

Centro Universitario de la Defensa en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Identificación de patologías en estructuras mediante deep-learning

Grado en Ingeniería Mecánica

ALUMNO: Jaime Rosa Caballos

DIRECTORES: Jorge Feijoo Conde

Iván Garrido González

CURSO ACADÉMICO: 2025-2026

Universidade Vigo



Centro Universitario de la Defensa
en la Escuela Naval Militar

TRABAJO FIN DE GRADO

Identificación de patologías en estructuras mediante deep-learning

Grado en Ingeniería Mecánica

Intensificación en Tecnología Naval
Cuerpo General

Universidade Vigo

RESUMEN

La detección temprana de patologías en materiales de construcción constituye un factor clave para garantizar la seguridad, durabilidad y sostenibilidad de las edificaciones, especialmente en el ámbito de la conservación preventiva del patrimonio. En este contexto, las técnicas de inspección no destructivas, como la termografía infrarroja, permiten identificar anomalías térmicas asociadas a procesos de degradación antes de que estos sean visibles, facilitando un diagnóstico precoz y fiable.

El presente Trabajo Fin de Grado (TFG) aborda la identificación automática de patologías estructurales mediante la aplicación de técnicas de *deep learning* al análisis de secuencias térmicas. En continuidad con estudios previos, se plantea un enfoque basado en dos etapas: la identificación del material de construcción y la clasificación del tipo de sal presente, considerando que el anión constituye el principal factor de alteración. Para ello, se emplean datos obtenidos a partir de ciclos de calentamiento y enfriamiento registrados mediante termografía infrarroja, aplicando diferentes estrategias de preprocesado.

En este trabajo se evalúan distintos modelos de *deep learning*, entre los que se incluyen las redes neuronales recurrentes *RNN (Recurrent Neural Networks)*, *LSTM (Long Short-Term Memory)* y *GRU (Gated Recurrent Unit)*, analizando su capacidad para modelizar la evolución temporal de las señales térmicas. Los resultados evidencian el potencial de estos modelos para la automatización del diagnóstico, así como sus limitaciones derivadas de la escasez de datos. Este trabajo contribuye al desarrollo de herramientas inteligentes orientadas a la evaluación estructural y al mantenimiento predictivo.

PALABRAS CLAVE

Termografía infrarroja, Ensayos no destructivos, Patologías estructurales, Sales solubles, *Deep learning*

AGRADECIMIENTOS

Me gustaría dedicar unas palabras de agradecimiento a todas aquellas personas que, de una forma u otra, han estado presentes durante la realización de este trabajo y a lo largo de esta etapa.

En primer lugar, quiero agradecer a mis amigos de Málaga, con los que, a pesar de la distancia y de no vernos más allá de periodos vacacionales, siempre he mantenido un vínculo muy especial. Gracias por seguir ahí con el paso del tiempo, por mantener el contacto y por demostrar que la amistad va más allá de la cercanía física.

A mi familia, por ser siempre el principal apoyo. A mi padre, José Manuel, por todo lo que me ha enseñado a lo largo de los años. Aunque en ocasiones pueda extenderse más de la cuenta, sus consejos y los valores que me ha transmitido han sido siempre fundamentales, y les tengo un cariño muy especial. A mi hermano, Álvaro, que, aunque desde Madrid no podamos compartir el día a día, siempre está presente de una forma u otra, y al que quiero profundamente.

Y, de manera muy especial, a mi madre, María del Carmen. Por su fortaleza, su valentía y su capacidad de seguir adelante en los momentos más difíciles. Este trabajo también es para ella, que ha tenido que enfrentarse a situaciones muy duras con los tres cánceres que ha padecido, y que aun así sigue siendo el mayor ejemplo de lucha y superación que tengo. La he tenido muy presente en cada paso de este camino.

Gracias a todos por hacerlo posible.

CONTENIDO

Contenido 1

Índice de Figuras 3

Índice de Tablas 7

1 Introducción y objetivos 8
1.1 Contexto 8
1.2 Objetivos 9
1.3 Estructura del trabajo 9

2 Estado del arte 11
2.1 Patología de los Materiales en la Edificación 11
2.1.1 Alteración y deterioro de los materiales de construcción 11
2.1.1.1 Factores intrínsecos 11
2.1.1.2 Factores extrínsecos 14
2.1.2 Manifestaciones de las patologías estructurales 16
2.2 Técnicas de Inspección y Diagnóstico 18
2.2.1 Métodos tradicionales de inspección 18
2.2.2 Técnicas no destructivas (NDT) en edificación 18
2.3 Termografía infrarroja 19
2.3.1 Fundamentos científicos y digitalización de la información térmica 19
2.3.2 Principios de radiación y leyes físicas en cuerpos reales 20
2.3.3 Radiación detectada y modelos de compensación paramétrica 22
2.3.4 Mecanismos de transporte de calor en materiales porosos 23
2.3.5 Cámara termográfica 23
2.4 Fundamentos de la Inteligencia Artificial y *Deep Learning* 26
2.4.1 Modelos de Deep Learning para datos secuenciales 28
2.4.1.1 Redes Neuronales Recurrentes (RNN) 28
2.4.1.2 Redes LSTM (Long Short-Term Memory) 29
2.4.1.3 Redes GRU (Gated Recurrent Unit) 31
2.4.1.4 Comparación entre modelos 31

3 Desarrollo del TFG 33
3.1 Selección y preparación de materiales 33
3.1.1 Caracterización de los granitos utilizados 34
3.1.1.1 Granito de Madrid 34
3.1.1.2 Granito Azul Alpendorada 35
3.1.2 Roca sedimentaria carbonatada (caliza) 35
3.1.2.1 Caliza Bateig 35
3.1.3 Rocas sedimentarias siliciclásticas (areniscas) 37
3.1.3.1 Arenisca Dorada Urbión 37
3.1.3.2 Arenisca Sierra de la Demanda 38
3.1.3.3 Arenisca Villamayor 39
3.1.4 Identificación de las muestras 40
3.1.5 Sales solubles seleccionadas 41
3.1.5.1 Cloruro de potasio (KCl) – 14 % en peso 41

3.1.5.2 Sulfato de sodio (Na_2SO_4) – 14 % en peso 41
3.1.5.3 Nitrato de sodio ($NaNO_3$) – 14 % en peso 41
3.1.5.4 Sulfato de magnesio ($MgSO_4$) – 14 % en peso 42
3.1.5.5 Cloruro de sodio (NaCl) – 20 % en peso 42
3.1.6 Enumeración de las disoluciones salinas....42
3.2 Técnicas de caracterización mecánica 43
3.2.1 Ultrasonidos....43
3.2.2 Dureza superficial....44
3.3 Contaminación de muestras mediante disoluciones salinas 45
3.3.1 Cuarto ciclo .... **¡Error! Marcador no definido.**
3.3.2 Quinto ciclo .... **¡Error! Marcador no definido.**
3.3.3 Sexto ciclo .... **¡Error! Marcador no definido.**
3.3.4 Séptimo ciclo .... **¡Error! Marcador no definido.**
3.3.5 Octavo ciclo .... **¡Error! Marcador no definido.**
3.4 Termografía infrarroja 45
3.4.1 Grabación térmica de los ciclos de enfriamiento y calentamiento de las muestras 45
3.5 Procesamiento, análisis y modelización de datos térmicos mediante *Deep Learning* 47
3.5.1 Primera etapa. Detección de materiales 48
3.5.2 Segunda etapa. Detección de sales 55

4 Resultados / Validación / Prueba 61
4.1 Caracterización mecánica de las muestras....61
4.1.1 Propiedades de las muestras previo a la adición de sales....61
4.1.2 Propiedades de las muestras tras la adición de sales 62
4.2 Contaminación de las muestras con sales 68
4.3 Análisis termográfico 76
4.4 Análisis de datos con los modelos *Deep Learning* 89
4.4.1 Primera etapa. Detección de Materiales 89
4.4.2 Segunda etapa. Detección de sales 92

5 Conclusiones y líneas futuras....101
5.1 Conclusiones....101
5.2 Líneas futuras 102

6 Bibliografía 104

Anexo I: Implicaciones Sociales, y/o Económicas, y/o Ambientales....108

Anexo II: Reflexiones Éticas y Sociales 109

Anexo III: Especificaciones de la cámara flir 110

Anexo IV: Código empleado para la detección de materiales 111

Anexo V: Código empleado para la detección de sales 131

Anexo VI: Uso ético y responsable de la inteligencia artificial....149

ÍNDICE DE FIGURAS

Figura 1-1. Colapso del centro comercial Algo Centre Mall (Extraído de [2]) 8
Figura 2-1. Microfotografías de areniscas en sección delgada donde se observan los poros intergranulares entre los granos minerales (a) feldespato con evidencias de disolución (b) estructura fantasma ferruginosa resultante de su disolución completa (c-d) formación de caolinita autigénica con morfología vermiforme asociada a la alteración del feldespato (Extraído de [9]) 12
Figura 2-2. Daños por sales debidos a la ascensión capilar a) Muro afectado hasta la altura máxima de ascenso capilar b)Esquema del proceso: ascenso, evaporación y formación de sales c)Etapa inicial de ensayo experimental en caliza porosa d) Etapa intermedia en ensayo experimental: aparición de eflorescencias superficiales y deterioro interno por sales e)Etapa final del ensayo: desarrollo intenso de eflorescencias (Extraído de [12]) 14
Figura 2-3. Alveolización presente en la fachada de la Catedral de Valencia (Extraído de [13]) 17
Figura 2-4. Presencia de eflorescencias en sistemas de mampostería (Extraído de [14]) 17
Figura 2-5. Presencia de criptoflorescencias en muro de ladrillo (Extraído de [15]) 17
Figura 2-6. Diagrama del espectro electromagnético (Composición realizada mediante inteligencia artificial Gemini) 20
Figura 2-7. Gráfica representativa de la Ley de Planck. (a) Los objetos con mayor temperatura emiten principalmente en el infrarrojo de onda media. (b) Los objetos con menor temperatura lo hacen en el infrarrojo de onda larga. (Extraído de [20]) 21
Figura 2-8. Ley de Stefan-Boltzmann (Extraído de [20]) 22
Figura 2-9. Radiación recibida a través de una cámara infrarroja (Elaboración propia con Canva) 22
Figura 2-10. Cámara termográfica FLIR (Elaboración propia) 24
Figura 2-11. Termografía activa vs pasiva (Elaboración propia en Canva a partir de imágenes generadas con inteligencia artificial Gemini) 25
Figura 2-12. Proceso general de aprendizaje de una red neuronal (Extraído de [35]) 27
Figura 2-13. Red Neuronal Recurrente desplegada (Extraído de [37]) 29
Figura 2-14. Arquitectura de la red LSTM (Extraído de [39]) 30
Figura 2-15. Estructura de la red GRU (Extraído de [41]) 31
Figura 3-1. Proceso de dimensionado y descontaminación de las muestras previo al estudio (Elaboración propia) 33
Figura 3-2. Localización de extracción de la muestra de Granito de Madrid (Elaboración propia con canva e inteligencia artificial Gemini) 34
Figura 3-3. Localización de extracción de la muestra de Granito Azul Alpendorada (Elaboración propia con canva e inteligencia artificial Gemini) 35
Figura 3-4. Lugar de extracción de la Caliza Bateig (Elaboración propia con canva e inteligencia artificial Gemini) 36
Figura 3-5. Localización de extracción de la muestra de Arenisca Dorada Urbion (Elaboración propia con canva e inteligencia artificial Gemini) 37
Figura 3-6. Localización de extracción de la muestra de Arenisca de Sierra de la Demanda (Elaboración propia con canva e inteligencia artificial Gemini) 38
Figura 3-7. Localización de extracción de la muestra de Arenisca Villamayor (Elaboración propia con canva e inteligencia artificial Gemini) 39
Figura 3-8. Muestras de referencia con su respectiva identificación de material (Elaboración propia) 40
Figura 3-9. Proceso de elaboración de las disoluciones homogéneas de sales solubles (Elaboración propia) 41
Figura 3-10. Disoluciones de las diferentes sales utilizadas en el proyecto embotelladas y precintadas (Elaboración propia) 42
Figura 3-11. Prueba de velocidad de transmisión de ultrasonidos (Elaboración propia) 43
Figura 3-12. Medición de dureza superficial mediante durómetro (Elaboración propia) 44
Figura 3-13. Muestras contaminadas con Na2SO4 tras el tercer ciclo (Elaboración propia) 68

Figura 3-14. Muestras tras el cuarto ciclo de contaminación con sales solubles a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na2SO4 c) Muestras contaminadas con NaNO3; d) Muestras contaminadas con MgSO4 e) Muestras contaminadas con NaCl (Elaboración propia) 69
Figura 3-15. Muestras tras el quinto ciclo de contaminación con sales solubles a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na2SO4 c) Muestras contaminadas con NaNO3; d) Muestras contaminadas con MgSO4 e) Muestras contaminadas con NaCl (Elaboración propia) 70
Figura 3-16. Muestras tras el sexto ciclo de contaminación con sales solubles a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na2SO4 c) Muestras contaminadas con NaNO3; d) Muestras contaminadas con MgSO4 e) Muestras contaminadas con NaCl (Elaboración propia) 71
Figura 3-17. Muestras tras el séptimo ciclo de contaminación con sales solubles a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na2SO4 c) Muestras contaminadas con NaNO3; d) Muestras contaminadas con MgSO4 e) Muestras contaminadas con NaCl (Elaboración propia) 72
Figura 3-18. Muestras tras el octavo ciclo de contaminación con sales solubles a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na2SO4 c) Muestras contaminadas con NaNO3; d) Muestras contaminadas con MgSO4 e) Muestras contaminadas con NaCl (Elaboración propia) 73
Figura 3-19. Grabación térmica de las muestras con la cámara FLIR E85 (Elaboración propia) 46
Figura 3-20. Proceso de exportación de datos numéricos de enfriamiento de una muestra (Elaboración propia) 47
Figura 3-21. Muestra de una parte de los datos recopilados por píxeles en cada fila y por instante de tiempo en cada columna en el enfriamiento de la muestra de Granito de Madrid sin contaminar (Elaboración propia) .... **¡Error! Marcador no definido.**
Figura 3-22. Representación gráfica del enfriamiento de KCl en las diferentes muestras (Elaboración propia) 92
Figura 3-23. Representación gráfica del enfriamiento de Na2SO4 en las diferentes muestras (Elaboración propia) 93
Figura 3-24. Representación gráfica del enfriamiento de NaNO3 en las diferentes muestras (Elaboración propia) 93
Figura 3-25. Representación gráfica del enfriamiento de MgSO4 en las diferentes muestras (Elaboración propia) 93
Figura 3-26. Representación gráfica del enfriamiento de NaCl en las diferentes muestras (Elaboración propia) 93
Figura 3-27. Representación gráfica del calentamiento de KCl en las diferentes muestras (Elaboración propia) 94
Figura 3-28. Representación gráfica del calentamiento de Na2SO4 en las diferentes muestras (Elaboración propia) 94
Figura 3-29. Representación gráfica del calentamiento de NaNO3 en las diferentes muestras (Elaboración propia) 94
Figura 3-30. Representación gráfica del calentamiento de MgSO4 en las diferentes muestras (Elaboración propia) 95
Figura 3-31. Representación gráfica del calentamiento de NaCl en las diferentes muestras (Elaboración propia) 95
Figura 3-32. Diagrama de flujo de la primera etapa de Predicción de Materiales (Elaborado con inteligencia artificial Gemini) 54
Figura 3-33. Diagrama de flujo de la segunda etapa de Predicción de Sales (Elaborado con inteligencia artificial Gemini) 60
Figura 4-1. Referencia para la toma de datos de las muestras (Elaboración propia) 61

Figura 4-2. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia) 74
Figura 4-3. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia) 74
Figura 4-4. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia) 74
Figura 4-5. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia) 75
Figura 4-6. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia) 75
Figura 4-7. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia) 75
Figura 4-8. Captura inicial y final del enfriamiento de las muestras de Granito de Madrid (Elaboración propia) 77
Figura 4-9. Representaciones gráficas del enfriamiento de las muestras de Granito de Madrid (Elaboración propia) 77
Figura 4-10. Captura inicial y final del enfriamiento de las muestras de Granito Azul (Elaboración propia) 78
Figura 4-11. Representaciones gráficas del enfriamiento de las muestras de Granito Azul (Elaboración propia) 78
Figura 4-12. Captura inicial y final del enfriamiento de las muestras de Caliza Bateig (Elaboración propia) 79
Figura 4-13. Representaciones gráficas del enfriamiento de las muestras de Caliza de Bateig (Elaboración propia) 79
Figura 4-14. Captura inicial y final del enfriamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia) 80
Figura 4-15. Representaciones gráficas del enfriamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia) 80
Figura 4-16. Captura inicial y final del enfriamiento de las muestras de Arenisca de Sierra de la Demanda (Elaboración propia) 81
Figura 4-17. Representaciones gráficas del enfriamiento de las muestras de Arenisca de Sierra de la Demanda (Elaboración propia) 81
Figura 4-18. Captura inicial y final del enfriamiento de las muestras de Arenisca Villanueva (Elaboración propia) 82
Figura 4-19. Representaciones gráficas del enfriamiento de las muestras de Arenisca Villanueva (Elaboración propia) 82
Figura 4-20. Captura inicial y final del calentamiento de las muestras de Granito de Madrid (Elaboración propia) 83
Figura 4-21. Representaciones gráficas del calentamiento de las muestras de Granito de Madrid (Elaboración propia) 83
Figura 4-22. Captura inicial y final del calentamiento de las muestras de Granito Azul (Elaboración propia) 84
Figura 4-23. Representaciones gráficas del calentamiento de las muestras de Granito Azul (Elaboración propia) 84
Figura 4-24. Captura inicial y final del calentamiento de las muestras de Caliza Bateig (Elaboración propia) 85
Figura 4-25. Representaciones gráficas del calentamiento de las muestras de Caliza Bateig (Elaboración propia) 85
Figura 4-26. Captura inicial y final del calentamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia) 86
Figura 4-27. Representaciones gráficas del calentamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia) 86

Figura 4-28. Captura inicial y final del calentamiento de las muestras de Arenisca Sierra de la Demanda (Elaboración propia) 87
Figura 4-29. Representaciones gráficas del calentamiento de las muestras de Arenisca Sierra de la Demanda (Elaboración propia) 87
Figura 4-30. Captura inicial y final del calentamiento de las muestras de Arenisca Villanueva (Elaboración propia) 88
Figura 4-31. Representaciones gráficas del calentamiento de las muestras de Arenisca Villanueva (Elaboración propia) 88
Figura 4-32. Gráficas de validación y entrenamiento en la predicción de Materiales con valores absolutos y dt=3 (Elaboración propia) 89
Figura 4-33. Gráficas de validación y entrenamiento en la predicción de Materiales sin valores absolutos y dt=5 (Elaboración propia) 91
Figura 4-34. Gráficas de validación y entrenamiento en la predicción de Materiales sin valores absolutos y dt=3 (Elaboración propia) 91
Figura 4-35. Gráficas de validación y entrenamiento en la predicción de Materiales sin valores absolutos y dt=1 (Elaboración propia) 92
Figura 4-36. Gráficas de validación y entrenamiento en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=5 y sin logaritmo (Elaboración propia) 95
Figura 4-37. Gráficas de validación y entrenamiento en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=7 y con logaritmo (Elaboración propia) 96
Figura 4-38. Gráficas de validación y entrenamiento en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=1 y con logaritmo (Elaboración propia) 97
Figura 4-39. Gráficas de validación y entrenamiento en la predicción de Sales aplicando áreas acumuladas, sin valores absolutos, dt=5 y sin logaritmo (Elaboración propia) 98
Figura 4-40. Precisión óptima obtenida en los modelos de *Deep-Learning* (Elaboración propia) 100
Figura 5-1. Caso práctico para la predicción de patología en una estructura 102
Figura A3-1. Especificaciones técnicas de cámaras termográficas FLIR Exx-Series. Las características de la cámara utilizada (E85) se encuentran en la tercera columna (Extraído de 110

ÍNDICE DE TABLAS

Tabla 2-1. Resumen comparativo entre los modelos recurrentes (Elaboración propia) 32
Tabla 3-1. Referencia de las muestras con el tipo material y sal empleados (Elaboración propia) 40
Tabla 4-1. Medición de ultrasonidos en las muestras previo a la contaminación con sales (Elaboración propia) 62
Tabla 4-2. Medición de la prueba de dureza en las muestras previo a la contaminación con sales (Elaboración propia) 62
Tabla 4-3. Medición de ultrasonidos en las muestras tras la contaminación con sales (Elaboración propia) 65
Tabla 4-4. Medición de la prueba de dureza en las muestras tras la contaminación con sales (Elaboración propia) 68
Tabla 4-5. Tabla de validación en la predicción de Materiales con valores absolutos y dt=3 (Elaboración propia) 90
Tabla 4-6. Tabla de validación en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=5 y sin logaritmo (Elaboración propia) 96
Tabla 4-7. Tabla de validación en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=7 y con logaritmo (Elaboración propia) 97
Tabla 4-8. Tabla de validación en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=1 y con logaritmo (Elaboración propia) 98
Tabla 4-9. Tabla de validación en la predicción de Sales aplicando áreas acumuladas, sin valores absolutos, dt=1 y con logaritmo (Elaboración propia) 99

1 INTRODUCCIÓN Y OBJETIVOS

1.1 Contexto

La detección temprana de patologías en materiales de construcción constituye un aspecto fundamental para garantizar la durabilidad, seguridad y conservación de las estructuras, especialmente en entornos exigentes como los asociados al ámbito naval y marítimo. En estos contextos, la presencia de agentes externos como la humedad o las sales solubles acelera significativamente los procesos de degradación, dando lugar a alteraciones que, en muchos casos, no son visibles en sus etapas iniciales. Este tipo de deterioro progresivo puede comprometer gravemente la integridad estructural si no se identifica a tiempo.

