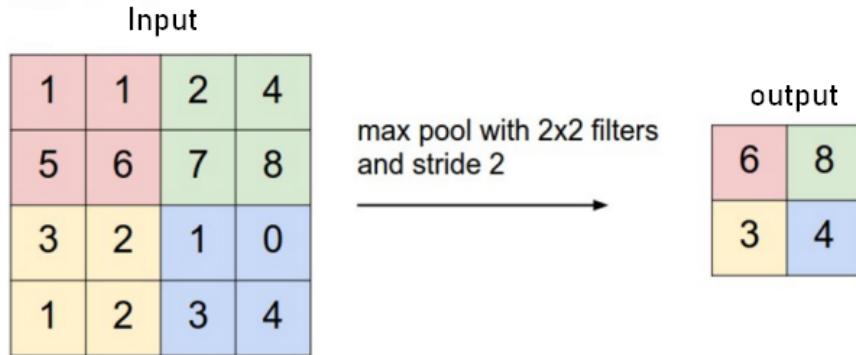


Figura A.14: Operación de *max-pooling* sobre un tensor de entrada 4x4. Fuente: [Jesús \(2020\)](#)

- **Operación de *pooling*:** la operación de *pooling* cumple la función de simplificar la información recopilada por la capa convolucional, creando así una versión condensada de los datos contenidos en dicha capa. Su objetivo principal es reducir la dimensión del mapa de características generado por un filtro, preservando al mismo tiempo la relación espacial entre las distintas características. Esta reducción contribuye a disminuir el número de parámetros de la red, lo que favorece la eficiencia del modelo y acelera el proceso de entrenamiento.

Existen varias versiones de *pooling*, entre las cuales se destacan *max-pooling* y *average-pooling*. *Max-pooling*, por ejemplo, selecciona el valor más alto de una región específica, mientras que *average-pooling* toma el promedio de los valores en dicha región.

La Figura A.14 adjunta ilustra la operación de *max-pooling* utilizando un filtro de tamaño 2x2, donde se selecciona el valor más alto de cada región 2x2 en la imagen de entrada. Este proceso de *pooling* contribuye a resaltar las características más relevantes y significativas en el mapa de características, proporcionando así una representación más condensada y eficaz para la red neuronal convolucional.

A.4. TIPOS DE REDES NEURONALES

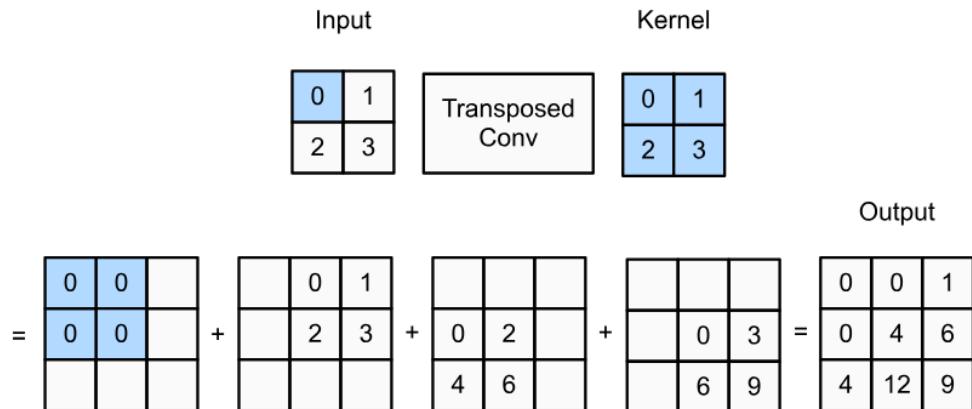


Figura A.15: Operación de convolución transpuesta sobre una entrada 2×2 y con un filtro 2×2 . Fuente: [Banerjee \(2022\)](#)

- **Operación de convolución transpuesta:** en ciertos problemas donde es necesario mantener las dimensiones de la imagen de entrada en la salida, las redes neuronales exclusivamente convolucionales presentan desafíos, ya que aumentan el costo computacional al requerir la preservación de la resolución a lo largo de todas las capas de la red. Esto es especialmente notable en problemas de segmentación de imágenes, donde la salida debe conservar la misma resolución que la entrada. Una capa convolucional transpuesta, por otro lado, es capaz de generar un mapa de características de salida con dimensiones mayores que el mapa de características de entrada. En la Figura A.15 se representa un ejemplo de convolución transpuesta.
- **Operación de aplanamiento (*flattening*):** para que las características de la imagen se introduzcan en una red neuronal, deben representarse en forma de un vector de características. El aplanamiento (*flattening*) es la operación que realiza esta transformación de la matriz de características $2D$ en un vector, representada en la Figura A.16.