La relevancia de estos procesos queda reflejada en diversos fallos estructurales documentados a lo largo de las últimas décadas. Un caso especialmente representativo es el colapso del centro comercial Algo Centre Mall, en Elliot Lake (Canadá), en el año 2012 [1]. En este suceso, la infiltración prolongada de agua con sales de deshielo provocó una corrosión severa en los elementos estructurales metálicos, reduciendo progresivamente su sección resistente hasta desencadenar el colapso parcial de la cubierta. Este fallo no fue repentino, sino el resultado de un proceso de degradación acumulativo que no fue correctamente identificado durante años, a pesar de la presencia de indicios visibles como filtraciones y deterioro superficial. Este tipo de situaciones pone de manifiesto la dificultad de detectar patologías internas mediante técnicas convencionales y la necesidad de desarrollar metodologías más avanzadas que permitan identificar estos procesos en fases tempranas (ver Figura 1-1).

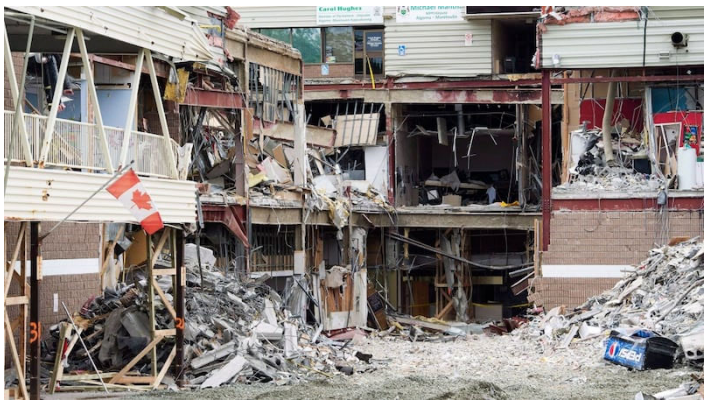


Figura 1-1. Colapso del centro comercial Algo Centre Mall (Extraído de [2])

1.2 Objetivos

El objetivo principal es ampliar el alcance del trabajo presentado en la referencia [3], correspondiente a un Trabajo de Fin de Grado del CUD ENM realizado hace dos años. En particular, este trabajo aborda el desarrollo y la evaluación de tres modelos distintos de *deep learning*, con el fin de comparar su rendimiento y determinar la arquitectura más adecuada. A diferencia del estudio previo, centrado exclusivamente en la identificación de la presencia y el tipo de sal, en el presente trabajo se introduce un enfoque más completo, permitiendo no solo la detección del tipo de sal, sino también la identificación del material de construcción, independientemente de su estado (contaminado o no). Para ello, se emplean como datos de entrada las evoluciones temporales de la temperatura obtenidas mediante termografía infrarroja en una cara representativa de cada muestra. Asimismo, se ha ampliado tanto el número de muestras analizadas como la variedad de sales consideradas, con el objetivo de mejorar la robustez y aplicabilidad del modelo desarrollado:

- Materiales empleados: dos tipos de granitos, un tipo de caliza y dos tipos de arenisca (en el trabajo anterior se empleó tres tipos de granitos y dos tipos de morteros de cal.

- Sales empleadas: Cloruro de Potasio, Cloruro de Sodio, Sulfato de Magnesio, Sulfato de Sodio y Nitrato de Sodio. En el trabajo anterior se emplearon sólo el Cloruro de Sodio, Sulfato de Sodio y Nitrato de Sodio.

Para ello, se plantea el estudio de la respuesta térmica de los distintos materiales sometidos a ciclos de calentamiento y enfriamiento, con el fin de identificar patrones característicos asociados a sus propiedades físicas y a las sales contaminantes. A partir de este análisis mediante la monitorización a partir de una cámara termográfica, se propone desarrollar un procedimiento de preprocesado de las secuencias de las imágenes térmicas que permita resaltar la información más relevante mediante modelos de aprendizaje automático, en este caso, la predicción del tipo de material y de la sal (el aniónante).

En concreto, sobre estas señales procesadas, se pretende implementar y evaluar distintas arquitecturas de redes neuronales recurrentes, analizando su capacidad para capturar la dinámica temporal de los datos térmicos, empleando las arquitecturas más utilizadas en el estado del arte: RNN (*Recurrent Neural Network*), LSTM (*Long Short-Term Memory*) y GRU (*Gated Recurrent Unit*).

Asimismo, se plantea analizar la influencia de las distintas configuraciones de los datos de entrada a los modelos de *deep learning*, con el objetivo de determinar las configuraciones más adecuadas para un aprendizaje óptimo y evaluar las limitaciones del enfoque propuesto, especialmente en relación con la disponibilidad de datos.

1.3 Estructura del trabajo

El presente Trabajo Fin de Grado (TFG) se organiza en varios capítulos que permiten desarrollar de forma progresiva los objetivos planteados.

En primer lugar, en el Capítulo 1 se introduce el contexto del trabajo, se exponen los objetivos perseguidos y se describe la estructura general del documento.

El Capítulo 2 recoge el estado del arte, en el que se revisan los principales conceptos relacionados con las patologías asociadas a la presencia de sales en materiales de construcción, la termografía infrarroja, y las técnicas de *deep learning* aplicadas al análisis de datos.

En el Capítulo 3 se describe la metodología empleada, incluyendo la preparación de los materiales y de las disoluciones salinas, la contaminación de las sales en los materiales, la caracterización mecánica mediante ultrasonidos y dureza superficial previo y posterior a la contaminación de las muestras, la campaña de las mediciones térmicas y el desarrollo de los modelos de aprendizaje profundo utilizados en el estudio.

El Capítulo 4 presenta los resultados obtenidos, tanto respecto a la caracterización mecánica como al comportamiento de los modelos tanto en la clasificación de materiales como en la predicción de sales.

Finalmente, en el Capítulo 5 se exponen las conclusiones del trabajo y se proponen posibles líneas futuras de investigación orientadas a la mejora y ampliación del estudio.

2 ESTADO DEL ARTE

Este capítulo se centra en el estudio de los procesos de deterioro que afectan a los materiales utilizados en la edificación, analizando especialmente el papel que desempeñan las sales solubles en su degradación. Junto con la presencia de humedad, estas sustancias constituyen uno de los factores más influyentes en la aparición de daños constructivos, además de representar uno de los problemas más complejos desde el punto de vista de su control y tratamiento. En el ámbito de la construcción es habitual encontrar distintos tipos de sales, como carbonatos, fosfatos, cloruros, nitratos y sulfatos, siendo estos últimos, junto con los cloruros, los más frecuentes y perjudiciales debido a su elevada capacidad de alteración.

A lo largo del capítulo se examinan los distintos mecanismos mediante los cuales estas sales provocan patologías en los materiales, prestando especial atención a las edificaciones situadas en entornos marítimos, como las instalaciones de la Armada, donde las condiciones ambientales intensifican estos fenómenos. Asimismo, se estudia la influencia de variables climáticas, especialmente la temperatura y el grado de humedad, en la evolución de los procesos de degradación. Finalmente, se revisan las principales técnicas empleadas actualmente para la detección y evaluación de estos daños, destacando el uso de métodos no invasivos y, en particular, la termografía infrarroja, por su rapidez, eficiencia y potencial como herramienta de diagnóstico de bajo coste integrada en sistemas basados en aprendizaje profundo.

2.1 Patología de los Materiales en la Edificación

2.1.1 Alteración y deterioro de los materiales de construcción

El deterioro de los materiales pétreos empleados en edificación es consecuencia de la acción conjunta de distintos factores que influyen en su comportamiento a lo largo del tiempo. Para su estudio, resulta habitual clasificar dichos factores en dos grandes grupos: los factores intrínsecos, vinculados a las propiedades físicas, químicas y estructurales propias del material, y los factores extrínsecos, relacionados con el entorno y con los agentes externos que actúan sobre él. Esta distinción permite analizar con mayor precisión los mecanismos de alteración que afectan a la roca y facilita la posterior identificación de las causas del deterioro y de sus manifestaciones visibles [4].

2.1.1.1 Factores intrínsecos

Los factores intrínsecos son aquellos que dependen directamente de la naturaleza del material pétreo y que condicionan su calidad, así como su resistencia frente a los procesos de deterioro. Estos factores

están estrechamente relacionados con las características petrográficas que la roca adquiere durante su formación geológica, las cuales determinan en gran medida su comportamiento cuando es utilizada como material de construcción [5].

Entre estos factores destaca, en primer lugar, la composición mineralógica y química de la roca. La vulnerabilidad de un material pétreo comienza en su escala microscópica, ya que la naturaleza química de los minerales que lo constituyen define su reactividad frente a los fluidos presentes en el entorno. Muchos minerales que se formaron en condiciones geológicas específicas resultan termodinámicamente inestables en la superficie terrestre. Por ello, al entrar en contacto con el aire, el agua y los organismos vivos, tienden a experimentar procesos de transformación o descomposición química con el fin de alcanzar un nuevo estado de equilibrio en el entorno superficial. Además, determinados minerales presentan una reactividad específica que condiciona el tipo de alteración predominante. Así, por ejemplo, los feldespatos son especialmente susceptibles a los procesos de hidrólisis, mediante los cuales pueden transformarse en minerales arcillosos, mientras que los minerales carbonatados son más propensos a disolverse en medios acuosos, especialmente cuando estos presentan cierto grado de acidez [5].

Otro de los factores intrínsecos de gran relevancia es la porosidad y el sistema poroso del material pétreo. Desde el punto de vista petrofísico, la red de espacios vacíos presente en la roca constituye la principal vía de entrada y circulación de fluidos, especialmente agua y soluciones salinas, que intervienen en numerosos procesos de alteración. La naturaleza de este sistema poroso varía según el tipo de roca: en las rocas detríticas predomina la porosidad intergranular, generada por los espacios existentes entre los granos minerales, mientras que en las rocas cristalinas la circulación de fluidos se produce principalmente a través de sistemas de fisuras y microfisuras .

El tamaño, la distribución y la conexión entre los poros condicionan además el comportamiento del material frente al agua y otros fluidos. Los poros de menor diámetro presentan una mayor superficie de contacto entre el sólido y los fluidos, lo que favorece las reacciones químicas y aumenta la retención de agua en el interior de la roca. De este modo, el sistema poroso regula procesos como la absorción capilar, la evaporación del agua o la generación de presiones internas asociadas a la cristalización de sales o a la congelación del agua. Asimismo, el incremento de la porosidad suele ir acompañado de una disminución de la resistencia mecánica del material, ya que el aumento del volumen de poros y fisuras reduce la cohesión entre los granos minerales y, en consecuencia, la capacidad resistente de la roca (ver Figura 2-1) [4], [5], [6], [7], [8].

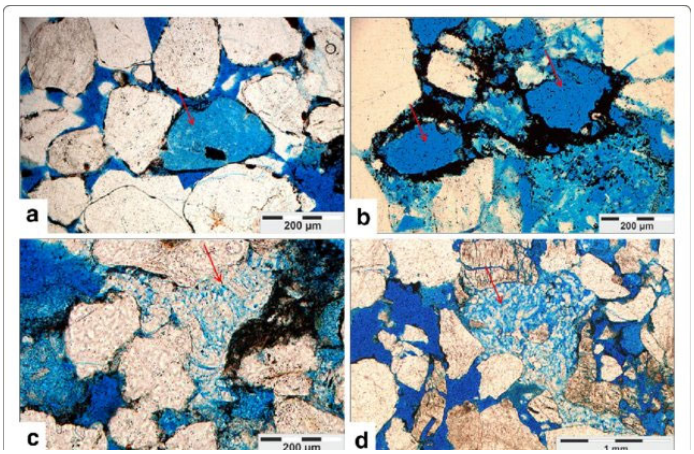


Figura 2-1. Microfotografías de areniscas en sección delgada donde se observan los poros intergranulares entre los granos minerales
(a) feldespato con evidencias de disolución
(b) estructura fantasma ferruginosa resultante de su disolución completa
(c-d) formación de caolinita autigénica con morfología vermiforme asociada a la alteración del feldespato
(Extraído de [9])

Los caracteres estructurales, relacionados con la textura y la fábrica de la roca, constituyen otro aspecto fundamental que influye en su comportamiento frente al deterioro. La organización espacial de los minerales y su modo de agregación pueden generar planos de debilidad estructural que condicionan la forma en que el material responde a los distintos agentes de alteración. En este contexto, muchas rocas presentan anisotropía, es decir, propiedades físicas que varían en función de la dirección considerada. Este fenómeno suele estar asociado a la orientación preferencial de determinados minerales, como las micas en el granito, o a estructuras sedimentarias como los planos de estratificación en las areniscas. Cuando los elementos pétreos se colocan en obra sin respetar esta orientación natural, por ejemplo situando los planos de estratificación en posición vertical en una fachada, se incrementa la probabilidad de que se produzcan lesiones mecánicas, como procesos de exfoliación o desprendimiento [5].

En algunos casos, la propia estructura cristalina del material puede presentar ciertas vulnerabilidades de fábrica que favorecen su deterioro. Determinadas disposiciones mineralógicas, como las fábricas idiotípicas zonadas, pueden mostrar una predisposición inherente a la alteración. En estos casos, la disolución diferencial de los núcleos de los cristales puede generar porosidad intragranular o intracristalina, lo que conduce a una progresiva disgregación del material y a fenómenos de arenización [10].

Finalmente, las propiedades térmicas de las rocas también influyen en su durabilidad. Los materiales pétreos están sometidos continuamente a variaciones de temperatura ambiental que provocan procesos de dilatación y contracción en los granos minerales. Cuando la roca presenta anisotropía, estas dilataciones no se producen de manera uniforme, lo que puede generar tensiones internas acumulativas y dar lugar a fenómenos de fatiga térmica [11]. Asimismo, los gradientes térmicos bruscos, como los que se producen entre el día y la noche o entre zonas expuestas al sol y zonas en sombra, pueden generar tensiones suficientemente elevadas como para provocar la fracturación de piezas pétreas de gran tamaño, incluso en ausencia de cargas mecánicas significativas [6]. En el caso de los granitos, por ejemplo, la dilatación térmica suele ser mayor en la dirección paralela a los planos de orientación de las micas, debido a la menor amortiguación que presentan los espacios existentes en esa dirección [11].

Las propiedades mecánicas constituyen otro de los factores intrínsecos que influyen de manera decisiva en el comportamiento de los materiales pétreos frente a los procesos de deterioro. Estas propiedades determinan la capacidad del soporte pétreo para resistir los esfuerzos internos que se generan como consecuencia de distintos procesos patológicos que afectan al material a lo largo del tiempo. En este sentido, la resistencia mecánica de la roca condiciona su durabilidad, ya que actúa como un límite frente a las tensiones internas que pueden producirse en su estructura[10].

Uno de los aspectos más relevantes es la relación entre la resistencia del material y las presiones internas que se generan en el interior de la roca. La durabilidad del material pétreo se ve comprometida cuando las tensiones originadas por determinados procesos superan su resistencia a la tracción. Este fenómeno puede observarse, por ejemplo, en los procesos de cristalización de sales en el interior de los poros, donde el crecimiento de los cristales produce un aumento de volumen que genera presiones internas significativas. De forma similar, los procesos de heladicidad asociados a la congelación del agua presente en los poros pueden ejercer presiones muy elevadas, que en algunos casos alcanzan valores de $500 kg/cm^2$. Cuando estas tensiones superan la resistencia mecánica del material, se producen fracturas, disgregación granular o pérdida de cohesión del soporte pétreo [5], [6].

Asimismo, existe una estrecha relación entre las propiedades mecánicas y la porosidad del material. Numerosos estudios han demostrado que el incremento de la porosidad accesible suele ir acompañado de una disminución de la resistencia mecánica, especialmente en lo que respecta a la resistencia a compresión y a flexotracción. A medida que aumenta el volumen de poros y fisuras en el interior de la roca, la cohesión entre los granos minerales se reduce, lo que provoca una pérdida progresiva de la

capacidad resistente del material. En consecuencia, los materiales que han experimentado un mayor grado de alteración presentan generalmente una menor coherencia estructural y una mayor susceptibilidad a la rotura cuando se ven sometidos a esfuerzos mecánicos [4], [11].

2.1.1.2 Factores extrínsecos

Además de las propiedades inherentes al material pétreo, el estado de conservación de los elementos constructivos depende en gran medida de la acción de diversos agentes externos. Estos factores extrínsecos comprenden el conjunto de condiciones ambientales y antrópicas que actúan sobre el material una vez incorporado a la edificación, condicionando el ritmo y la intensidad de su deterioro. A diferencia de los factores intrínsecos, vinculados a la naturaleza y características petrográficas de la roca, los factores extrínsecos están relacionados con el entorno en el que se encuentra el edificio y con los procesos físicos, químicos y biológicos que se desarrollan en él. La interacción entre estos agentes puede generar microclimas específicos en determinadas zonas del edificio, lo que favorece la aparición de mecanismos de degradación más intensos que los que experimentaría el material en su estado natural en cantera [5], [8].

Entre todos los agentes externos, el agua constituye uno de los factores más determinantes en los procesos de alteración de los materiales de construcción. En el ámbito de la edificación, el agua interviene en la mayoría de los mecanismos de deterioro, ya sea como reactivo químico o como medio de transporte de sustancias disueltas. Su presencia facilita reacciones químicas, moviliza sales solubles y contribuye a la generación de presiones internas en el interior de los materiales porosos [11].

El agua puede penetrar en los materiales a través de diferentes mecanismos. Uno de los más frecuentes es la ascensión capilar (ver Figura 2-2), fenómeno por el cual el agua procedente del subsuelo, o de escorrentías filtradas, asciende a través de la red porosa del material debido a la tensión superficial. Según la ley de Jurin, la altura que puede alcanzar el agua es inversamente proporcional al radio de los poros, de modo que los materiales con porosidad fina tienden a presentar ascensiones capilares más elevadas [4], [6].

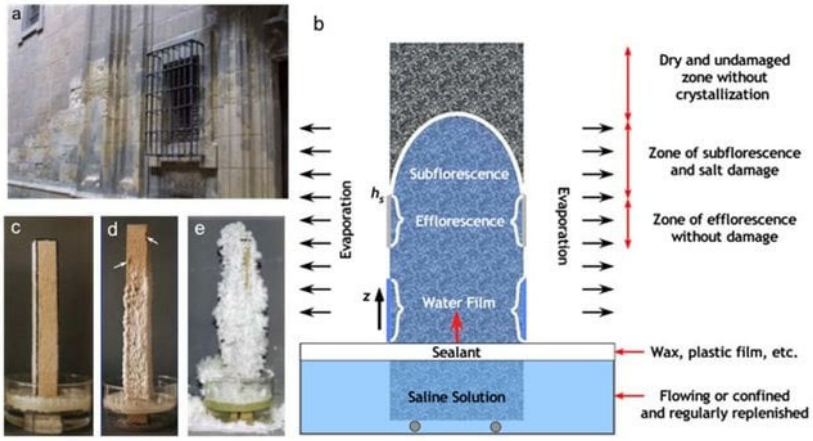


Figura 2-2. Daños por sales debidos a la ascensión capilar
a) Muro afectado hasta la altura máxima de ascenso capilar
b)Esquema del proceso: ascenso, evaporación y formación de sales
c)Etapa inicial de ensayo experimental en caliza porosa
d) Etapa intermedia en ensayo experimental: aparición de eflorescencias superficiales y deterioro interno por sales
e)Etapa final del ensayo: desarrollo intenso de eflorescencias
(Extraído de [12])

Otro mecanismo habitual es la infiltración o filtración de agua líquida a través de fisuras, juntas o discontinuidades presentes en el material. Este proceso puede producirse por efecto de la gravedad, por la presión hidrostática o por la acción del viento, facilitando la penetración directa del agua en el interior del sistema poroso. Asimismo, el vapor de agua presente en el ambiente puede condensarse cuando entra en contacto con superficies cuya temperatura se encuentra por debajo del punto de rocío. Este fenómeno es especialmente frecuente en las denominadas paredes frías, asociadas a la presencia de puentes térmicos deficientes [6].

En zonas costeras o en entornos próximos al mar, la presencia de sales solubles adquiere una relevancia particular. Los aerosoles marinos transportan cloruros y otras sales altamente higroscópicas que penetran en los poros de los materiales pétreos y cerámicos. Cuando las condiciones de humedad y temperatura cambian, estas sales pueden disolverse y recristalizar repetidamente. Durante la evaporación del agua, las sales disueltas precipitan formando depósitos cristalinos. Si la cristalización tiene lugar en la superficie del material se originan eflorescencias, que se manifiestan como manchas blanquecinas de carácter principalmente estético. Sin embargo, cuando la cristalización se produce en el interior de los poros se generan criptoflorescencias o subeflorescencias, que provocan importantes presiones internas capaces de superar la resistencia a tracción del material y desencadenar procesos de disgregación mecánica [6].

La contaminación atmosférica constituye otro factor de degradación significativo, especialmente en entornos urbanos o industriales. Gases como el dióxido de azufre (SO_2) y el dióxido de carbono (CO_2) reaccionan con la humedad atmosférica formando soluciones ácidas que dan lugar al fenómeno conocido como lluvia ácida. Mientras que el dióxido de carbono favorece la disolución de los carbonatos de calcio presentes en muchas rocas, el dióxido de azufre puede provocar procesos de sulfatación que transforman el carbonato cálcico en yeso. Este compuesto suele acumularse en forma de costras negras que atrapan partículas de hollín y polvo atmosférico; cuando estas costras se desprenden, arrastran consigo parte del material original [8].

Las condiciones climáticas también influyen de forma notable en el deterioro de los materiales pétreos. Las variaciones de temperatura generan procesos de dilatación y contracción diferencial entre los distintos minerales que componen la roca, produciendo tensiones internas que pueden dar lugar a microfisuras o fracturas progresivas [5]. Un fenómeno particularmente destructivo es el asociado a los ciclos de hielo y deshielo. Cuando el agua retenida en los poros se congela, experimenta un aumento de volumen cercano al 9 %, lo que genera presiones internas que pueden alcanzar valores muy elevados. Este proceso resulta especialmente perjudicial en materiales saturados y con porosidad fina, donde la repetición de ciclos de congelación y deshielo provoca la degradación progresiva del material [6].

El viento actúa igualmente como agente de erosión mecánica. Al transportar partículas en suspensión, puede desgastar las superficies pétreas, suavizar relieves y generar cavidades o alvéolos superficiales que modifican la morfología original del material [6].

Junto a los factores físicos y químicos, los organismos vivos desempeñan también un papel relevante en los procesos de deterioro. Este fenómeno, conocido como biodeterioro, se produce cuando distintos organismos colonizan los sustratos pétreos, alterándolos tanto desde el punto de vista físico como químico [7]. Microorganismos y algas favorecen la retención de humedad sobre las superficies, creando condiciones propicias para la colonización de plantas superiores. Por su parte, líquenes y musgos segregan ácidos orgánicos, como los oxalatos, capaces de disolver parcialmente el sustrato mineral. Además, sus estructuras de anclaje pueden penetrar en fisuras y poros, ejerciendo presiones mecánicas que contribuyen a la degradación del material [2]. Del mismo modo, la presencia de determinados animales puede generar alteraciones notables; los excrementos de aves, especialmente de palomas, poseen un carácter ácido que provoca reacciones químicas con la superficie pétrea, además de causar un importante impacto estético [5], [8], [10].

Finalmente, no debe olvidarse la influencia de los factores antrópicos, es decir, aquellos derivados de la actividad humana. En muchas ocasiones, las intervenciones constructivas o de restauración inadecuadas constituyen una de las principales causas de deterioro. El empleo de elementos metálicos como espigas o anclajes de hierro puede generar problemas cuando estos sufren procesos de oxidación. La corrosión del hierro implica un aumento considerable de volumen, que puede alcanzar hasta un 250%, generando tensiones internas capaces de fracturar el material circundante. Del mismo modo, el uso de morteros de cemento excesivamente rígidos o con elevado contenido en sales solubles sobre fábricas de piedra blanda puede provocar incompatibilidades mecánicas y químicas que favorecen el desprendimiento del material original. A estos problemas se suma la falta de mantenimiento de los elementos constructivos, como la ausencia de limpieza en canalones y bajantes, que puede provocar acumulaciones de agua y la aparición de humedades persistentes en los paramentos [6], [8].

2.1.2 Manifestaciones de las patologías estructurales

Las lesiones o manifestaciones de deterioro constituyen los efectos visibles que los distintos procesos de alteración producen sobre los materiales y los elementos constructivos. En el diagnóstico estructural resulta fundamental diferenciar estas manifestaciones de sus causas, ya que una misma lesión puede ser consecuencia de diferentes mecanismos de degradación. El análisis de la morfología del daño permite identificar el origen de la patología y evaluar el grado de pérdida de material o de capacidad resistente del elemento afectado [2].

Entre las manifestaciones más frecuentes se encuentran los procesos de disgregación y arenización, caracterizados por la pérdida progresiva de cohesión interna del material. En el caso de la arenización, los granos minerales que componen la piedra o el ladrillo se separan gradualmente hasta desprenderse en forma de partículas semejantes a arena. Este fenómeno es habitual en rocas con matrices débiles o en aquellas que presentan determinadas disposiciones cristalinas en las que la disolución diferencial de los núcleos minerales genera porosidad intracristalina. De manera similar, la descomposición del material puede producirse como consecuencia de procesos químicos de alteración mineralógica, como la hidrólisis de los feldespatos, que da lugar a la formación de minerales arcillosos o la disolución de carbonatos, lo que debilita progresivamente la estructura pétrea [5], [10].

Otro grupo de lesiones frecuentes está relacionado con la pérdida de material en forma de láminas o fragmentos, comprometiendo directamente la sección resistente del elemento constructivo. La exfoliación consiste en la separación del material en capas paralelas a la superficie, fenómeno que suele estar asociado a la anisotropía de la roca o a la orientación inadecuada de sus planos de estratificación. Cuando estos planos se disponen verticalmente en elementos como sillares o placas, las tensiones internas generadas por cambios térmicos o por cristalización de sales favorecen la apertura de estas superficies naturales. Cuando la pérdida de material afecta a placas de mayor espesor se habla de desplacación o desconchado, procesos que se originan generalmente cuando las presiones internas generadas por la cristalización de sales o por la formación de hielo superan la resistencia a tracción del material. En los revestimientos de fachada también pueden producirse desprendimientos debidos a la falta de adherencia entre capas o a diferencias en los coeficientes de dilatación térmica entre los distintos materiales que componen el sistema constructivo [6].

Otra manifestación característica es la alveolización (ver Figura 2-3), que se identifica por la aparición de cavidades irregulares distribuidas sobre la superficie del paramento, generando un aspecto similar al de un panal de abeja. Este fenómeno suele ser consecuencia de la acción combinada del viento, que transporta partículas erosivas, y de la cristalización de sales en zonas protegidas de la lluvia directa, donde la evaporación es constante [6].



Figura 2-3. Alveolización presente en la fachada de la Catedral de Valencia (Extraído de [13])

Las alteraciones asociadas a la cristalización de sales constituyen algunas de las patologías más visibles en ambientes húmedos o costeros. Las eflorescencias (ver Figura 2-4) aparecen como depósitos blanquecinos o amarillentos que precipitan sobre la superficie al evaporarse el agua que transporta las sales disueltas. Aunque su impacto inicial es principalmente estético, su presencia indica la existencia de circulación interna de humedad en el material. Más perjudiciales resultan las criptoflorescencias (ver Figura 2-5), que se desarrollan bajo la superficie y generan presiones internas capaces de provocar abombamientos, disgregaciones y desprendimientos del material superficial. Debido a su carácter interno, este tipo de patología suele detectarse en fases avanzadas del deterioro, lo que dificulta su diagnóstico y su tratamiento efectivo [5], [6].



Figura 2-4. Presencia de eflorescencias en sistemas de mampostería (Extraído de [14])



Figura 2-5. Presencia de criptoflorescencias en muro de ladrillo (Extraído de [15])

Las fisuras, grietas y fracturas constituyen otro tipo de manifestaciones frecuentes. Las fisuras superficiales suelen presentar una abertura reducida y afectan principalmente a los revestimientos, aunque facilitan la penetración de agua y agentes contaminantes. Las grietas estructurales, por el contrario, atraviesan elementos portantes y suelen estar asociadas a movimientos de cimentación, empujes estructurales o sobrecargas no previstas. En estructuras de hormigón armado puede producirse además la rotura del recubrimiento debido a la corrosión de las armaduras metálicas. La oxidación del acero genera un aumento considerable de volumen que provoca tensiones internas capaces de fracturar el hormigón circundante, dejando el acero expuesto y acelerando el proceso de deterioro estructural [6], [8].

Finalmente, existen alteraciones que afectan principalmente al aspecto visual de los materiales. Las alteraciones cromáticas y la formación de pátinas pueden indicar procesos químicos en curso. Entre ellas destacan las costras negras, originadas por la reacción del dióxido de azufre con el carbonato cálcico de

la piedra, dando lugar a la formación de yeso que incorpora partículas de contaminación atmosférica. Del mismo modo, pueden aparecer manchas de oxidación de tonalidad parda o rojiza asociadas a la corrosión de elementos metálicos presentes en la fachada. Asimismo, la colonización biológica por hongos, algas o líquenes provoca cambios de coloración y textura en la superficie del material, además de contribuir a su degradación a largo plazo mediante la retención de humedad y la secreción de compuestos químicos agresivos [6], [10].

2.2 Técnicas de Inspección y Diagnóstico

La realización de un diagnóstico estructural riguroso constituye un elemento clave para asegurar la estabilidad, la seguridad y la vida útil de los edificios. En el caso de las construcciones vinculadas al entorno de la Armada, el deterioro suele estar asociado a la acción prolongada de diversos agentes ambientales y a procesos de degradación que se desarrollan de forma progresiva [16].

Por este motivo, un análisis eficaz no debe limitarse únicamente a la detección de daños visibles, sino que debe profundizar en el estudio de sus causas, integrando el conocimiento de las patologías con las condiciones climáticas y los antecedentes constructivos del inmueble. Asimismo, la elaboración de una cartografía de daños resulta indispensable como fase previa a cualquier intervención, ya que permite localizar y representar con precisión las lesiones existentes mediante herramientas como la imagen digital, la fotogrametría y los sistemas LiDAR (*Light Detection And Ranging)*[16].

2.2.1 Métodos tradicionales de inspección

Tradicionalmente, la evaluación del estado de los edificios se ha basado en la inspección visual, que constituye una primera aproximación al diagnóstico. Sin embargo, esta metodología presenta limitaciones, ya que depende en gran medida de la experiencia del técnico y no permite obtener información cuantitativa precisa [17].

Para superar estas limitaciones, se han empleado técnicas destructivas basadas en la extracción de muestras del material. Estas permiten determinar propiedades físico-mecánicas como la resistencia a compresión, la porosidad o la absorción de agua [17]. No obstante, este tipo de ensayos implica una alteración del elemento constructivo y solo proporciona información puntual del estado del material.

En el análisis de materiales pétreos, técnicas como la difracción de rayos X o la microscopía permiten caracterizar la composición mineralógica y la microestructura. En particular, el estudio de sales mediante técnicas analíticas resulta clave, ya que la presencia de cloruros, sulfatos o nitratos está directamente relacionada con los procesos de degradación [16].

Sin embargo, el carácter invasivo de estas técnicas y su limitada capacidad para representar el estado global del edificio han impulsado el desarrollo de métodos alternativos basados en técnicas no destructivas [18].

2.2.2 Técnicas no destructivas (NDT) en edificación

Las técnicas no destructivas (NDT, *Non-Destructive Testing*) permiten evaluar el estado de los materiales sin alterar su integridad, lo que las convierte en herramientas fundamentales en el diagnóstico del patrimonio construido [18]. Estas técnicas permiten analizar tanto propiedades superficiales como internas, facilitando una evaluación más completa del estado del material.

El uso combinado de distintas técnicas resulta especialmente relevante, ya que cada una proporciona información complementaria. La integración de resultados mejora la interpretación de las patologías y reduce la incertidumbre en el diagnóstico [16].

Entre las principales técnicas destacan:

- Velocidad de pulsos ultrasónicos (UPV, *Ultrasonic Pulse Velocity*): permite evaluar la integridad del material mediante la medición de la velocidad de propagación de ondas. Esta velocidad está relacionada con propiedades mecánicas como la resistencia a compresión y permite detectar fisuras o discontinuidades internas [19].

- Tomografía de resistividad eléctrica (ERT, *Electrical Resistivity Tomography*): permite identificar variaciones en la resistividad del material, asociadas a la presencia de humedad, sales o discontinuidades internas [16].

- Termografía infrarroja: permite detectar anomalías térmicas en superficie sin contacto, siendo útil para identificar zonas con diferente comportamiento higrotérmico o defectos ocultos [16].

- Ensayos de impedancia acústica: permiten caracterizar el estado superficial del material a partir de su respuesta acústica, siendo sensibles a cambios en la porosidad y degradación superficial [16].

- Conductividad eléctrica y capacitancia: permiten medir el contenido de humedad en superficie y en profundidad. La presencia de sales incrementa la conductividad eléctrica, lo que permite identificar zonas afectadas por procesos de deterioro [16].

Estas técnicas permiten detectar tanto patologías superficiales como internas, incluyendo fisuras, vacíos, pérdida de material o acumulación de humedad, sin necesidad de intervenir directamente sobre el elemento constructivo [17].

La humedad y la cristalización de sales constituyen uno de los principales mecanismos de degradación en materiales de construcción, ya que generan tensiones internas que provocan disgregación y pérdida de cohesión [16]. Por ello, el uso de técnicas no destructivas resulta esencial para evaluar estos procesos y establecer estrategias de conservación adecuadas.

2.3 Termografía infrarroja

2.3.1 Fundamentos científicos y digitalización de la información térmica

La termografía infrarroja (IRT, *Infrared Thermography*) se define como la ciencia que se ocupa de la obtención y el procesamiento de datos térmicos mediante el uso de instrumentos de medición que no requieren contacto físico con el objeto de estudio [20]. Esta tecnología aprovecha la existencia de la radiación infrarroja, una forma de energía electromagnética con longitudes de onda superiores a las de la luz visible, situándose específicamente por debajo del color rojo en el espectro [21]. El principio físico rector de esta disciplina establece que cualquier cuerpo real su temperatura se encuentra por encima del cero absoluto (0 K o −273,15 °C) emitiendo radiación infrarroja de manera natural [22]. Dado que el ojo humano no tiene la capacidad biológica de percibir este rango espectral, es imprescindible el uso de dispositivos optoelectrónicos para captar esta energía y transformarla en señales eléctricas que puedan ser procesadas digitalmente [20].

A diferencia de las imágenes convencionales en el espectro visible, que se basan en la luz reflejada por los objetos, los termogramas representan la radiación emitida por la propia escena, convirtiéndola en la fuente primaria de energía [21]. Esta característica otorga a la IRT (*Infrared Termography)* ventajas críticas para la inspección técnica de materiales: es una herramienta no destructiva y no invasiva, proporciona mapas bidimensionales de temperatura que permiten comparar distintas áreas de una muestra en tiempo real y, a diferencia de otras técnicas como los rayos X, carece de efectos nocivos derivados de radiación ionizante [22]. Estas propiedades son esenciales para monitorizar de forma segura y prolongada la cinética térmica de muestras sometidas a ciclos de calentamiento y enfriamiento [20].

El proceso de digitalización térmica comienza en el detector de la cámara, un transductor que transforma la radiación incidente en un voltaje proporcional [20]. La calidad de los datos técnicos resultantes depende de dos fases fundamentales: el muestreo y la cuantización. El muestreo está determinado por la resolución espacial de la cámara, es decir, el número de elementos detectores en su matriz de plano focal (FPA, *Focal Plane Array*). Por su parte, la cuantización define el número de niveles digitales o bits utilizados para representar la intensidad de la señal medida.

2.3.2 Principios de radiación y leyes físicas en cuerpos reales

La interacción entre la temperatura de un material y la energía que este radia está gobernada por leyes físicas que describen la distribución espectral de la energía. El espectro infrarrojo (ver Figura 2-6) se subdivide habitualmente en bandas según la longitud de onda: infrarrojo cercano (NIR, *Near Infrared*, 0,75–3 µm), de onda corta (SWIR, *Short-Wave Infrared*), de onda media (MWIR, *Mid-Wave Infrared*, 3–6 µm) y de onda larga (LWIR, *Long-Wave Infrared*, 6–15 µm) [22].

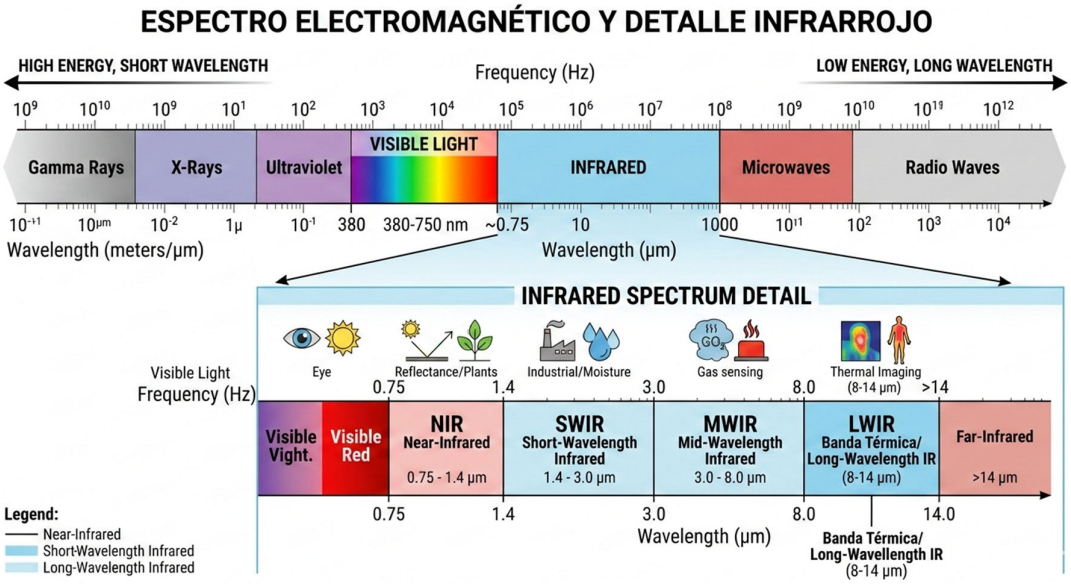


Figura 2-6. Diagrama del espectro electromagnético
(Composición realizada mediante inteligencia artificial Gemini)

Para el análisis de materiales de construcción a temperaturas ambientales, se prefiere mayoritariamente la banda LWIR (*Long-Wave Infrared*) debido a que la atmósfera presenta ventanas de alta transmitancia en este rango y a que el pico de emisión de los objetos a temperatura ambiente se sitúa precisamente en estas longitudes[20] [22].