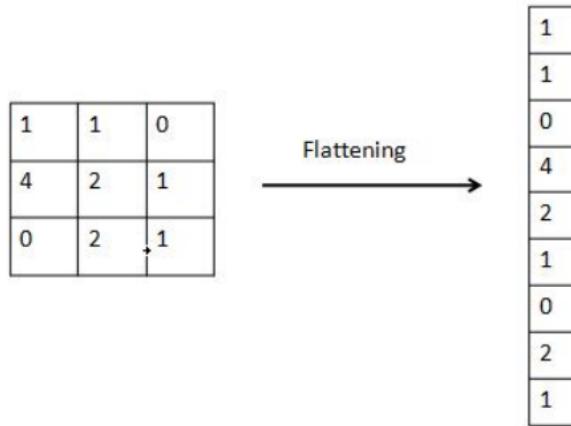


Figura A.16: Operación de aplanamiento de un tensor 3x3 en un vector. Fuente: Tata, Ravi Kumar and Kakumanu, Bhavitha and Teja, Adusumalli and Mothe, Saiteja (2020)

A.4.3. RN Recurrentes

Hasta el momento, las RN presentadas no tienen memoria interna, ya que cada entrada es procesada de manera independiente y no se conserva ningún estado entre ellas. En el caso de una entrada que represente una serie temporal de datos, sería necesario presentar toda la secuencia a la red de una vez. La idea detrás de las redes neuronales recurrentes (*RNN, Recurrent Neural Network*) es procesar una secuencia de datos iterando sobre los elementos de la secuencia, manteniendo un estado que contiene información relevante sobre lo procesado hasta ese momento. Este tipo de red incluye un bucle interno, y el estado interno (o *state*) de la *RNN* se reinicia entre el procesamiento de dos secuencias independientes.

Cada elemento de una secuencia se denomina *timestamp*, y una neurona en una *RNN* puede concebirse como la consecución de procesamientos sucesivos llamados *timestamps* de la serie temporal de datos. Cada punto en el tiempo en el que una neurona procesa un elemento se llama *timestep*. La Figura A.17 describe el desenrollado temporal de una neurona recurrente, donde cada *timestep* recibe el estado interno de la red y el siguiente elemento de la secuencia para generar una salida y actualizar el estado interno.

Sobre el concepto básico de recurrencia, surgen variantes de *RNN* como las *GRU*