La base teórica de la emisión térmica reside en el concepto de cuerpo negro, un radiador ideal que absorbe la totalidad de la radiación incidente sin reflejar ni transmitir energía [20].

La ley de Planck (ver Ecuación 1) describe la radiancia espectral de este cuerpo ideal $W_{\lambda,b}$ ($W\ m^{-2}\ sr^{-1}\ m^{-1}$) en función de su temperatura absoluta (T) y la longitud de onda (λ):

$$W_{\lambda,b} = \frac{C_1 \lambda^{-5}}{e^{\frac{C_2}{\lambda T}} - 1} \qquad \text{(Ecuación 1)}$$

Donde $C_1 = 3.7418 \times 10^{-16}\ [W\ m^2]$ y $C_2 = 1{,}4388 \times 10^{-2}\ [m\ K]$ son las constantes universales de radiación [20] [21]. A partir de esta ley, se deduce la ley de desplazamiento de Wien (ver Ecuación 2), que permite identificar la longitud de onda donde se produce el pico de máxima intensidad radiante (λ_{peak}):

$$\lambda_{peak} = \frac{2898}{T}\ [\mu m \cdot K] \qquad \text{(Ecuación 2)}$$

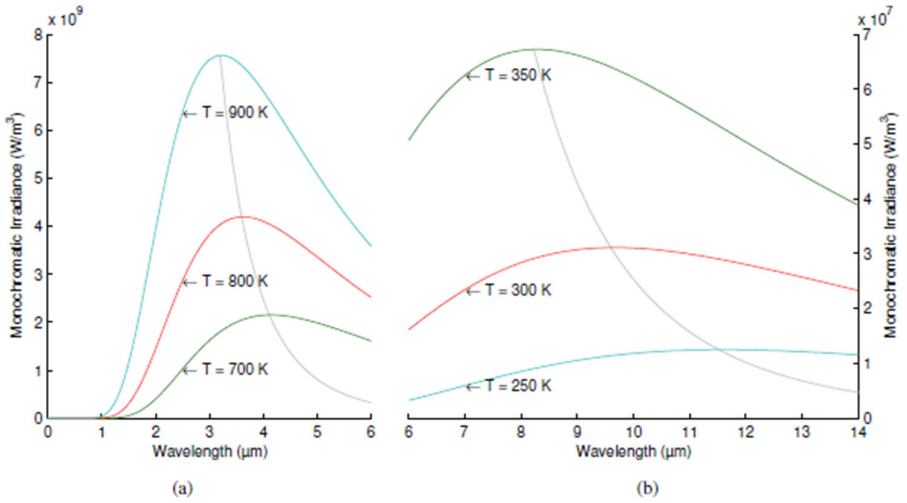


Figura 2-7. Gráfica representativa de la Ley de Planck.
(a) Los objetos con mayor temperatura emiten principalmente en el infrarrojo de onda media.
(b) Los objetos con menor temperatura lo hacen en el infrarrojo de onda larga.
(Extraído de [20])

Esto explica por qué, a medida que una muestra de construcción se calienta, el máximo de su emisión se desplaza hacia longitudes de onda más cortas. Para determinar la potencia total radiada por unidad de área, se aplica la ley de Stefan-Boltzmann (ver Ecuación 3 y Figura 2-8):

$$W_b = \sigma \cdot T^4 \qquad \text{(Ecuación 3)}$$

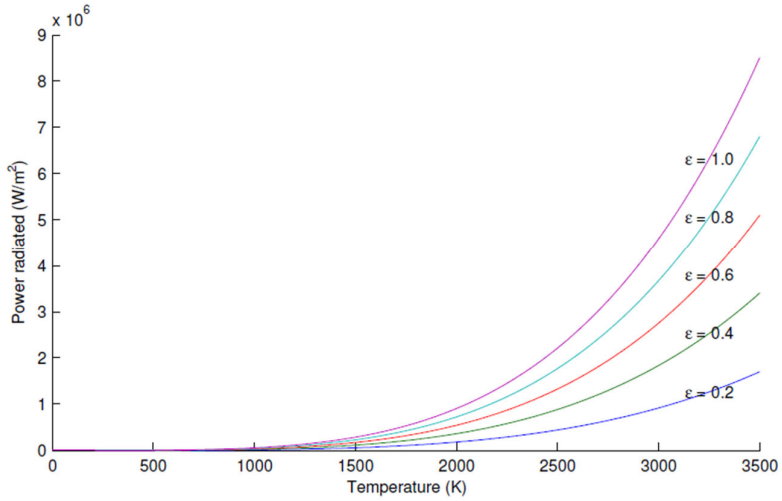


Figura 2-8. Ley de Stefan-Boltzmann (Extraído de [20])

Donde σ es la constante de Stefan-Boltzmann ($5{,}67 \times 10^{-8}\ W/m^2K^4$). Sin embargo, los materiales del mundo real, como el acero AISI o el mortero, no son emisores perfectos y se denominan cuerpos grises. La emisividad (ϵ) es un parámetro crítico definido como la relación entre la energía emitida por un objeto real y la de un cuerpo negro a la misma temperatura. Según la ley de Kirchhoff, para materiales opacos en equilibrio térmico, la emisividad es igual a la absortividad, lo que implica que la suma de la emisividad y la reflectividad (ρ) es igual a la unidad ($\epsilon + \rho = 1$). Es fundamental considerar que la emisividad no es estática, ya que varía sustancialmente con la rugosidad superficial, el ángulo de visión y el estado de oxidación del material [20] [22] [23].

2.3.3 Radiación detectada y modelos de compensación paramétrica

En un escenario de medición experimental, la radiación total captada por el sensor de la cámara (W_{tot}) es una mezcla compleja de tres contribuciones energéticas distintas que deben ser aisladas para determinar la temperatura real de la muestra: la emisión propia del objeto (E_{obj}), la radiación del entorno reflejada por la superficie (E_{refl}) y la radiación emitida por la atmósfera presente entre la cámara y el objetivo (E_{atm}) (ver Figura 2-9) [20].

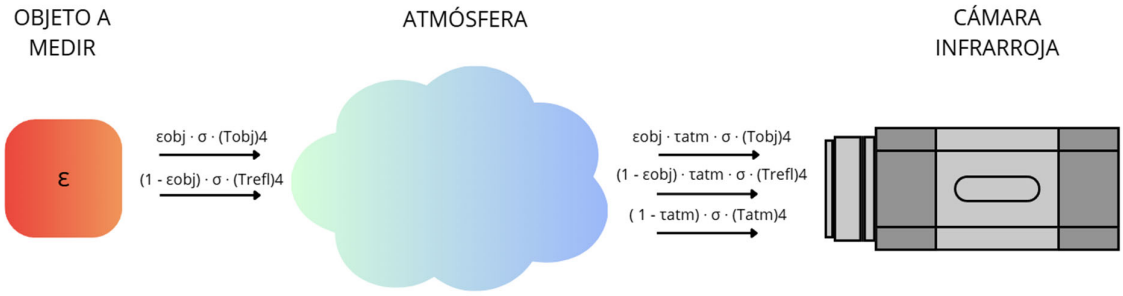


Figura 2-9. Radiación recibida a través de una cámara infrarroja (Elaboración propia con Canva)

La ecuación matemática fundamental para realizar esta compensación es (ver Ecuación 4):

$$W_{tot} = \epsilon \cdot \tau_{atm} \cdot \sigma \cdot T_{obj}^4 + (1-\epsilon) \cdot \tau_{atm} \cdot \sigma \cdot T_{refl}^4 + (1-\tau_{atm}) \cdot \sigma \cdot T_{atm}^4 \quad \text{(Ecuación 4)}$$

En esta expresión, τ_{atm} representa la transmitancia de la atmósfera transmitancia de la atmósfera (inversamente proporcional al aumento de distancia, humedad relativa y temperatura ambiental), T_{refl} es la temperatura ambiental reflejada y T_{atm} es la temperatura del aire. El parámetro con mayor impacto en la incertidumbre es la emisividad (ϵ), especialmente en materiales metálicos o reflectantes, donde errores mínimos en su estimación provocan desviaciones térmicas masivas [22].

2.3.4 Mecanismos de transporte de calor en materiales porosos

El análisis de muestras de construcción mediante ciclos de calentamiento y enfriamiento permite caracterizar su estructura interna a través de los fenómenos de transporte de energía. La transferencia de calor ocurre a través de tres mecanismos que interactúan simultáneamente: conducción, convección y radiación [24].

- **Conducción:** es el transporte interno de calor regido por la ley de Fourier ($q = -k\nabla \mathrm{T}$), donde k es la conductividad térmica (W/(mK)) [24]. En materiales contaminados con sales, la presencia de estas sustancias en la red de poros altera significativamente la densidad y la estructura interna, modificando la difusividad térmica del espécimen. El análisis de la conducción tangencial en la superficie es vital para corregir errores en áreas con altos gradientes térmicos [25].

- **Convección:** describe el intercambio de calor entre la superficie sólida y el fluido circundante, modelado por la ley de enfriamiento de Newton ($Q_c = h(T_s - T_r)$), donde h es el coeficiente de convección ($W/((m^2K)$). Durante el enfriamiento prolongado de una muestra, la temperatura suele seguir un decaimiento exponencial hacia el equilibrio térmico ambiental [24].

- **Radiación:** es el mecanismo de transferencia de energía en forma de ondas electromagnéticas entre cuerpos a distinta temperatura, sin necesidad de un medio material [26].

Para inducir contrastes térmicos que revelen patologías ocultas o la presencia de contaminantes químicos, se utiliza la termografía activa. Una técnica común es el calentamiento de paso (*Step Heating*), donde se aplica un estímulo térmico continuo y se monitoriza la evolución temporal de la temperatura superficial. Los diferentes componentes químicos, debido a su inercia térmica diferenciada, actúan como barreras térmicas que alteran la velocidad de propagación del calor, generando "firmas térmicas" únicas capturadas por la cámara [20].

2.3.5 Cámara termográfica

Las cámaras termográficas (ver Figura 2-10) son dispositivos de medida que permiten determinar la distribución de temperaturas superficiales de un objeto mediante la detección de la radiación infrarroja que este emite en función de su temperatura. Todo cuerpo cuya temperatura es superior al cero absoluto emite energía en forma de radiación electromagnética, cuya intensidad y distribución espectral están descritas por las leyes de la radiación térmica [27].



Figura 2-10. Cámara termográfica FLIR (Elaboración propia)

A diferencia de las cámaras convencionales, que operan en el espectro visible, las cámaras termográficas permiten obtener información térmica sin contacto físico con el objeto analizado. Esta característica las convierte en una técnica especialmente adecuada para el estudio de materiales de construcción, donde resulta fundamental no alterar el estado del material durante el ensayo [27], [28].

El funcionamiento de estos dispositivos se basa en la captación de la radiancia térmica mediante un sensor infrarrojo, generalmente un microbolómetro no refrigerado. Este sensor transforma la energía incidente en una señal eléctrica que posteriormente es procesada por la electrónica interna de la cámara. A partir de esta señal, el equipo estima una temperatura aparente mediante modelos radiométricos que tienen en cuenta parámetros como la emisividad del material, la radiación reflejada del entorno y las condiciones atmosféricas [29].

En aplicaciones relacionadas con materiales de construcción y ensayos a temperatura ambiente, las cámaras operan habitualmente en la banda de infrarrojo de onda larga (LWIR, *Long Wave Infrared*, 8–14 μm), ya que en este intervalo se concentra la mayor parte de la radiación emitida por los objetos a temperaturas moderadas y se minimizan los efectos de absorción atmosférica [26], [27].

Desde el punto de vista metodológico, la termografía infrarroja (IRT, *Infrared Thermography*) se emplea como una técnica de ensayo no destructivo (NDT, *Non-Destructive Testing*) que permite analizar la respuesta térmica de un material ante distintas condiciones de excitación.

En este contexto, se distinguen dos enfoques principales: la termografía infrarroja pasiva y la termografía infrarroja activa (ver Figura 2-11).



Figura 2-11. Termografía activa vs pasiva (Elaboración propia en Canva a partir de imágenes generadas con inteligencia artificial Gemini)

La termografía pasiva se basa en la observación de la radiación térmica emitida de forma natural por los objetos, sin aplicar ningún estímulo externo. Las diferencias de temperatura detectadas se deben a condiciones ambientales, como la radiación solar o los gradientes térmicos naturales, siendo especialmente útil en inspecciones a gran escala o en estructuras donde existen contrastes térmicos espontáneos [27], [28].

Por el contrario, la termografía activa consiste en la aplicación de un estímulo térmico externo, como un calentamiento o enfriamiento controlado, con el objetivo de inducir contrastes térmicos en el material. La respuesta térmica generada depende de las propiedades internas del material, lo que permite identificar heterogeneidades, defectos o variaciones en su composición [27].

La base física de esta metodología radica en los mecanismos de transferencia de calor en el interior de los materiales, fundamentalmente conducción, convección y radiación, que determinan la evolución térmica observada en superficie [24]. En materiales porosos, la presencia de humedad y sales solubles modifica tanto el transporte de calor como el transporte de masa, alterando su comportamiento térmico global [25].

Como consecuencia, las zonas afectadas presentan una respuesta diferente frente a un estímulo térmico, lo que se traduce en variaciones detectables en los termogramas. Este fenómeno constituye el fundamento de la aplicación de la termografía infrarroja al estudio de patologías en materiales de construcción.

Las cámaras termográficas permiten registrar secuencias temporales de temperatura superficial, en las que cada píxel contiene información dependiente del tiempo. Estos datos pueden exportarse en formato (*CSV*, *Comma-Separated Values*), lo que facilita su tratamiento posterior mediante herramientas de análisis numérico y la obtención de curvas de evolución térmica.

El análisis de dichas curvas permite caracterizar la cinética térmica de los materiales, directamente relacionada con sus propiedades internas. La comparación entre diferentes materiales o estados de alteración facilita la identificación de comportamientos diferenciales asociados a la presencia de humedad o sales.

Para el tratamiento de las secuencias térmicas, pueden emplearse técnicas como el análisis de componentes principales (PCA, Principal Component Analysis), que permite reducir la complejidad de

los datos y resaltar los patrones térmicos más relevantes [30]. Asimismo, el procesamiento avanzado de señales termográficas contribuye a mejorar la calidad de la información y a minimizar el ruido asociado al sistema de adquisición [20], [31].

En este contexto, la termografía infrarroja se configura como una herramienta de análisis que permite establecer relaciones entre la respuesta térmica superficial y las propiedades internas del material, siendo especialmente útil en la detección y caracterización de procesos de degradación en materiales de construcción.

2.4 Fundamentos de la Inteligencia Artificial y *Deep Learning*

La Inteligencia Artificial (IA) se ha consolidado como una de las disciplinas más relevantes dentro del ámbito científico y tecnológico, cuyo objetivo fundamental es el desarrollo de sistemas capaces de emular procesos cognitivos humanos, tales como el aprendizaje, la percepción o la toma de decisiones. Dentro de este campo, el aprendizaje automático (*machine learning*) ha adquirido una importancia central, al permitir la construcción de modelos que mejoran su rendimiento a partir de los datos, sin necesidad de programación explícita para cada tarea [32], [33]

El *machine learning* se basa en el desarrollo de algoritmos capaces de identificar patrones en los datos mediante técnicas estadísticas y matemáticas. Tradicionalmente, estos modelos requieren un proceso de ingeniería de características, en el que el experto define manualmente qué variables o representaciones son relevantes para el problema. Ejemplos de estos modelos incluyen máquinas de soporte vectorial, árboles de decisión o regresiones lineales.

En contraste, el *deep learning* puede entenderse como una subárea del *machine learning* que automatiza este proceso mediante el uso de redes neuronales profundas. Estas redes son capaces de aprender directamente a partir de los datos sin necesidad de una extracción manual de características, construyendo representaciones jerárquicas en las que las capas iniciales detectan patrones simples, mientras que las capas profundas capturan estructuras complejas. Esta diferencia constituye uno de los factores clave que han impulsado el éxito del *deep learning* en tareas complejas como el reconocimiento de imágenes, el procesamiento del lenguaje natural o el análisis de señales [34].

El proceso de aprendizaje en una red neuronal profunda puede entenderse como un mecanismo iterativo cuyo objetivo es minimizar una función de pérdida que cuantifica la diferencia entre las predicciones del modelo y los valores reales. Tal y como se muestra en la Figura 2-12, este proceso implica un flujo bidireccional de información: una fase de propagación hacia adelante, en la que se obtiene una predicción, y una fase de propagación hacia adelante (*forward propagartion*), en la que se obtiene una predicción, y una fase de retropropagación (*backforward propagation*), en la que el error se utiliza para ajustar los parámetros del modelo, es decir, sus pesos (*weights*) [35].

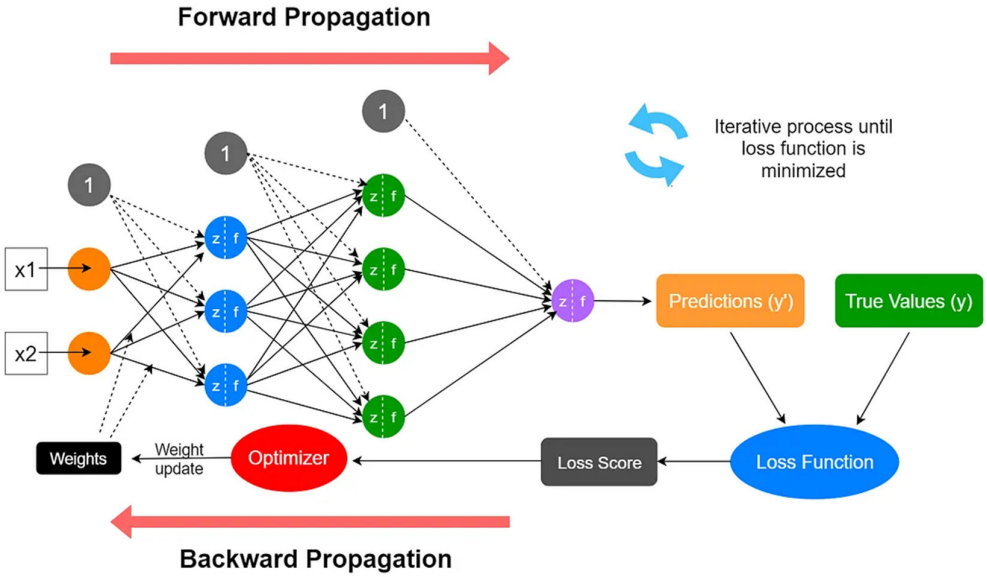


Figura 2-12. Proceso general de aprendizaje de una red neuronal (Extraído de [35])

El funcionamiento interno de una red neuronal se basa en la combinación de transformaciones lineales y no lineales. Siguiendo la formulación clásica de redes neuronales profundas descrita por Goodfellow, la salida intermedia de una capa se obtiene mediante la ecuación de propagación hacia adelante (ver Ecuaciones 6 y 7) [34]:

$$z^{(l)} = W^{(l)}a^{(l-1)} + b^{(l)} \quad \text{(Ecuación 6)}$$

$$a^{(l)} = \sigma(z^{(l)}) \quad \text{(Ecuación 7)}$$

donde $W^{(l)}$representa la matriz de pesos, $b^{(l)}$el vector de sesgos, $a^{(l-1)}$las activaciones de la capa anterior y σla función de activación. En general, estas magnitudes son adimensionales, ya que en la mayoría de aplicaciones de *deep learning* los datos han sido previamente normalizados o escalados [34].

La variable intermedia $z^{(l)}$, resultado de la combinación lineal de estas magnitudes, es igualmente adimensional. No obstante, cuando las variables de entrada $x$poseen unidades físicas (por ejemplo, temperatura o intensidad térmica), dichas unidades pueden considerarse implícitamente en las primeras etapas del procesamiento, aunque pierden su interpretación directa como consecuencia de las transformaciones no lineales que tienen lugar en la red.