A.4. TIPOS DE REDES NEURONALES

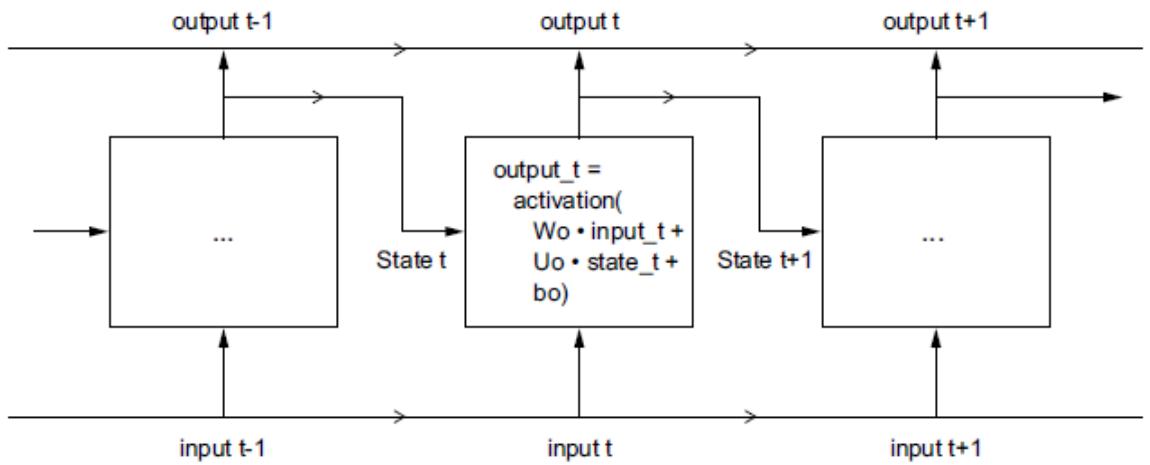


Figura A.17: Desenrollado temporal de una neurona recurrente. Fuente: [Chollet \(2021\)](#)

o *LSTM* (*Long Short Term Memory*), siendo estas últimas las utilizadas en este trabajo. Un problema común que presentan las *RNN* convencionales es el desvanecimiento de información relevante durante la fase de retropropagación (*backpropagation*). Las redes *LSTM* incorporan un mecanismo más avanzado para controlar y modificar la información que se agrega, elimina o conserva en el estado interno de la red.

Apéndice B

Desarrollos complementarios

B.1. Análisis del conjunto de etiquetas generado por la arquitectura *U-Net*

La Tabla B.1 presenta el recuento de etiquetas asociadas a la serie temporal de cada turbina eólica generadas por la arquitectura *U-Net* descrita en el apartado 5.2.3 **Arquitectura *U-Net* previa**. Se puede observar una mayor distribución de las etiquetas 2, 3, 4 y 5 en la turbina eólica 2.

La Tabla B.2 presenta la distribución de etiquetas para las 25 turbinas eólicas en el año 2017. Se observa que las etiquetas 2, 3, 4 y 5 son menos frecuentes, lo que podría representar una dificultad en la fase de prueba de los modelos al contener pocas etiquetas anómalas.

La Tabla B.3 presenta el recuento de las etiquetas para las 25 series temporales de las 25 turbinas eólicas en el año 2016. El número de etiquetas anómalas (2, 3, 4, 5) de la turbina eólica 2 en 2016 muestra un mayor equilibrio. A la hora de abordar el estudio de modelos *baseline*, se decide utilizar la serie temporal de la turbina eólica 2 generada por la *U-Net*, dividiendo la serie temporal en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba. El primero recoge los datos desde 2009 hasta 2015, ambos inclusive; mientras que el segundo incluye el año 2016.

B.1. ANÁLISIS DEL CONJUNTO DE ETIQUETAS GENERADO POR LA ARQUITECTURA U-NET



IdTurbina	Etiq:0	Etiq:1	Etiq:2	Etiq:3	Etiq:4	Etiq:5
1	4858	24699	1265	39	92	214
2	3094	34454	4986	448	228	530
3	2937	26698	6764	689	168	217
4	4483	24058	2095	151	120	339
5	4232	23843	2143	254	112	349
6	5488	28897	2119	135	124	187
7	3188	24105	2884	244	132	315
8	1312	29126	5622	513	151	218
9	3046	30705	2595	123	127	251
10	149	21441	7319	1111	250	330
11	906	16968	5657	720	239	221
12	5266	18434	795	47	71	124
13	2385	20732	1353	93	122	156
14	7179	28712	681	69	161	463
15	6021	23558	896	96	131	277
16	1852	23023	5521	229	158	234
17	6309	27501	2348	164	197	503
18	5561	23171	1502	120	143	248
19	3625	19333	1292	79	93	233
20	3750	32516	5922	314	186	313
21	4171	18907	1074	95	111	184
22	3115	14831	328	52	65	92
23	2806	25806	1735	163	111	240
24	4069	19921	346	33	76	245
25	3424	27612	4987	621	149	247

Tabla B.1: Recuento del número de etiquetas de las 25 series temporales de etiquetas generadas por el modelo *U-Net*.

B.1. ANÁLISIS DEL CONJUNTO DE ETIQUETAS GENERADO POR LA ARQUITECTURA U-NET

IdTurbina	Etiq:0	Etiq:1	Etiq:2	Etiq:3	Etiq:4	Etiq:5
1	344	423	2	0	0	0
2	271	794	1	0	1	7
3	313	597	5	0	2	4
4	331	434	0	0	0	2
5	330	403	0	0	0	34
6	408	497	0	0	0	15
7	311	437	0	0	0	20
8	142	771	2	0	0	6
9	183	726	0	0	0	11
10	19	666	18	0	1	63
11	97	503	11	0	1	2
12	357	256	0	0	0	0
13	241	372	0	0	0	1
14	446	348	0	0	0	95
15	458	305	3	0	0	2
16	215	540	7	0	0	5
17	537	370	0	0	0	13
18	427	298	0	0	0	43
19	326	282	6	0	0	1
20	363	642	4	2	1	62
21	326	286	2	0	0	0
22	239	211	0	0	0	9
23	304	458	4	0	0	2
24	298	297	1	0	0	16
25	321	552	6	0	6	35

Tabla B.2: Recuento del número de etiquetas en el año 2017 de las 25 series temporales generadas por el modelo *U-Net*.

B.1. ANÁLISIS DEL CONJUNTO DE ETIQUETAS GENERADO POR LA ARQUITECTURA U-NET



IdTurbina	Etiq:0	Etiq:1	Etiq:2	Etiq:3	Etiq:4	Etiq:5
1	1021	3051	16	1	1	7
2	471	4908	85	15	58	172
3	769	4004	43	5	4	7
4	976	3085	15	2	4	14
5	907	3114	21	2	0	51
6	1174	3708	18	0	0	13
7	743	3305	32	2	4	10
8	216	4552	85	5	2	54
9	509	4246	32	4	14	15
10	28	3907	103	18	5	26
11	213	2930	60	7	3	15
12	1130	2145	5	0	0	0
13	505	2756	17	2	0	0
14	1613	3290	9	0	0	8
15	1379	2706	11	0	0	3
16	406	3599	27	5	9	54
17	1572	3299	24	1	2	19
18	1347	2736	9	0	0	8
19	905	2356	18	0	1	0
20	866	4816	32	3	3	20
21	996	2270	13	0	0	0
22	702	1716	13	3	1	16
23	649	3373	27	0	0	51
24	790	2463	15	0	0	12
25	814	4056	39	4	4	3

Tabla B.3: Recuento del número de etiquetas en el año 2016 de las 25 series temporales generadas por el modelo *U-Net*.

B.2. Estructura por capas del modelo *U-Net + LSTM propuesto*

Model: "UNET_LSTM_Regression_One_Step"

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
input_2 (InputLayer)	[1, 65, 16, 16, 3]	0	[]
time_distributed_28 (TimeD istributed)	(1, 65, 16, 16, 64)	1792	['input_2[0][0]']
time_distributed_29 (TimeD istributed)	(1, 65, 16, 16, 64)	0	['time_distributed_28[0][0]']
time_distributed_30 (TimeD istributed)	(1, 65, 16, 16, 64)	36928	['time_distributed_29[0][0]']
time_distributed_31 (TimeD istributed)	(1, 65, 8, 8, 64)	0	['time_distributed_30[0][0]']
time_distributed_32 (TimeD istributed)	(1, 65, 8, 8, 128)	73856	['time_distributed_31[0][0]']
time_distributed_33 (TimeD istributed)	(1, 65, 8, 8, 128)	0	['time_distributed_32[0][0]']
time_distributed_34 (TimeD istributed)	(1, 65, 8, 8, 128)	147584	['time_distributed_33[0][0]']
time_distributed_35 (TimeD istributed)	(1, 65, 8, 4, 128)	0	['time_distributed_34[0][0]']
time_distributed_36 (TimeD istributed)	(1, 65, 8, 4, 256)	295168	['time_distributed_35[0][0]']
time_distributed_37 (TimeD istributed)	(1, 65, 8, 4, 256)	0	['time_distributed_36[0][0]']
time_distributed_38 (TimeD istributed)	(1, 65, 8, 4, 256)	590080	['time_distributed_37[0][0]']
time_distributed_39 (TimeD istributed)	(1, 65, 4, 4, 256)	0	['time_distributed_38[0][0]']
time_distributed_40 (TimeD istributed)	(1, 65, 4, 4, 512)	1180160	['time_distributed_39[0][0]']
time_distributed_41 (TimeD istributed)	(1, 65, 4, 4, 512)	0	['time_distributed_40[0][0]']
time_distributed_42 (TimeD istributed)	(1, 65, 4, 4, 512)	2359808	['time_distributed_41[0][0]']
time_distributed_43 (TimeD istributed)	(1, 65, 8, 4, 256)	1179904	['time_distributed_42[0][0]']
Concat3 (Concatenate)	(1, 65, 8, 4, 512)	0	['time_distributed_43[0][0]', 'time_distributed_38[0][0]']