A medida que la información atraviesa la red, las representaciones internas se vuelven progresivamente más abstractas hasta alcanzar la capa de salida, donde se obtiene una predicción $\hat{y}$, que presenta las mismas unidades que la variable objetivo y, en caso de que estas existan, lo que se denomina como aprendizaje supervisado. Esta predicción se compara con el valor real mediante una función de pérdida $L(y, \hat{y})$, que en el caso del error cuadrático medio se expresa como:

$$L = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \quad \text{(Ecuación 8)}$$

El valor de esta función permite cuantificar el error del modelo y constituye la base del proceso de aprendizaje. A partir de él, se inicia la fase de retropropagación, en la que se calculan los gradientes de la función de pérdida respecto a los pesos del modelo mediante la regla de la cadena[34]:

$$\frac{\partial L}{\partial W^{(l)}} \quad \text{(Ecuación 9)}$$

Estos gradientes se emplean para actualizar los pesos mediante algoritmos de optimización, como el descenso por gradiente:

$$W^{(l)} = W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}} \quad \text{(Ecuación 10)}$$

donde η es la tasa de aprendizaje, considerada también una magnitud adimensional. Este proceso se repite de manera iterativa durante múltiples épocas (*epochs*), permitiendo que el modelo converja progresivamente hacia una solución óptima [34].

2.4.1 Modelos de Deep Learning para datos secuenciales

En muchos problemas reales, los datos no son independientes, sino que presentan una estructura secuencial en la que el orden es relevante. Este tipo de situaciones requiere modelos capaces de capturar dependencias temporales, lo que dio lugar al desarrollo de las redes neuronales recurrentes (RNN, *Recurrent Neural Network*) [36].

Las RNN (*Recurrent Neural Network*) introducen un estado interno que actúa como memoria, permitiendo que la información pasada influya en el procesamiento actual [36].

2.4.1.1 Redes Neuronales Recurrentes (RNN)

Las redes neuronales recurrentes representan la formulación más básica de este tipo de arquitecturas. En ellas, el modelo procesa una secuencia de datos de manera iterativa, actualizando en cada instante un estado oculto que sintetiza la información relevante de los pasos anteriores.

Desde un punto de vista formal, el estado oculto se actualiza en función de la entrada actual y del estado previo, tal y como se describe en la formulación clásica de las RNN (*Recurrent Neural Network*) [34]:

$$a^{(t)} = g(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad \text{(Ecuación 11)}$$

$$y^{<t>} = g(W_{ya}a^{<t>} + b_y) \quad \text{(Ecuación 12)}$$

- $x^{(t)}$: vector de entrada en el instante t, que contiene los datos de la secuencia (por ejemplo, valores de temperatura) ([adimensional] o en unidades físicas según los datos de entrada, p. ej., K o °C).
- $a^{(t)}$: estado oculto en el instante t, que actúa como memoria interna de la red (adimensional).
- $a^{(t-1)}$: estado oculto del instante anterior, que introduce la dependencia temporal (adimensional).
- $y^{(t)}$: salida del modelo en el instante t(adimensional o en unidades del problema; por ejemplo, temperatura).
- W_{ax}: matriz de pesos que conecta la entrada con el estado oculto (adimensional).
- W_{aa}: matriz de pesos recurrentes que conecta el estado anterior con el actual (adimensional).
- W_{ya}: matriz de pesos que conecta el estado oculto con la salida (adimensional).
- b_a, b_y: vectores de sesgo (bias), que permiten ajustar la activación del modelo (adimensionales).
- g: función de activación no lineal (por ejemplo, tangente hiperbólica o sigmoide), que introduce no linealidad en el modelo (adimensional).

Estas ecuaciones derivan directamente de la extensión temporal del modelo de red neuronal multicapa, en la que los pesos se comparten a lo largo del tiempo, lo que permite tratar secuencias de longitud variable.

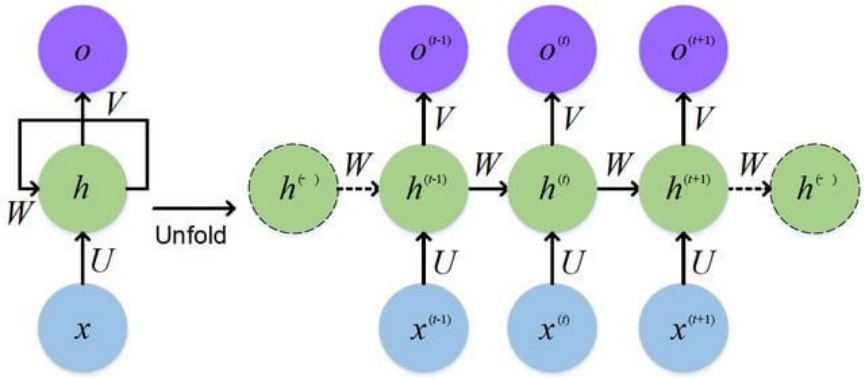


Figura 2-13. Red Neuronal Recurrente desplegada (Extraído de [37])

Desde una perspectiva conceptual, el estado oculto $a^{<t>}$puede interpretarse como una memoria acumulativa, en la que se condensa la información relevante de la secuencia hasta el instante actual. Sin embargo, esta memoria presenta una limitación fundamental: su capacidad para retener información a largo plazo es muy reducida. A medida que la secuencia se hace más extensa, la influencia de los primeros elementos se diluye progresivamente, lo que se traduce en una pérdida de contexto.

Este fenómeno, conocido como problema del gradiente desvaneciente, se produce durante el entrenamiento y dificulta la propagación del error a lo largo de múltiples pasos temporales, limitando la capacidad del modelo para aprender dependencias a largo plazo. Como consecuencia, las RNN (*Recurrent Neural Network*) resultan adecuadas únicamente para secuencias cortas o problemas donde el contexto temporal es limitado [36].

2.4.1.2 Redes LSTM (Long Short-Term Memory)

Con el objetivo de superar las limitaciones de las RNN (*Recurrent Neural Network*) tradicionales, Hochreiter y Schmidhuber propusieron las redes LSTM (*Long Short-Term Memory*), que introducen un

mecanismo de memoria más sofisticado capaz de conservar información durante largos periodos de tiempo [38].

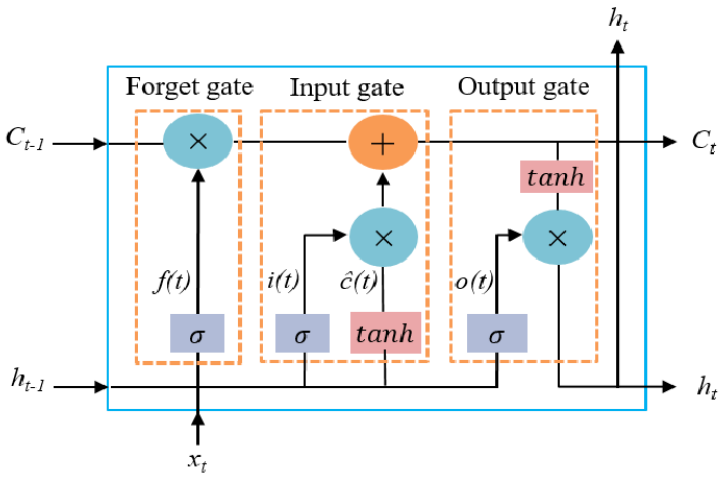


Figura 2-14. Arquitectura de la red LSTM (Extraído de [39])

El elemento central de las LSTM (*Long Short-Term Memory*) es el denominado estado de celda C_t, que actúa como una memoria a largo plazo. A diferencia de las RNN (*Recurrent Neural Network*) simples, este estado no se modifica de forma directa en cada paso, sino que se actualiza mediante un conjunto de mecanismos de control conocidos como compuertas.

La ecuación fundamental que describe la actualización de la memoria es la siguiente [38]:

$$C_t = f_t C_{t-1} + i_t \tilde{C}_t \qquad \text{(Ecuación 13)}$$

- C_t: estado de celda en el instante t, que representa la memoria a largo plazo (adimensional).
- C_{t-1}: estado de celda anterior (adimensional).
- $\tilde{C}_t$: información candidata que se puede añadir a la memoria (adimensional).
- f_t: compuerta de olvido, que determina qué información se conserva (adimensional, valores entre 0 y 1).
- i_t: compuerta de entrada, que controla la incorporación de nueva información (adimensional, entre 0 y 1).

Esta formulación se basa en la combinación de dos procesos: por un lado, la retención de información previa mediante la compuerta de olvido f_t, y por otro, la incorporación de nueva información a través de la compuerta de entrada i_t.

Desde un punto de vista intuitivo, este mecanismo permite que la red decida de manera explícita qué información debe mantenerse y cuál debe descartarse. La compuerta de olvido elimina información irrelevante, la compuerta de entrada añade nueva información significativa y la compuerta de salida regula qué parte de la memoria se utiliza en cada instante.

Este diseño resulta especialmente eficaz porque evita que la información relevante se degrade con el tiempo, permitiendo que el modelo capture dependencias a largo plazo. En términos conceptuales, las LSTM pueden interpretarse como un sistema de memoria estructurada, en el que la información no solo se almacena, sino que se gestiona de manera activa.

2.4.1.3 Redes GRU (Gated Recurrent Unit)

Las GRU (*Gated Recurrent Unit*) representan una evolución posterior de las LSTM (*Long Short-Tem Memory*), introducida con el objetivo de simplificar su arquitectura manteniendo un rendimiento comparable [40].

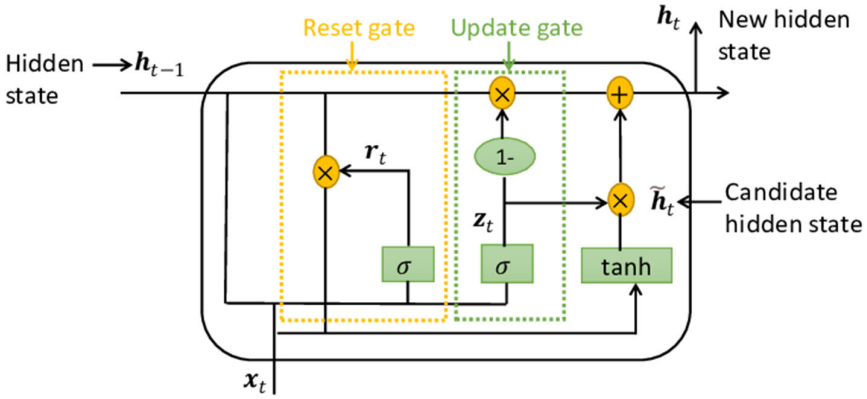


Figura 2-15. Estructura de la red GRU (Extraído de [41])

En este modelo, el estado oculto se actualiza mediante la siguiente expresión [40]:

$$h_t = (1 - z_t)h_{t-1} + z_t\tilde{h}_t \qquad \text{(Ecuación 14)}$$

- h_t: estado oculto actual (adimensional).
- h_{t-1}: estado oculto anterior (adimensional).
- $\tilde{h}_t$: estado candidato (adimensional).
- z_t: compuerta de actualización, que controla cuánto del pasado se mantiene (adimensional, entre 0 y 1).

Esta expresión define cómo se construye el estado oculto en el instante ta partir de una combinación ponderada entre la información pasada y la nueva información generada en el instante actual.

A diferencia de las LSTM (*Long Short-Term Memory*), las GRU (*Gated Recurrent Unit*) combinan el estado de memoria y el estado oculto en una única estructura, eliminando la necesidad de una memoria separada. Además, reducen el número de compuertas a dos: la compuerta de actualización z_t, que determina la cantidad de información pasada que se mantiene, y la compuerta de reinicio, que regula la incorporación de nueva información.

Desde una perspectiva conceptual, las GRU (*Gated Recurrent Unit*) pueden interpretarse como una versión más eficiente de las LSTM (*Long Short-Term Memory*), en la que el control de la memoria se realiza de forma más directa. Esta simplificación reduce el número de parámetros del modelo, lo que se traduce en una mayor eficiencia computacional y, en muchos casos, en un aprendizaje más rápido.

2.4.1.4 Comparación entre modelos

El desarrollo de las arquitecturas recurrentes puede entenderse como una evolución progresiva en la forma en que los modelos de *deep learning* gestionan la información temporal y, en particular, la memoria. Cada una de las arquitecturas analizadas, RNN, LSTM y GRU, responde a una misma necesidad fundamental: incorporar el contexto pasado en el procesamiento presente. Sin embargo, difieren significativamente en la forma en que abordan este objetivo.

Las redes recurrentes simples (RNN) representan el punto de partida de esta evolución. Su planteamiento es conceptualmente sencillo: mantener un estado interno que se actualiza de manera iterativa a medida que se procesa la secuencia. Esta simplicidad constituye, al mismo tiempo, su principal ventaja y su mayor limitación. Por un lado, permite un modelo fácil de implementar y con bajo coste computacional; por otro, carece de mecanismos que regulen el flujo de información, lo que provoca que la memoria se degrade con rapidez. En consecuencia, las RNN resultan adecuadas únicamente en situaciones donde las dependencias temporales son cortas y el contexto necesario es limitado.

Con el fin de superar estas limitaciones, surgen las redes LSTM, que introducen una concepción más sofisticada de la memoria. A diferencia de las RNN, las LSTM no se limitan a acumular información, sino que incorporan un sistema de control basado en compuertas que permite decidir de forma explícita qué información debe conservarse, cuál debe descartarse y qué parte debe utilizarse en cada instante. Este mecanismo dota al modelo de una gran capacidad para gestionar dependencias a largo plazo, evitando la degradación progresiva de la información. No obstante, esta mejora en capacidad de modelado implica también un incremento en la complejidad del modelo, tanto en términos de número de parámetros como de coste computacional.

En este contexto, las GRU pueden interpretarse como una solución intermedia que busca equilibrar capacidad y eficiencia. Estas arquitecturas conservan la idea fundamental de las LSTM , el control del flujo de información mediante compuertas, pero simplifican su estructura eliminando la separación entre memoria y estado oculto. Como resultado, las GRU presentan una menor complejidad estructural, lo que se traduce en tiempos de entrenamiento más reducidos y una mayor eficiencia computacional. A pesar de esta simplificación, en muchos casos logran un rendimiento comparable al de las LSTM, lo que las convierte en una alternativa especialmente atractiva en aplicaciones prácticas.

En conjunto, estas tres arquitecturas reflejan un compromiso progresivo entre simplicidad, capacidad de memoria y coste computacional. Mientras que las RNN destacan por su sencillez pero presentan limitaciones en el manejo de dependencias largas, las LSTM ofrecen una solución robusta a este problema a costa de una mayor complejidad. Por su parte, las GRU representan un equilibrio entre ambas, proporcionando una buena capacidad de modelado con un menor coste. La elección entre uno u otro modelo dependerá, en última instancia, de las características específicas del problema, como la longitud de las secuencias, la complejidad de las dependencias temporales y los recursos computacionales disponibles.

Modelo	Característica principal	Limitación
RNN	Memoria básica acumulativa	No captura largo plazo
LSTM	Memoria controlada mediante compuertas	Mayor complejidad
GRU	Memoria simplificada y eficiente	Menos expresividad

Tabla 2-1. Resumen comparativo entre los modelos recurrentes (Elaboración propia)

3 DESARROLLO DEL TFG

Para el desarrollo de este trabajo se han seleccionado un conjunto de materiales representativos con el fin de evaluar su comportamiento térmico, tras los ensayos de contaminación con distintas sales, mediante los modelos de Deep Learning (DL) seleccionados en las imágenes térmicas adquiridas con la cámara termográfica. De forma cronológica, se detalla a continuación en los siguientes apartados cada uno de las fases.

3.1 Selección y preparación de materiales

Se han empleado seis materiales diferentes: dos variedades de granito, una caliza y dos tipos de arenisca.

Con el objetivo de garantizar la correcta ejecución del estudio experimental, todas las muestras fueron previamente seccionadas mediante una cortadora hasta obtener piezas con dimensiones uniformes (5 cm x 5 cm x 5 cm) . Posteriormente, las muestras se sometieron a un proceso de acondicionamiento consistente en varios ciclos de lavado con agua destilada seguido de un secado controlado en horno (ver Figura 3-1). Este procedimiento se realizó con la finalidad de eliminar posibles sales solubles o contaminantes presentes en los materiales, asegurando así que todas las muestras partieran de unas condiciones iniciales homogéneas antes del inicio de los ensayos experimentales.

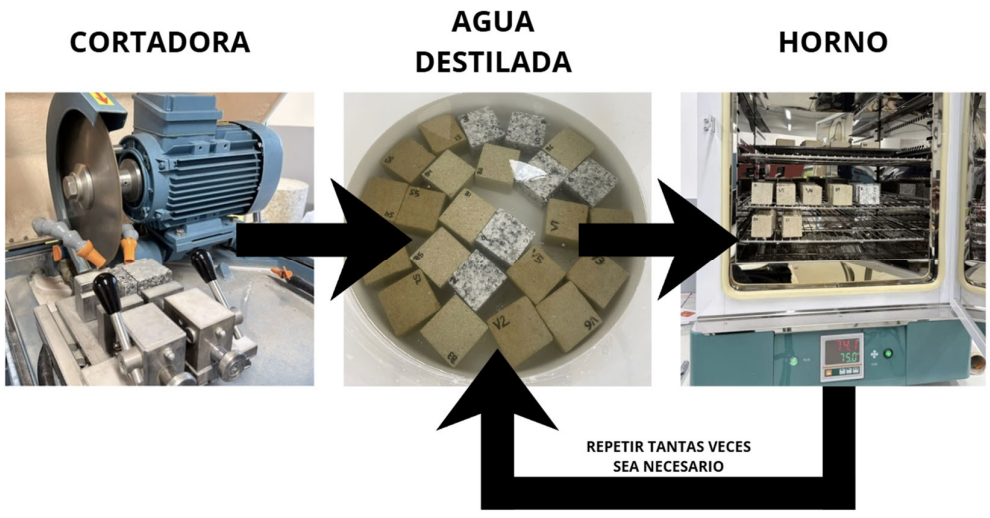


Figura 3-1. Proceso de dimensionado y descontaminación de las muestras previo al estudio (Elaboración propia)

En los apartados siguientes se describen de forma detallada las características de cada uno de los materiales utilizados en el estudio.

3.1.1 Caracterización de los granitos utilizados

3.1.1.1 Granito de Madrid

Este tipo de material corresponde a una variedad de roca granítica ampliamente presente en el Sistema Central, especialmente en la Sierra de Guadarrama (ver Figura 3-2), que se extiende por las comunidades de Madrid y Castilla y León. Este material ha sido utilizado históricamente como piedra de construcción en numerosos edificios y obras civiles de la región, debido a su abundancia, resistencia y durabilidad [42].



Figura 3-2. Localización de extracción de la muestra de Granito de Madrid (Elaboración propia con canva e inteligencia artificial Gemini)

Desde el punto de vista geológico, se trata de una roca ígnea plutónica formada por la solidificación lenta del magma en profundidad durante el Paleozoico, lo que da lugar a una textura cristalina bien desarrollada. El granito de Madrid presenta generalmente una textura de grano medio a grueso, con cristales visibles a simple vista [43].

En cuanto a su composición mineralógica, está constituido principalmente por cuarzo, feldespatos (tanto feldespato potásico como plagioclasa) y micas, fundamentalmente biotita y, en menor proporción, moscovita. La combinación de estos minerales confiere a la roca una tonalidad generalmente gris clara a gris ligeramente rosada, con una estructura granular relativamente homogénea [42].

3.1.1.2 Granito Azul Alpendorada

Este granito es otra variedad granítica ampliamente explotada en el norte de Portugal, especialmente en la zona de Alpendorada, situada en el municipio de Marco de Canaveses, dentro del distinto de Oporto (ver Figura 3-3). Este material es conocido por su tonalidad característica y por su uso frecuente en arquitectura y obra pública [44].