Figura B.1: Primera parte de la arquitectura *U-Net + LSTM*.

B.2. ESTRUCTURA POR CAPAS DEL MODELO U-NET + LSTM PROPUESTO

Concat3 (Concatenate)	(1, 65, 8, 4, 512)	0	['time_distributed_43[0][0]', 'time_distributed_38[0][0]']
time_distributed_44 (TimeD istributed)	(1, 65, 8, 4, 256)	1179904	['Concat3[0][0]']
time_distributed_45 (TimeD istributed)	(1, 65, 8, 4, 256)	0	['time_distributed_44[0][0]']
time_distributed_46 (TimeD istributed)	(1, 65, 8, 4, 256)	590080	['time_distributed_45[0][0]']
time_distributed_47 (TimeD istributed)	(1, 65, 8, 8, 128)	295040	['time_distributed_46[0][0]']
Concat2 (Concatenate)	(1, 65, 8, 8, 256)	0	['time_distributed_47[0][0]', 'time_distributed_34[0][0]']
time_distributed_48 (TimeD istributed)	(1, 65, 8, 8, 128)	295040	['Concat2[0][0]']
time_distributed_49 (TimeD istributed)	(1, 65, 8, 8, 128)	0	['time_distributed_48[0][0]']
time_distributed_50 (TimeD istributed)	(1, 65, 8, 8, 128)	147584	['time_distributed_49[0][0]']
time_distributed_51 (TimeD istributed)	(1, 65, 16, 16, 64)	73792	['time_distributed_50[0][0]']
Concat1 (Concatenate)	(1, 65, 16, 16, 128)	0	['time_distributed_51[0][0]', 'time_distributed_30[0][0]']
time_distributed_52 (TimeD istributed)	(1, 65, 16, 16, 64)	73792	['Concat1[0][0]']
time_distributed_53 (TimeD istributed)	(1, 65, 16, 16, 64)	0	['time_distributed_52[0][0]']
time_distributed_54 (TimeD istributed)	(1, 65, 16, 16, 64)	36928	['time_distributed_53[0][0]']
time_distributed_55 (TimeD istributed)	(1, 65, 16384)	0	['time_distributed_54[0][0]']

Figura B.2: Primera parte de la arquitectura *U-Net + LSTM*.

B.2. ESTRUCTURA POR CAPAS DEL MODELO U-NET + LSTM PROPUESTO

<code>lstm_3 (LSTM)</code>	(1, 65, 32)	2101376	<code>['time_distributed_55[0][0]']</code>
<code>batch_normalization_2 (BatchNormalization)</code>	(1, 65, 32)	128	<code>['lstm_3[0][0]']</code>
<code>lstm_4 (LSTM)</code>	(1, 65, 32)	8320	<code>['batch_normalization_2[0][0]']</code>
<code>batch_normalization_3 (BatchNormalization)</code>	(1, 65, 32)	128	<code>['lstm_4[0][0]']</code>
<code>lstm_5 (LSTM)</code>	(1, 64)	24832	<code>['batch_normalization_3[0][0]']</code>
<code>dense_2 (Dense)</code>	(1, 64)	4160	<code>['lstm_5[0][0]']</code>
<code>dropout_16 (Dropout)</code>	(1, 64)	0	<code>['dense_2[0][0]']</code>
<code>dense_3 (Dense)</code>	(1, 32)	2080	<code>['dropout_16[0][0]']</code>
<code>dropout_17 (Dropout)</code>	(1, 32)	0	<code>['dense_3[0][0]']</code>
<code>Classifier (Dense)</code>	(1, 1)	33	<code>['dropout_17[0][0]']</code>
<hr/>			
Total params: 10698497 (40.81 MB)			
Trainable params: 2140929 (8.17 MB)			
Non-trainable params: 8557568 (32.64 MB)			

Figura B.3: Tercera parte de la arquitectura *U-Net + LSTM*.

Referencias

- Banerjee, A. (2022, 6). Transposed Convolutions (Deep Learning) - Arinjoy Banerjee - Medium.
- Bilendo, F., Meyer, A., Badihi, H., Lu, N., Cambron, P., y Jiang, B. (2022). Applications and modeling techniques of wind turbine power curve for wind farms—A review. *Energies*, 16(1), 180.
- Black, I. M., Richmond, M., y Kolios, A. (2021). Condition monitoring systems: a systematic literature review on machine-learning methods improving offshore-wind turbine operational management. *International Journal of Sustainable Energy*, 40(10), 923-946. Descargado de <https://doi.org/10.1080/14786451.2021.1890736> doi: 10.1080/14786451.2021.1890736
- Cambero Rojas, C. (2023). *Ajuste del diseño de una red neuronal U-Net para la predicción de comportamientos anómalos en turbinas eólicas* (Trabajo Fin de Grado). Escuela Politécnica, Universidad de Extremadura.
- Chollet, F. (2021). *Deep learning with Python* (2.^a ed.). Manning Pub.
- Ciresan, D., Giusti, A., Gambardella, L., y Schmidhuber, J. (2012). Deep neural networks segment neuronal membranes in electron microscopy images. *Advances in neural information processing systems*, 25.
- Delgado Muñoz, E. (2022). *Aplicación de técnicas de clasificación mediante red neuronal U-Net a datos provenientes de turbinas eólicas* (Trabajo Fin de Grado). Escuela Politécnica, Universidad de Extremadura.

REFERENCIAS

- Eriksson, S., Bernhoff, H., y Leijon, M. (2008). Evaluation of different turbine concepts for wind power. *renewable and sustainable energy reviews*, 12(5), 1419–1434.
- Grosse, R. (2019). Multilayer Perceptrons (Lecture 5). *Inf. Téc.*
- Jesús. (2020, 7). *Redes Neuronales Convolucionales en Profundidad*. Descargado de <https://datasmarts.net/es/redes-neuronales-convolucionales-en-profundidad/>
- LeCun, Y., Bottou, L., Bengio, Y., y Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), 2278–2324.
- Lee, H., Grosse, R., Ranganath, R., y Ng, A. Y. (2011). Unsupervised learning of hierarchical representations with convolutional deep belief networks. *Communications of the ACM*, 54(10), 95–103.
- Orozco, C. I., Buemi, M. E., y Berlles, J. J. (2021). CNN–LSTM con mecanismo de atención suave para el reconocimiento de acciones humanas en vídeos. *Elektron*, 5(1), 37–44.
- Petković, D., Čojbašić, Ž., y Nikolić, V. (2013). Adaptive neuro-fuzzy approach for wind turbine power coefficient estimation. *Renewable and Sustainable Energy Reviews*, 28, 191–195.
- Rafaat, S. M., y Hussein, R. (2018). Power maximization and control of variable-speed wind turbine system using extremum seeking. *Journal of Power and Energy Engineering*, 6(01), 51.
- Ronneberger, O., Fischer, P., y Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. En *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III* 18 (pp. 234–241).