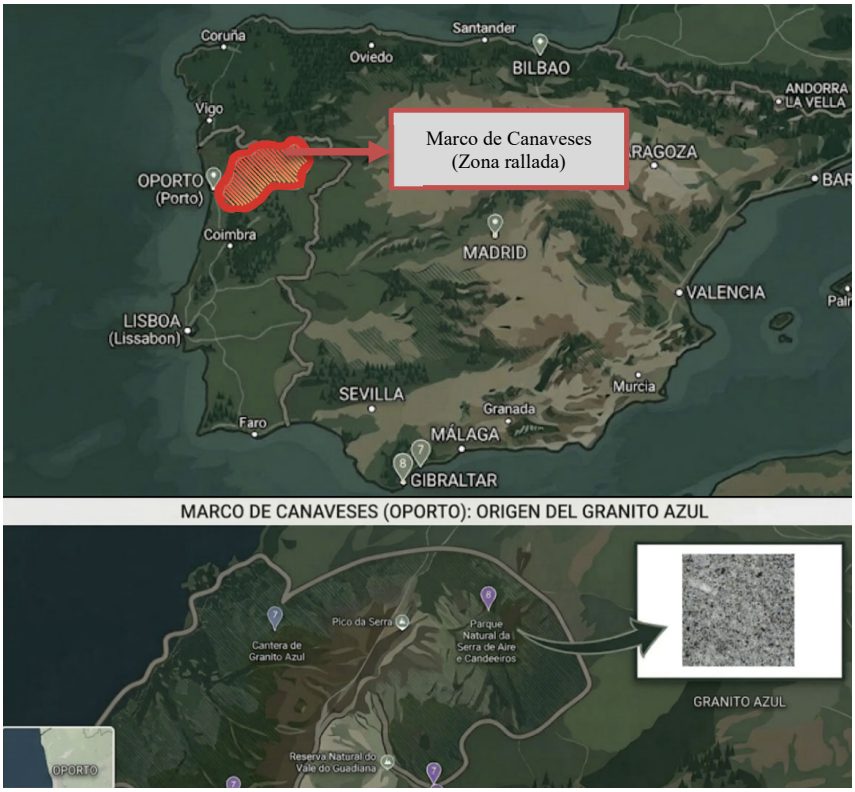


Figura 3-3. Localización de extracción de la muestra de Granito Azul Alpendorada (Elaboración propia con canva e inteligencia artificial Gemini)

En comparación con otros granitos utilizados en construcción, este material puede presentar una mayor heterogeneidad textural, lo que puede influir en la distribución de poros y microfisuras dentro de la roca. Estas características pueden afectar tanto a sus propiedades mecánicas como a su comportamiento frente a procesos de alteración.

3.1.2 Roca sedimentaria carbonatada (caliza)

3.1.2.1 Caliza Bateig

La piedra Bateig es una roca sedimentaria carbonatada ampliamente utilizada en el sureste de España, especialmente en la provincia de Alicante (ver Figura 3-4), donde ha sido empleada de forma tradicional en arquitectura y patrimonio.



Figura 3-4. Lugar de extracción de la Caliza Bateig (Elaboración propia con canva e inteligencia artificial Gemini)

Las propiedades de esta piedra están fuertemente condicionadas por su composición mineralógica, su textura y su estructura porosa, factores que determinan su comportamiento frente a los agentes de deterioro [45].

En particular, la caliza Bateig presenta una significativa capacidad de absorción y desorción de agua, asociada a su sistema poroso, lo que influye directamente en su resistencia frente a procesos de alteración como la cristalización de sales o los ciclos de humedad-sequedad [45].

3.1.3 Rocas sedimentarias siliciclásticas (areniscas)

3.1.3.1 Arenisca Dorada Urbión

La arenisca Dorada Urbión es una roca sedimentaria detrítica utilizada como material de construcción en el norte de España (ver Figura 3-5), caracterizada por su composición mayoritariamente silícea y su estructura granular.



Figura 3-5. Localización de extracción de la muestra de Arenisca Dorada Urbion (Elaboración propia con canva e inteligencia artificial Gemini)

Las areniscas empleadas en construcción presentan una gran variabilidad composicional y textural, siendo su comportamiento físico y mecánico dependiente de factores como la naturaleza del cemento, la granulometría y la conectividad del sistema poroso [46].

Estas rocas suelen presentar una porosidad significativa y una elevada permeabilidad, lo que facilita la circulación de agua y la cristalización de sales en su interior, condicionando su durabilidad frente a agentes ambientales [46].

En el contexto de este estudio, la arenisca Dorada Urbión se incorpora como ejemplo de roca siliciclástica con porosidad intergranular, permitiendo evaluar la influencia de la humedad y las sales.

3.1.3.2 Arenisca Sierra de la Demanda

La arenisca de la Sierra de la Demanda es una roca sedimentaria detrítica ampliamente utilizada como material de construcción en el norte de España, especialmente en la provincia de Burgos (ver Figura 3-6). Este tipo de piedra destaca por su disponibilidad, su facilidad de extracción y su valor estético, lo que ha favorecido su empleo en arquitectura y patrimonio construido [46].



Figura 3-6. Localización de extracción de la muestra de Arenisca de Sierra de la Demanda (Elaboración propia con canva e inteligencia artificial Gemini)

Desde el punto de vista petrográfico, se trata de una arenisca compuesta mayoritariamente por cuarzo, lo que le confiere una elevada dureza y resistencia. Presenta además una estructura granular con una red de poros bien desarrollada, lo que determina en gran medida su comportamiento físico y mecánico [46].

En cuanto a sus propiedades petrofísicas, esta roca se caracteriza por ser porosa y permeable, con un sistema poroso que controla tanto la circulación de fluidos como su respuesta frente a agentes de alteración. La presencia de agua y sales puede modificar la geometría y conectividad de los poros, afectando a propiedades como la absorción, la permeabilidad y la anisotropía del material [46].

3.1.3.3 Arenisca Villamayor

La arenisca de Villamayor es una roca sedimentaria ampliamente utilizada como material de construcción en la ciudad de Salamanca y su entorno (ver Figura 3-7), donde constituye el principal material del patrimonio arquitectónico.



Figura 3-7. Localización de extracción de la muestra de Arenisca Villamayor (Elaboración propia con canva e inteligencia artificial Gemini)

Desde el punto de vista litológico, se trata de una arenisca eocena de grano fino a muy fino, compuesta principalmente por cuarzo, feldespatos y minerales arcillosos que actúan como cemento [47].

Las propiedades físicas de esta piedra están fuertemente condicionadas por su elevada porosidad, su capacidad de absorción de agua y su comportamiento capilar, lo que la hace especialmente susceptible a procesos de alteración relacionados con la humedad y la cristalización de sales [48].

3.1.4 Identificación de las muestras

Todas las muestras han sido identificadas mediante un sistema de codificación ubicado en la cara superior, concretamente en la esquina superior izquierda de cada probeta. Este sistema se compone de una letra, con la excepción del Granito de Madrid, que carece de ella, que indica el tipo de material, y un número que corresponde al tipo de sal aplicada en cada caso.

La relación completa entre la codificación empleada y las características de las muestras se presenta en la Tabla 3-1:

Material/Sal	**KCl (1)**	**Na_2SO_4 (2)**	**Na_2SO_4 (3)**	**Na_2SO_4 (4)**	**NaCl (5)**	**Sin Sal (6)**
Granito de Madrid (sin letra)	1	2	3	4	5	6
Granito Azul (A)	A1	A2	A3	A4	A5	A6
Caliza Bateig (B)	B1	B2	B3	B4	B5	B6
Arenisca Dorada Urbión (D)	D1	D2	D3	D4	D5	D6
Arenisca Sierra de la Demanda (S)	S1	S2	S3	S4	S5	S6
Arenisca Villanueva (V)	V1	V2	V3	V4	V5	V6

Tabla 3-1. Referencia de las muestras con el tipo material y sal empleados (Elaboración propia)

Asimismo, tal y como se refleja en dicha tabla, las muestras identificadas con el número 6 se han considerado como muestras de referencia, dado que no han sido sometidas a ningún tratamiento con sales (véase Figura 3-8).

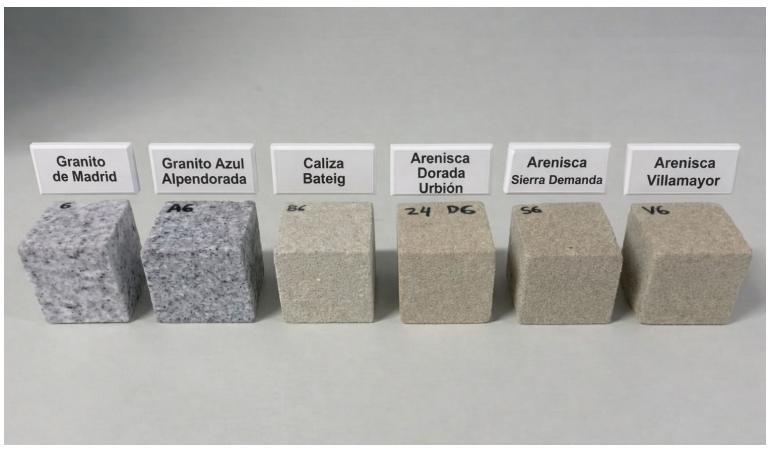


Figura 3-8. Muestras de referencia con su respectiva identificación de material (Elaboración propia)

3.1.5 Sales solubles seleccionadas

Para la realización del proceso de contaminación salina de las muestras se prepararon diferentes disoluciones acuosas utilizando agua destilada como disolvente. Las sales seleccionadas son compuestos frecuentemente asociados a procesos de deterioro en materiales de construcción, especialmente en rocas naturales y morteros porosos. La presencia de sales solubles constituye uno de los principales mecanismos de degradación en materiales pétreos, ya que la cristalización de estas sales en el interior de los poros puede generar tensiones internas capaces de provocar fisuración, descamación y pérdida de material [49] [50].

Cada disolución se preparó en botellas de 1 litro de capacidad, en las que se introdujo la cantidad de sal correspondiente según el porcentaje en masa establecido. Para las disoluciones al 14 % en peso se emplearon 140 g de sal, mientras que para la disolución al 20 % en peso se utilizaron 200 g de sal, completándose el volumen restante con agua destilada hasta alcanzar el litro total de disolución (ver Figura 3-9). Estas concentraciones fueron seleccionadas para garantizar la solubilidad de la sal en la disolución en todo momento, sin precipitación.

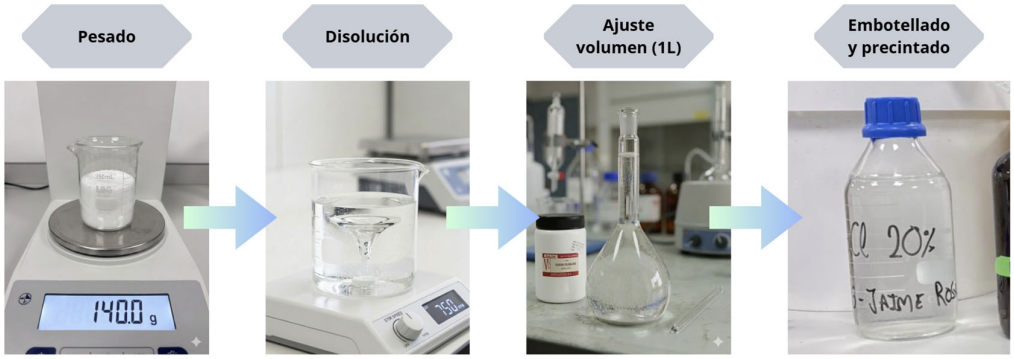


Figura 3-9. Proceso de elaboración de las disoluciones homogéneas de sales solubles (Elaboración propia)

3.1.5.1 Cloruro de potasio (KCl) – 14 % en peso

El cloruro de potasio es una sal altamente soluble en agua que puede encontrarse en determinados ambientes naturales y en suelos con presencia de sales solubles. Su elevada solubilidad facilita su transporte a través de la red porosa de los materiales cuando existe presencia de humedad. Durante los procesos de evaporación del agua, la sal puede cristalizar en el interior de los poros, generando tensiones internas que contribuyen al deterioro del material [49].

3.1.5.2 Sulfato de sodio (Na_2SO_4) – 14 % en peso

El sulfato de sodio es una de las sales más estudiadas en procesos de deterioro de materiales pétreos. Su relevancia se debe a su capacidad para experimentar transiciones entre fases hidratadas y anhidras, especialmente entre thenardita (Na_2SO_4) y mirabilita ($Na_2SO_4 \cdot 10H_2O$). Estos cambios de fase implican importantes variaciones de volumen que pueden generar elevadas presiones de cristalización en el interior de los poros, provocando daños significativos en materiales porosos [51] [52].

3.1.5.3 Nitrato de sodio ($NaNO_3$) – 14 % en peso

El nitrato de sodio es una sal muy soluble que presenta una elevada movilidad en solución acuosa. Este compuesto puede encontrarse en ambientes contaminados o en zonas donde existe infiltración de agua con contenido en nitratos. Debido a su carácter higroscópico, puede modificar las condiciones de

humedad en los materiales porosos, influyendo en los procesos de absorción y evaporación del agua [53].

3.1.5.4 Sulfato de magnesio ($MgSO_4$) – 14 % en peso

El sulfato de magnesio puede formar diferentes fases hidratadas durante los ciclos de hidratación y deshidratación. Estas transformaciones generan variaciones de volumen que pueden provocar tensiones internas dentro del material, contribuyendo a su degradación progresiva, especialmente en materiales con mayor porosidad [50] [52].

3.1.5.5 Cloruro de sodio (NaCl) – 20 % en peso

El cloruro de sodio es una de las sales más comunes en ambientes marinos o en zonas donde se utilizan sales de deshielo. Debido a su elevada solubilidad y facilidad de transporte en solución acuosa, puede penetrar fácilmente en la estructura porosa de los materiales. Aunque generalmente produce presiones de cristalización menores que algunos sulfatos, su abundancia y presencia frecuente en entornos naturales lo convierten en un agente relevante en procesos de deterioro por sales [49] [54].

El uso de estas diferentes disoluciones permite reproducir en condiciones controladas procesos de contaminación salina similares a los que pueden producirse en materiales de construcción expuestos a ambientes agresivos, facilitando así el análisis de su influencia en el comportamiento térmico de los materiales durante los ensayos realizados mediante termografía infrarroja y su posterior procesamiento de datos con modelos de *Deep Learning*.

3.1.6 Enumeración de las disoluciones salinas

Una vez preparadas las disoluciones conforme al procedimiento descrito, se procede a su correcta numeración, tal y como se muestra en la Figura 3-10 y se ha mencionado anteriormente. Este proceso permite asignar un identificador único a cada disolución, facilitando su reconocimiento y manejo a lo largo del desarrollo del trabajo, así como evitando posibles confusiones durante las distintas fases experimentales.

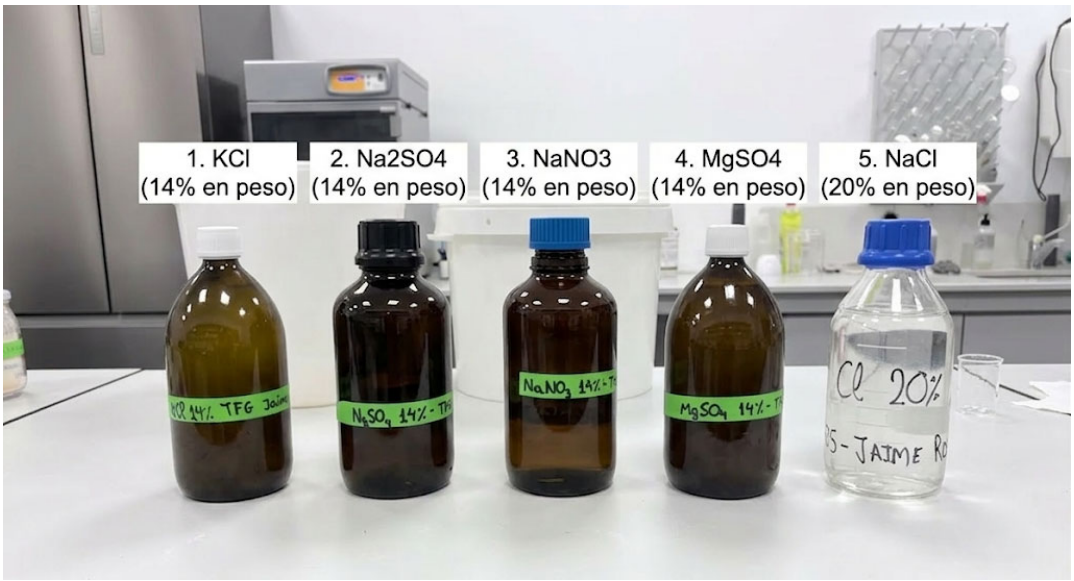


Figura 3-10. Disoluciones de las diferentes sales utilizadas en el proyecto embotelladas y precintadas (Elaboración propia)

3.2 Técnicas de caracterización mecánica

En este apartado se detalla las técnicas de caracterización mecánica utilizadas con el fin de garantizar, como comprobación previa antes de los ensayos termográficos, que las propiedades mecánicas de las piezas han sido modificadas debido a la contaminación de las sales, comprobando entre fase inicial antes de contaminación y fase final después de contaminación.

3.2.1 Ultrasonidos

La prueba de velocidad de transmisión de ultrasonidos se realiza con el objetivo de evaluar las características internas de las muestras pétreas. Debido a su carácter no destructivo, es un método empleado de manera habitual en la caracterización de materiales pétreos y en el estudio de materiales de construcción, ya que permite obtener información relevante sobre el estado interno del material sin alterar su integridad [55] .

Para la realización del ensayo se utiliza un equipo de ultrasonidos compuesto por un generador de impulsos y dos transductores piezoeléctricos, uno emisor y otro receptor. El procedimiento consiste en generar un pulso ultrasónico que atraviesa la muestra desde el transductor emisor hasta el receptor, registrándose el tiempo que tarda la señal en recorrer la distancia entre ambos puntos, y sabiendo la distancia mediante medición en la muestra, calcular la magnitud de la velocidad. Antes de iniciar las mediciones, se preparan las superficies de contacto de las muestras para asegurar una correcta transmisión de la señal ultrasónica. Para ello, se aplica un material acoplante entre los transductores y la superficie de la muestra, con el fin de eliminar posibles capas de aire que puedan dificultar la propagación de la onda. Una vez posicionados los transductores en caras opuestas de la muestra, el equipo genera el pulso ultrasónico y registra automáticamente el tiempo de tránsito de la señal a través del material [56].

Las mediciones se realizan siguiendo una configuración de transmisión directa, en la que los transductores se colocan en lados opuestos de la muestra (ver Figura 3-11), permitiendo que la onda ultrasónica recorra el trayecto más corto y proporcionando así resultados más precisos. Para mejorar la fiabilidad de los resultados, el ensayo se realiza en cada orientación de la muestra; es decir, entre la cara superior y la inferior, entre la cara izquierda y la derecha, y entre la cara frontal y la posterior [57].

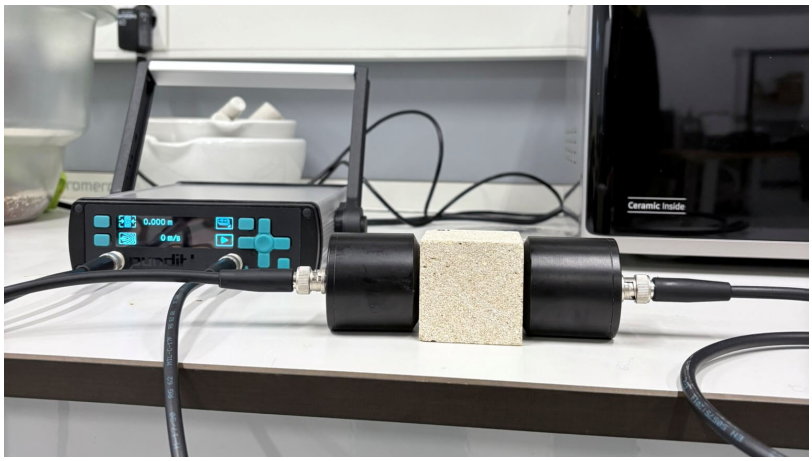


Figura 3-11. Prueba de velocidad de transmisión de ultrasonidos (Elaboración propia)

3.2.2 Dureza superficial

La prueba de dureza superficial se realiza con el objetivo de evaluar la resistencia del material pétreo frente a la penetración o deformación producida por un impacto controlado. Este tipo de ensayo permite obtener una estimación indirecta de la resistencia mecánica del material, así como detectar posibles variaciones en su grado de alteración, compactación o cohesión interna. Debido a su carácter no destructivo y a la rapidez con la que puede realizarse, este método se emplea con frecuencia en la caracterización de rocas utilizadas como materiales de construcción y en estudios de diagnóstico del estado de conservación de materiales pétreos [58].