REFERENCIAS

- Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6), 386.
- Ruiz, M., Mujica, L. E., Alferez, S., Acho, L., Tutiven, C., Vidal, Y., ... Pozo, F. (2018). Wind turbine fault detection and classification by means of image texture analysis. *Mechanical Systems and Signal Processing*, 107, 149–167.
- Semeniuta, S., Severyn, A., y Barth, E. (2016). Recurrent dropout without memory loss. *arXiv preprint arXiv:1603.05118*.
- Shahriar, M. R., Ahsan, T., y Chong, U. (2013). Fault diagnosis of induction motors utilizing local binary pattern-based texture analysis. *EURASIP Journal on Image and Video Processing*, 2013, 1–11.
- Soother, D. K., Kalwar, I. H., Hussain, T., Chowdhry, B. S., Ujjan, S. M., y Memon, T. D. (2021). A Novel Method Based on UNET for Bearing Fault Diagnosis. *Computers, Materials & Continua*, 69(1).
- Statista. (2023a). *Distribución porcentual de la generación de energía eléctrica en España en 2022, por tipo*. <https://es.statista.com/estadisticas/993747/porcentaje-de-la-produccion-de-energia-electrica-por-fuentes-energeticas-en-espana/>. (Acceso: 08/01/2024)
- Statista. (2023b). *Emisiones globales históricas de CO₂ procedentes de la actividad industrial y los combustibles fósiles de 1757 a 2022*. <https://es.statista.com/estadisticas/635382/emisiones-historicas-de-co2-globales/>. (Acceso: 08/01/2024)
- Sánchez-Fernández, A. J., González-Sánchez, J.-L., Luna-Rodríguez, I., Rodríguez, F. R., y Sánchez-Rivero, J. (2023). Reliability of onshore wind turbines based on linking power curves to failure and maintenance records: A case study in central Spain. *Wind Energy*, 26(4), 349-364. (<https://onlinelibrary.wiley.com/doi/pdf/10.1002/we.2793>) doi: doi.org/10.1002/we.2793

REFERENCIAS

- Tata, Ravi Kumar and Kakumanu, Bhavitha and Teja, Adusumalli and Mothe, Saiteja. (2020). A Model for Assessing the Nature of Car Crashes using Convolutional Neural Networks. *International Journal of Emerging Trends in Engineering Research*, 8, 859-863. doi: 10.30534/ijeter/2020/41832020
- Terven, J., Cordova-Esparza, D. M., Ramirez-Pedraza, A., y Chavez-Urbiola, E. A. (2023). Loss functions and metrics in deep learning. A review. *arXiv preprint arXiv:2307.02694*.
- Torres, J. (2020). *Python deep learning: Introducción práctica con Keras y TensorFlow* 2. Alpha Editorial.
- Turing, A. M. (2012). Computing machinery and intelligence (1950). *The Essential Turing: the Ideas That Gave Birth to the Computer Age*, 433–464.
- Tyme, J. (2023). *Understanding the Key Parts of a Wind Turbine*. Descargado de <https://titanww.com/understanding-the-key-parts-of-a-wind-turbine/> (Acceso: 08/01/2024)
- United States Department of Energy. (2008). *20% Wind Energy by 2030: Increasing Wind Energy's Contribution to U.S. Electricity Supply* (Report n.º DOE/GO-102008-2567). United States Department of Energy.
- Wang, L., y He, D.-C. (1990). Texture classification using texture spectrum. *Pattern recognition*, 23(8), 905–910.
- Wikipedia. (2022). *SCADA*. Descargado de <https://es.wikipedia.org/wiki/SCADA> (Acceso: 08/01/2024)
- Wikipedia. (2023). *Perceptrón*. Descargado de <https://es.wikipedia.org/wiki/Perceptron> (Acceso: 08/01/2024)
- Wikipedia. (2024). *Validación cruzada*. Descargado de https://es.wikipedia.org/wiki/Validacion_cruzada (Acceso: 08/01/2024)

REFERENCIAS

- Wind, T. (2008). Strategic Research Agenda and Market Deployment Strategy—From 2008 To 2030. *European wind Energy Technology Platform, Brussels*.
- Yamashita, R., Nishio, M., Do, R. K. G., y Togashi, K. (2018). Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9, 611–629.
- Yang, C., Zhang, D., y Ye, X. (2022). Analysis of the Development of Distributed Wind Power in China. En *2022 4th international conference on power and energy technology (icpet)* (pp. 655–659).
- Zhang, Y., Zhang, Z., Zhang, Y., Bao, J., Zhang, Y., y Deng, H. (2019). Human activity recognition based on motion sensor using u-net. *IEEE Access*, 7, 75213–75226.