Para la realización del ensayo se utiliza un durómetro portátil de rebote, como el que se muestra en la Figura 3-12, que permite medir la dureza superficial mediante el impacto controlado de un percutor sobre la superficie de la muestra. El funcionamiento de este tipo de dispositivos se basa en el principio del rebote del percutor, impulsado por un resorte contra la superficie del material. Tras el impacto, el equipo registra la intensidad del rebote producido, la cual depende de la capacidad del material para absorber o devolver la energía del impacto. Este valor de rebote, expresado como índice de dureza superficial (HLD), se relaciona con la resistencia mecánica del material y puede emplearse para estimar de forma indirecta la resistencia a compresión uniaxial (RCS) de la roca mediante distintas correlaciones empíricas propuestas en estudios de mecánica de rocas.



Figura 3-12. Medición de dureza superficial mediante durómetro (Elaboración propia)

Antes de iniciar las mediciones se comprueba que la superficie de la muestra se encuentra limpia y libre de irregularidades que puedan afectar al resultado del ensayo. Posteriormente, el dispositivo se coloca perpendicularmente sobre la superficie del espécimen pétreo y se acciona el mecanismo de impacto del equipo, registrándose automáticamente el valor de dureza en la pantalla del instrumento. Las mediciones se realizaron en cada una de las caras de la muestra, efectuándose impactos en el centro y en las cuatro esquinas de cada cara. En cada uno de estos puntos se llevaron a cabo cinco mediciones consecutivas, descartándose aquellos valores que presentaban desviaciones significativas y calculándose posteriormente el valor medio de los resultados obtenidos con el fin de mejorar la precisión y representatividad de la dureza superficial del material.

3.3 Contaminación de muestras mediante disoluciones salinas

Con el objetivo de reproducir en condiciones de laboratorio los procesos de cristalización de sales que afectan a los materiales pétreos expuestos a ambientes salinos, se llevó a cabo un procedimiento de contaminación controlada mediante distintas disoluciones salinas. Este proceso se aplicó al conjunto de muestras constituido por dos granitos, una caliza y tres areniscas, previamente identificadas y enumeradas mediante letras.

Para el desarrollo del experimento se han preparado cinco bandejas con diferentes disoluciones salinas, cuya composición y concentración se describen en el apartado, relativo a las sales solubles seleccionadas. Cada una de estas disoluciones se utilizó para contaminar una pieza de cada material, identificadas con el número correspondiente a la disolución. Las muestras identificadas con el número 6 permanecen sin contaminar para mantenerlas como muestras de referencia.

El proceso de contaminación se ha llevado a cabo mediante la realización de ocho ciclos consecutivos de absorción de disolución salina y secado, con el objetivo de favorecer la penetración de las soluciones en la red porosa del material y la posterior cristalización de las sales en su interior [59] [60].

Cada ciclo ha constado de dos fases diferenciadas con una duración de dos horas. En primer lugar, las muestras se han colocado sobre una malla plástica situada en el interior de las bandejas que contenían la disolución salina, manteniendo aproximadamente un tercio de la altura de cada muestra sumergido en la solución. La utilización de esta malla permite evitar el contacto directo de las muestras con el fondo de la bandeja y facilitar que la disolución pueda ser absorbida también por la cara inferior de las muestras, además de por sus caras laterales. De este modo, la solución salina asciende a través de la red porosa del material por capilaridad, favoreciendo una impregnación más homogénea de las muestras. Posteriormente, las muestras se han introducido en un horno a 75 °C, con el fin de provocar la evaporación del agua y favorecer la precipitación y cristalización de las sales en el interior del material. Este proceso de secado promueve la acumulación progresiva de sales en los poros, reproduciendo de manera acelerada los mecanismos de alteración que se producen en condiciones naturales.

3.4 Termografía infrarroja

3.4.1 Grabación térmica de los ciclos de enfriamiento y calentamiento de las muestras

En el presente apartado se aborda el análisis mediante termografía infrarroja, técnica no destructiva que permite evaluar el comportamiento térmico de los materiales a partir de la evolución de su temperatura superficial en el tiempo. Para ello, se ha empleado una cámara termográfica modelo FLIR E85, la cual ofrece una alta resolución térmica y una adecuada sensibilidad para detectar variaciones de temperatura en las muestras estudiadas.

El procedimiento experimental ha consistido en la realización de grabaciones continuas de todas las muestras durante un intervalo de dos horas (ver Figura 3-13), utilizando el software específico FLIR Tools, mediante el cual se ha configurado la adquisición de datos a una frecuencia de 10 fotogramas por segundo. Esta configuración ha permitido obtener un registro detallado y preciso de la evolución térmica de cada una de las muestras a lo largo del tiempo.

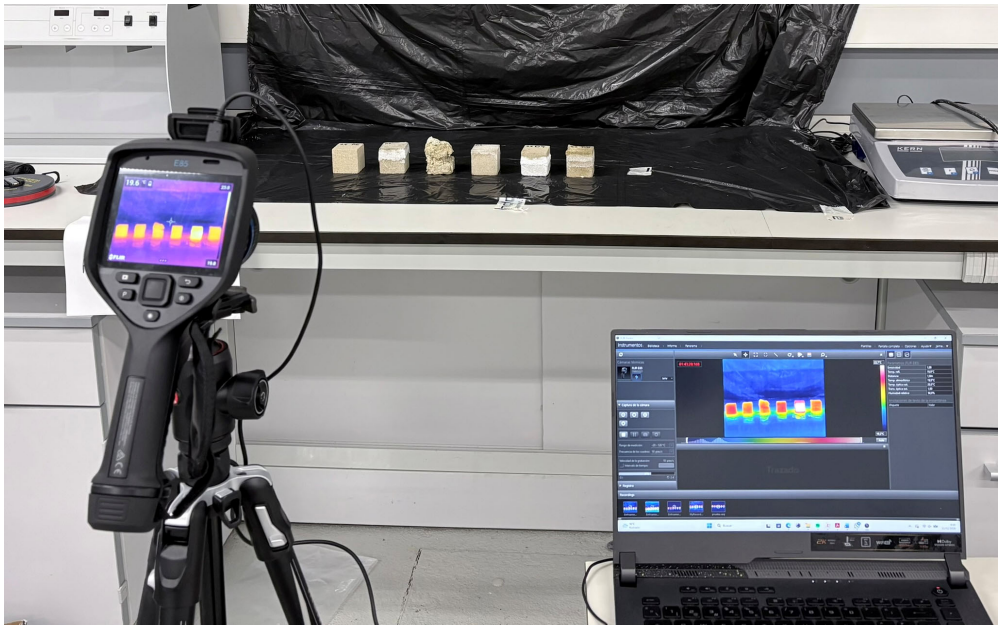


Figura 3-13. Grabación térmica de las muestras con la cámara FLIR E85 (Elaboración propia)

Como resultado de este proceso, se han generado vídeos térmicos correspondientes a los ciclos de enfriamiento y calentamiento de las muestras. En primer lugar, se han registrado los procesos de enfriamiento, para lo cual las muestras fueron previamente introducidas en una estufa a una temperatura de 75 °C y posteriormente extraídas para su enfriamiento progresivo hasta alcanzar la temperatura ambiente. De manera análoga, se han obtenido los vídeos correspondientes al calentamiento de las muestras, que en este caso fueron previamente enfriadas en una nevera a una temperatura aproximada de 5 °C, registrándose su evolución térmica hasta alcanzar nuevamente el equilibrio con el ambiente.

La disposición de las muestras durante la adquisición de las imágenes termográficas se ha mantenido constante y de forma frontal y misma distancia hacia la cámara, con el objetivo de garantizar la comparabilidad de los resultados. Tal y como se observa en la figura, la muestra situada en el extremo izquierdo corresponde al material de referencia, es decir, sin presencia de sales. A continuación, y siguiendo un orden de izquierda a derecha, se disponen las muestras tratadas con sales, identificadas con los números 1, 2, 3, 4 y 5, respectivamente, por tanto, en cada grabación se corresponde bien a un calentamiento o bien a un enfriamiento de un material determinado. Esta organización facilita la interpretación de los resultados, permitiendo establecer comparaciones directas entre el comportamiento térmico del material sin alteración y aquel sometido a distintos tipos de sales.

El conjunto de vídeos térmicos obtenidos durante los ensayos experimentales ha sido transformado en un formato estructurado de datos con el fin de facilitar su análisis cuantitativo y su posterior utilización en modelos de *Deep Learning*. En concreto, cada secuencia térmica ha sido convertida en archivos en formato *CSV* mediante recortes y exportación de datos térmicos en la aplicación de Flir Tools+, en los que se recoge la evolución temporal de la temperatura a nivel de píxel para cada una de las muestras analizadas. De este modo, se dispone de una representación detallada tanto de los procesos de calentamiento como de enfriamiento, permitiendo caracterizar el comportamiento térmico de los materiales con una alta resolución espacial y temporal. Esta transformación resulta fundamental, ya que convierte la información visual original en un conjunto de datos numéricos estructurados, aptos para su tratamiento estadístico y su posterior modelización mediante técnicas de aprendizaje profundo (ver Figura 3-14).



Figura 3-14. Proceso de exportación de datos numéricos de enfriamiento de una muestra (Elaboración propia)

3.5 Procesamiento, análisis y modelización de datos térmicos mediante *Deep Learning*

Con el fin de maximizar la capacidad de discriminación con los modelos de *deep learning*, se ha llevado a cabo un proceso de preprocesado específico de las secuencias de las imágenes térmicas obtenidas durante los ensayos de calentamiento y enfriamiento. Este tratamiento previo resulta determinante, ya que la calidad y representatividad de los datos de entrada condicionan directamente el rendimiento de los modelos, especialmente en problemas donde las diferencias entre clases son sutiles y están asociadas a variaciones térmicas de pequeña magnitud.

Las secuencias registradas presentan una evolución temporal continua ligada a los mecanismos de transferencia de calor en los materiales, por lo que han sido tratadas como series temporales. Para su análisis, se ha definido una transformación basada en el cálculo de un ratio temporal, entendido como la variación de temperatura entre dos instantes separados por un intervalo Δt, normalizada respecto a dicho intervalo. Esta transformación puede expresarse como (ver Ecuación 15):

$$R(t) = \frac{T(t) - T(t - \Delta t)}{\Delta t} \qquad \text{(Ecuación 15)}$$

donde $T(t)$ representa la temperatura en el instante t y Δt corresponden al intervalo temporal considerado. Este parámetro (Δt) actúa como un factor de escala que permite analizar la evolución térmica a distintas resoluciones temporales. En el presente trabajo se han evaluado valores de $\Delta t = 1, 3, 5$ y 7 minutos, con el objetivo de determinar el intervalo más adecuado para la extracción de características relevantes de la secuencia. El parámetro $T(t)$ en la predicción de materiales va a ser, bien la temperatura media, bien la temperatura más su desviación típica o su temperatura media menos su desviación típica. Debido a la limitación temporal del trabajo en la predicción de sales se ha utilizado

sólo la temperatura media, compensándolo con dos enfoques distintos en la diferencia de temperaturas: con escala real y con escala logarítmica, y usando otro tipo de transformación distinto al ratio de temperaturas que ha consistido en el cálculo del área acumulada, bajo las curvas de sales, mediante la regla del trapecio y según Δt. En ambas predicciones se ha analizado las diferencias de temperatura tanto en términos absolutos como en términos relativos. Esto se ve detallado en la explicación de los diferentes bloques.

Desde un punto de vista físico, la formulación del ratio temporal y las áreas acumuladas constituyen una aproximación discreta de la derivada temporal de la temperatura, lo que permite resaltar la dinámica de los procesos de calentamiento y enfriamiento de las muestras. Dado que el intervalo Δt permanece constante en cada experimento, esta normalización facilita la interpretación y comparación térmica.

El tratamiento de las secuencias se ha adaptado en función del problema de predicción abordado. En el caso de la identificación de materiales, se ha trabajado directamente con las curvas térmicas transformadas mediante el ratio temporal, sin aplicar modificaciones adicionales. Esta decisión se fundamenta en que las diferencias en la respuesta térmica entre materiales (derivadas de propiedades intrínsecas como la conductividad térmica, la porosidad o la inercia térmica) son suficientemente significativas como para permitir su discriminación directa. En consecuencia, el modelo aprende a partir del comportamiento térmico global de cada material, siendo estas diferencias las que contienen la información relevante para la clasificación.

Por el contrario, en el análisis de las sales ha sido necesario introducir un procesado adicional. La respuesta térmica de una muestra contaminada está fuertemente condicionada por el material base, lo que puede enmascarar el efecto específico de las sales. Con el fin de aislar dicha contribución, se ha llevado a cabo la sustracción de la curva térmica del material no contaminado respecto a la correspondiente muestra contaminada, obteniendo así una señal diferencial en la que se elimina la influencia del material y se pone de manifiesto el comportamiento térmico asociado exclusivamente a la presencia de sales.

Asimismo, debido a la limitada disponibilidad de datos experimentales y a la similitud en la respuesta térmica de algunas sales, se ha optado por reformular el problema de clasificación en términos del tipo de anión predominante, en lugar de la identificación directa de cada compuesto. Esta decisión se fundamenta tanto en criterios prácticos como en consideraciones físico-químicas, ya que el anión desempeña un papel determinante en los procesos de alteración de los materiales. En consecuencia, se han definido tres clases de salida: una primera asociada a sales con anión cloruro (KCl y NaCl), una segunda correspondiente a sales con anión sulfato (Na_2SO_4 y $MgSO_4$) y una tercera vinculada al anión nitrato ($NaNO_3$).

El diseño, implementación y configuración de los modelos para la predicción de materiales y sales, se describen de manera detallada en los apartados siguientes. En particular, el apartado 3.5.1 recoge el desarrollo del modelo de predicción de materiales, mientras que el apartado 3.5.2 aborda la predicción de sales, cuyos códigos completos se incluyen en los Anexos IV y V, respectivamente.

3.5.1 Primera etapa. Detección de materiales

En este apartado se describe de manera detallada la funcionalidad de cada uno de los bloques que componen el código desarrollado para la primera fase del estudio, centrada en la detección de materiales, independientemente de la presencia o ausencia de contaminación por disoluciones salinas. Asimismo, a continuación de dicha explicación, se incluye el *prompt* utilizado en Gemini para la generación del código implementado en Google Colab.

BLOQUE 1. Configuración inicial y montaje de Google Drive

Este bloque inicial tiene como finalidad preparar el entorno de trabajo en Google Colab mediante la importación de las librerías necesarias para la interacción con Google Drive. A continuación, se procede al montaje de la unidad de Google Drive dentro del entorno de ejecución, lo que permite acceder a los archivos almacenados en la nube de forma directa.

Como entrada, el sistema requiere la autenticación del usuario en su cuenta de Google Drive. Como resultado, se obtiene el montaje de la unidad en la ruta /content/drive, acompañado de un mensaje de confirmación que indica que el proceso se ha realizado correctamente.

Prompt empleado:
"Configura el entorno e interactúa con Google Drive para montar mi unidad en el entorno de Colab".

BLOQUE 2. Procesamiento de imágenes

El propósito fundamental de este bloque es transformar un archivo *CSV* de datos térmicos, procedente de los recortes realizados con FLIR Tools, en un conjunto de imágenes individuales y en una matriz de datos estructurada para su posterior análisis. Dicho archivo contiene múltiples imágenes apiladas verticalmente, por lo que este bloque automatiza su separación y reconstrucción en instantes temporales independientes.

Para su correcta ejecución, el bloque requiere varias entradas por parte del usuario: la ruta completa del archivo *CSV* en Google Drive, el número de filas (alto) y columnas (ancho) correspondientes a cada imagen individual, los fotogramas por segundo (*FPS*) de la grabación original y el intervalo temporal deseado para la extracción de imágenes. Este último parámetro permite reducir el volumen de datos procesados en función de las necesidades del análisis.

Como salida, el bloque genera dos tipos de resultados organizados en carpetas dentro del mismo directorio de entrada: por un lado, una carpeta con imágenes visuales en formato *JPG* correspondientes a cada instante temporal; por otro, un archivo *CSV* que contiene la matriz de datos procesada, donde cada fila representa un píxel y cada columna un instante temporal, almacenando los valores de temperatura en grados Celsius.

Prompt empleado:
"Genera un bloque de código Python para procesar un archivo *CSV* de datos térmicos. Necesito que el código lea un *CSV* donde las imágenes están apiladas verticalmente, las separe en imágenes individuales basándose en la resolución que proporcionaré (filas y columnas), y luego guarde tanto las imágenes visuales (*JPG*) como una matriz de datos procesada (*CSV*) con el tiempo en las columnas. Asegúrate de que el usuario pueda especificar el *FPS* de la grabación original y un intervalo de tiempo para guardar las imágenes".

BLOQUE 3. Análisis Estadístico y Gráfico de Evolución Temporal

Este bloque tiene como objetivo analizar los datos térmicos previamente procesados en el Bloque 2. A partir de los archivos *CSV* generados, se calculan estadísticas relevantes como la temperatura media y la desviación estándar para cada instante temporal, pudiendo realizarse este análisis sobre múltiples muestras de forma simultánea.

Asimismo, el bloque permite filtrar los datos en un intervalo temporal específico, lo que facilita el estudio de fases concretas del comportamiento térmico. Como salida, se generan representaciones gráficas de la evolución temporal de la temperatura media y su desviación, así como un archivo *CSV*

consolidado con todas las estadísticas, lo que permite comparaciones posteriores entre materiales o condiciones experimentales.

Prompt empleado:
"Necesito un bloque de código para analizar la evolución temporal de datos térmicos. Debe pedir la ruta de la carpeta donde están mis archivos *CSV* procesados y ofrecer la opción de filtrar los datos por un intervalo de tiempo. Como salida, quiero gráficos de la evolución de la temperatura media y su desviación, así como un *CSV* con todas las estadísticas comparativas. Por favor, asegúrate de que el código permita filtrar por un tiempo_min y un tiempo_max que te indico desde la terminal y que la carpeta de entrada y salida te lo diga el usuario".

BLOQUE 4. Carga de datos y etiquetado de clases

Este bloque constituye la fase inicial de preparación de los datos para su uso en modelos de *Deep Learning*. En primer lugar, se carga el archivo *CSV* generado en el bloque anterior, el cual contiene las estadísticas térmicas de los distintos materiales en función del tiempo y de su estado (con o sin presencia de sales).

Posteriormente, el sistema identifica los materiales presentes a partir de las cabeceras de las columnas y solicita al usuario la asignación de una etiqueta numérica para cada uno de ellos. Esta asignación es esencial para definir las clases del problema de clasificación. Finalmente, el bloque determina el número total de clases distintas que serán utilizadas durante el proceso de aprendizaje.

Prompt empleado:
"Escribe un script para un bloque de preparación de datos de *Deep Learning*. Primero, el script debe cargar un archivo *CSV* con estadísticas comparativas (media y desviación estándar de evolución térmica de materiales con/sin sal a lo largo del tiempo). Luego, debe extraer los nombres únicos de los materiales leyendo las cabeceras de las columnas (las cuales tienen el formato: letra_material + ID_sal). Una vez identificados los materiales únicos, el programa debe pedir al usuario por consola que introduzca una etiqueta numérica (0, 1, 2...) para cada uno de ellos. Finalmente, mapea estas etiquetas a los datos y calcula una variable con el número total de clases únicas identificadas".

BLOQUE 4.1 Definición de la ruta base para resultados

Este bloque establece una ruta principal en Google Drive destinada al almacenamiento de todos los resultados generados durante el experimento de *Deep Learning*. Su objetivo es centralizar la información producida, incluyendo modelos entrenados, gráficos de rendimiento y tablas de resultados.

Prompt empleado:
"Escribe un bloque de código que defina una ruta base en Google Drive donde se guardarán todos los resultados generados por un experimento de *Deep Learning* (modelos, gráficos, tablas). El código debe pedirle al usuario que ingrese la ruta por texto".

BLOQUE 5. Ingeniería de características: cálculo de ratios térmicos

En este bloque se lleva a cabo la ingeniería de características, transformando los datos originales en variables más representativas para los modelos de *Deep Learning*. En particular, se calcula la derivada de la temperatura respecto al tiempo ($\Delta T/\Delta t$) para tres métricas: temperatura media, media más desviación estándar y media menos desviación estándar.

El usuario puede decidir si la diferencia de temperatura se calcula en valor absoluto o no, así como el intervalo temporal utilizado. Posteriormente, los datos son normalizados y estructurados para su uso tanto en modelos univariados como multivariados.

Prompt empleado:
"Escribe un bloque que realice la ingeniería de características para un modelo de *Deep Learning* basado en series temporales. El código debe calcular la derivada de la temperatura respecto al tiempo (delta T / delta t) para la media, la media + desviación, y la media – desviación que proviene del Bloque 3, y luego normalizar estos resultados. Antes de calcular, el bloque debe preguntar al usuario por la terminal dos cosas: 1) Si la diferencia de temperatura en el numerador debe ser en valor absoluto o no, y 2) Cuál es el intervalo de tiempo en el denominador a utilizar. Finalmente, el código debe devolver los datos preparados y formateados tanto para modelos univariados como multivariados".

BLOQUE 6. Codificación de las etiquetas de salida

Este bloque transforma las etiquetas numéricas en formato One-Hot Encoding, necesario para el entrenamiento de modelos de clasificación multiclase en Keras/TensorFlow. Este formato permite que la red neuronal interprete correctamente las distintas categorías de salida.

Prompt empleado:
"Escribe un bloque que prepare las etiquetas de salida para entrenar un modelo de *Deep Learning* multiclase. El bloque debe tomar un array de etiquetas numéricas y convertirlo al formato de codificación One-Hot utilizando Keras/TensorFlow. Por favor, documenta el código explicando que este formato es el requerido para que la red neuronal procese correctamente las distintas clases de salida".

BLOQUE 7. División del conjunto de datos

Este bloque divide los datos de entrada (X) y las etiquetas de salida (Y) en conjuntos de entrenamiento, validación y prueba. El propósito es reparar los datos para el proceso de aprendizaje y evaluación de los modelos. La división asegura que los modelos sean entrenados en un subconjunto de datos, actualizando sus pesos, validados en otro subconjunto durante el entrenamiento para visualizar sus rendimientos en datos no vistos a lo largo de las iteraciones/épocas del aprendizaje (epochs), y finalmente evaluados en un subconjunto completamente independiente para estimar su rendimiento en datos no vistos. La selección de los datos es aleatoria, pero garantizando número de datos por clase igual o semejante en cada conjunto, permitiendo al usuario cuántos datos utilizar para cada conjunto.

Prompt empleado:
"Escribe un bloque que divida aleatoriamente un conjunto de datos (variables X y etiquetas Y) en conjuntos de Entrenamiento, Validación y Prueba para un modelo de *Deep Learning*. El código debe preguntar al usuario por terminal qué proporción o cantidad de datos quiere asignar a cada uno de los tres conjuntos. Es estrictamente necesario que la selección sea aleatoria pero estratificada, es decir, garantizando que el número de datos por clase sea proporcional y semejante en cada uno de los tres conjuntos resultantes".

BLOQUE 8. Definición de modelos de *Deep Learning*

Este bloque divide los datos de entrada (X) y las etiquetas de salida (Y) en conjuntos de entrenamiento, validación y prueba. El propósito es reparar los datos para el proceso de aprendizaje y evaluación de los modelos. La división asegura que los modelos sean entrenados en un subconjunto de datos, actualizando sus pesos, validados en otro subconjunto durante el entrenamiento para visualizar sus rendimientos en datos no vistos a lo largo de las iteraciones/épocas del aprendizaje (epochs), y finalmente evaluados en un subconjunto completamente independiente para estimar su rendimiento en

datos no vistos. La selección de los datos es aleatoria, pero garantizando número de datos por clase igual o semejante en cada conjunto, permitiendo al usuario cuántos datos utilizar para cada conjunto.

Prompt empleado:
"Escribe un bloque usando Keras/TensorFlow que defina las arquitecturas de tres modelos de *Deep Learning* para clasificación de series temporales: RNN simple, LSTM y GRU. El código debe preparar el terreno para crear versiones univariadas y multivariadas de cada modelo. Configura todos los modelos utilizando hiperparámetros estándar del estado del arte y compílalos estrictamente con el optimizador Adam, la función de pérdida categorical_crossentropy y la métrica de accuracy".

BLOQUE 9. Aprendizaje de los modelos de *Deep Learning* a usar (RNN, LSTM y GRU)

Este bloque entrena todos los modelos definidos en el Bloque 8 utilizando los datos de entrenamiento y validación del Bloque 7. El propósito es ajustar los parámetros de los datos de entrada a los modelos de *Deep Learning* (diferencia de temperatura absoluta sí o no, el intervalo de tiempo seleccionado para los ratios de cambio) para que aprendan a clasificar las series temporales lo mejor posible. Durante el entrenamiento, se monitoriza el rendimiento en el conjunto de validación para prevenir el sobreajuste y guardar la mejor versión del modelo, es decir, se para en el epoch donde la pérdida en validación llegue al mínimo. Por tanto, va haber un diccionario que almacena el historial de entrenamiento (pérdida y precisión para entrenamiento y validación por época) para cada modelo y una ruta al directorio donde se guardan las mejores versiones de cada modelo (los pesos con la menor pérdida de validación).

Prompt empleado:
"Escribe un bloque usando Keras que se encargue del entrenamiento de los modelos RNN, LSTM y GRU (tanto univariados como multivariados) definidos en el bloque previo. El código debe aprender cada modelo usando datos de entrenamiento y validación del Bloque 7. Es fundamental prevenir el sobreajuste: el código debe monitorizar el rendimiento y detenerse automáticamente en la época (epoch) donde la pérdida en validación llegue a su mínimo, guardando en una ruta del directorio la mejor versión del modelo. Además, el script debe inicializar un diccionario donde se guarde el historial completo de aprendizaje (precisión y pérdida en train y val por cada época) para todos los modelos evaluados".

BLOQUE 10. Establecer ruta para guardado del aprendizaje de los modelos de *Deep Learning*

Este bloque define un directorio específico para guardar los resultados de la evaluación de los modelos. El propósito es organizar los resultados generados durante la fase de evaluación, que son los gráficos de rendimiento (pérdida y precisión en entrenamiento y validación)) y tablas de probabilidades de predicción y las clases reales/predichas para cada conjunto de datos aplicando el modelo con sus pesos en el mejor epoch.

Prompt empleado:
"Escribe un bloque que defina y cree directorios específicos para guardar los resultados de evaluación de los modelos de *Deep Learning*. El código debe tomar la ruta base global (definida en Bloque 4.1) y crear dentro de ella una estructura organizada destinada a almacenar dos tipos de archivos: 1) Gráficos de rendimiento (curvas de pérdida y precisión en train/val), y 2) Tablas de probabilidades de predicción y clases reales vs predichas usando el modelo en su mejor epoch".

BLOQUE 11. Preparación previa para guardado del aprendizaje

Este bloque define funciones auxiliares que serán utilizadas en el siguiente bloque para evaluar y guardar los resultados de los modelos. El propósito es modularizar el código de evaluación y visualización, haciendo el Bloque 12 más limpio y fácil de entender. Incluye funciones para obtener los

datos de entrada correctos para cada modelo, plotear el historial de entrenamiento/validación y crear las tablas de predicción.

Prompt empleado:

"Escribe un bloque que defina las funciones auxiliares necesarias para evaluar modelos de *Deep Learning* (RNN, LSTM, GRU) en series temporales. El propósito es modularizar el código antes del Bloque 12. Necesito que definas al menos tres funciones: una para adaptar los datos de entrada al formato correcto del modelo, otra para graficar y guardar el historial de entrenamiento/validación (pérdida y precisión), y una última para generar y guardar un archivo *CSV* con las tablas de predicción (probabilidades, clase real y clase predicha)".

BLOQUE 12. Evaluación final y consolidación de resultados

Este último bloque evalúa cada modelo aprendido, genera gráficos de rendimiento y tablas de predicción tal y como se ha comentado en el Bloque 10, y guarda todos estos resultados. También identifica el modelo con el mejor rendimiento en el conjunto de validación y copia sus resultados a una carpeta de resumen. El propósito es realizar una evaluación exhaustiva de cada modelo aprendido, visualizar su rendimiento a lo largo de las épocas, analizar sus predicciones y consolidar los resultados del mejor modelo para una revisión fácil.

Prompt empleado:

"Escribe un bloque que actúe como la fase final de evaluación de un proyecto de *Deep Learning* con modelos RNN, LSTM y GRU. El código debe recorrer todos los modelos previamente aprendidos y utilizar funciones auxiliares (definidas en el bloque anterior) para generar y guardar sus gráficos de rendimiento por épocas y sus tablas de predicción *CSV*. Además, el bloque debe analizar los resultados en el conjunto de validación de cada modelo para identificar cuál de todos tuvo el mejor rendimiento. Finalmente, debe crear una carpeta de resumen y copiar allí de forma automática los gráficos y tablas de este modelo ganador, consolidando los resultados para una revisión fácil. ".

En la Figura 3-15 ubicada en la siguiente página se muestra el diagrama de flujo de los bloques de código desarrollados para esta primera fase.

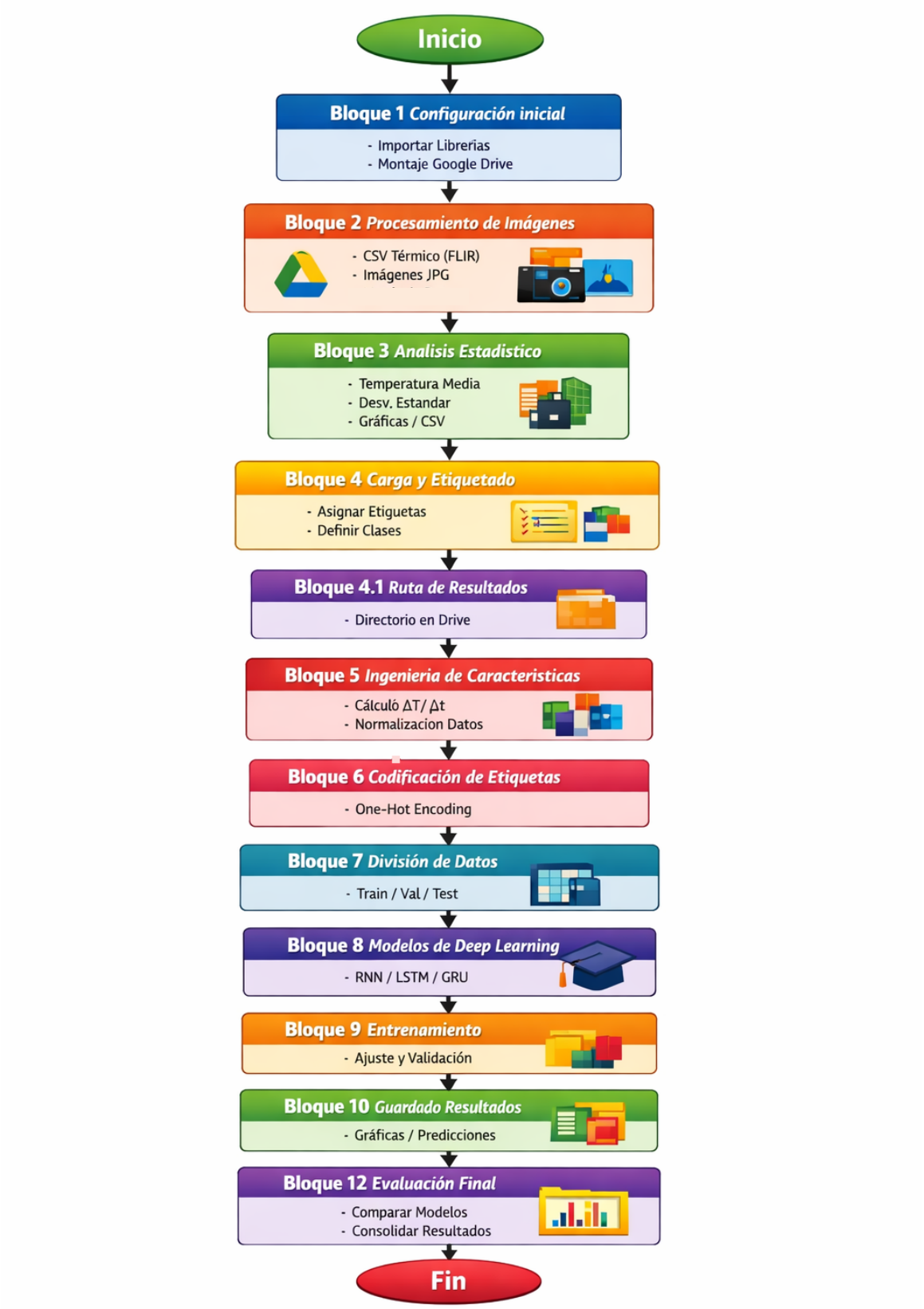


Figura 3-15. Diagrama de flujo de la primera etapa de Predicción de Materiales (Elaborado con inteligencia artificial Gemini)

3.5.2 Segunda etapa. Detección de sales

BLOQUE 1. Configuración Inicial y Montaje de Google Drive

Este bloque inicial se encarga de importar la librería necesaria para interactuar con Google Drive y procede a montar su unidad de Google Drive en el entorno de Colab. Esto permite que el código acceda a los archivos almacenados en su Drive. La entrada es su autenticación en Google Drive. La salida es el montaje de Google Drive en "/content/drive" y un mensaje de confirmación.

Prompt empleado:
"Configura el entorno e interactúa con Google Drive para montar mi unidad en el entorno de Colab".

BLOQUE 2. Cálculo de Diferencias de Temperatura y Generación de Curvas de Sal

Este bloque carga el archivo *CSV* de estadísticas generado en el código anterior, el de predicción de material. Identifica las columnas que representan la "media" de diferentes series y solicita al usuario un identificador numérico para usar como referencia (que será en este caso la referencia correspondiente a la muestra sin sal). Luego, calcula la diferencia de temperatura entre cada serie y su serie de referencia correspondiente, en cada tipo de material (por ejemplo, "Material1_X_Media" - "Material1_Referencia_Media"). El resultado es un nuevo archivo *CSV* que contiene el tiempo y las curvas de diferencia calculadas para cada serie. Las entradas son: la ruta del archivo "Estadisticas_Comparativa_Evolucion.csv", y la parte numérica del identificador de la columna de referencia (ej. "6"). La salida es: un nuevo archivo *CSV* (ej. " Curvas_Sal.csv") con las curvas de diferencia.

Prompt empleado:
"Genera el código para calcular la diferencia entre las columnas "_Media" de un archivo *CSV* de estadísticas y una columna de referencia específica para cada material. El usuario deberá proporcionar la ruta completa del archivo "Estadisticas_Comparativa_Evolucion.csv" y la parte numérica del identificador de la columna de referencia (ej. "6"). El resultado será un nuevo archivo *CSV* que contenga el tiempo y las curvas de diferencia calculadas para cada serie, guardado con un nombre como "Curvas_Sal.csv".

BLOQUE 3. Carga de datos de entrada para *Deep Learning* y etiquetado de clases

Este bloque carga el archivo *CSV* de diferencias generado previamente ("Curvas_Sal.csv"). Su función principal es identificar automáticamente los nombres de los "materiales" a partir de las columnas del *CSV* y luego solicitar al usuario que asigne una etiqueta numérica (clase) a cada uno de esos materiales. Estas etiquetas numéricas son la base para la clasificación en el modelo de *Deep Learning*. Las entradas son: la ruta del archivo *CSV* de diferencias, y las etiquetas numéricas que el usuario asigna a cada material. Las salidas son: un DataFrame con los datos de diferencias, un diccionario que mapea los nombres de los materiales a sus etiquetas numéricas y el número total de clases únicas.

Prompt empleado:
"Carga el archivo *CSV* de diferencias ("Curvas_Sal.csv") en un DataFrame. Identifica automáticamente los nombres de los materiales a partir de las columnas del *CSV*. Luego, solicita al usuario que asigne una etiqueta numérica (clase) a cada material. Las salidas deben ser: el DataFrame con los datos de diferencias, un diccionario que mapea los nombres de los materiales a sus etiquetas numéricas, y el número total de clases únicas".

BLOQUE 4.1. Definición de ruta base global para resultados del experimento

Este bloque solicita al usuario una ruta en Google Drive que servirá como directorio raíz para guardar todos los resultados del experimento de *Deep Learning*, incluyendo modelos entrenados, gráficos de rendimiento y tablas de predicción. Se asegura de que esta carpeta exista. La entrada es: la ruta completa de la carpeta donde se guardarán los resultados del experimento. La salida es: la creación del directorio especificado en Google Drive y la configuración de una variable global "base_experiment_dir" con esta ruta.

Prompt empleado:
"Configura una ruta global en Google Drive para guardar todos los resultados del experimento de *Deep Learning*. Se pedirá la ruta completa de la carpeta, la cual será creada si no existe".

BLOQUE 4.2. Preparación del formato y datos para el resultado

Este bloque es crucial para transformar los datos de diferencias en el formato adecuado para los modelos de *Deep Learning*. Permite al usuario elegir entre calcular "derivadas" (ratios de temperatura frente al tiempo) o "integrales" (áreas acumuladas mediante la regla del trapecio) a partir de las diferencias de temperatura. También ofrece opciones para aplicar valores absolutos o logaritmo a las diferencias de temperatura, como el intervalo de tiempo deseado a usar. Las características resultantes son normalizadas mediante escalado Min-Max y preparadas como series de tiempo univariadas para los modelos. Finalmente, crea un subdirectorio específico para este experimento. Las entradas son: el DataFrame de diferencias, las etiquetas de los materiales, y las elecciones del usuario sobre las características (valor absoluto, delta_t (que es el intervalo temporal), logaritmo, tipo de característica (que es elegir entre "derivadas" o "integrales"). Las salidas son: los datos de entrada "X_uni_media" listos para el aprendizaje (como un array NumPy con forma (número de muestras, longitud_secuencia, 1)), los nombres de los materiales, las etiquetas y la ruta del directorio específico del experimento.

Prompt empleado:
"Genera un bloque de código para la ingeniería y preparación de características para modelos de *Deep Learning*. Este bloque debe transformar los datos de diferencias en un formato adecuado para los modelos. Permite al usuario elegir entre calcular "derivadas" (ratios de temperatura) o "integrales" (áreas acumuladas) a partir de las diferencias de temperatura, el intervalo temporal a utilizar, y opcionalmente aplicar valores absolutos o el logaritmo base 10 a las diferencias de temperatura. Las características resultantes deben ser normalizadas mediante escalado Min-Max y preparadas como series de tiempo univariadas con la forma (muestras, longitud_secuencia, 1). Finalmente, crea un subdirectorio específico para este experimento en la ruta de resultados definida previamente y guarda en variables de salida los datos de entrada "X_uni_media" listos para el entrenamiento, los nombres de los materiales, las etiquetas y la ruta del directorio del experimento."

BLOQUE 5. Preprocesamiento de etiquetas de clase

Este bloque toma las etiquetas numéricas asignadas a los materiales y las convierte al formato "one-hot encoding". Este formato es esencial para la mayoría de los modelos de clasificación en *Deep Learning*, donde cada etiqueta se representa como un vector binario. Las entradas son: las etiquetas numéricas de los materiales y el número total de clases. La salida es: "Y_encoded", un array NumPy con las etiquetas en formato one-hot encoding.

Prompt empleado:
"Preprocesa las etiquetas de clase para *Deep Learning*. Se necesita convertir las etiquetas numéricas asignadas a los materiales al formato "one-hot encoding", que es esencial para la mayoría de los modelos de clasificación. Utiliza las etiquetas numéricas de los materiales (material_labels), los nombres únicos

de los materiales (unique_material_names_fe) y el número total de clases (num_classes) como entradas. La salida debe ser Y_encoded, un array NumPy con las etiquetas en formato one-hot encoding".

BLOQUE 6. División de datos en conjunto de entrenamiento, validación y prueba

Este bloque divide los datos de características ("X_uni_media") y las etiquetas ("Y_encoded") en tres subconjuntos: entrenamiento, validación y prueba. La división se realiza de manera aleatoria y estratificada para asegurar que la proporción de clases se mantenga similar en cada conjunto, lo cual es importante para un entrenamiento robusto. Las entradas son: los datos de características procesados, las etiquetas one-hot encoded, y los nombres de los materiales. Las salidas son: los conjuntos de datos de entrenamiento (X_train, Y_train), validación (X_val, Y_val) y prueba (X_test, Y_test), así como los índices originales de las muestras en cada conjunto.

Prompt empleado:
"Divide los datos de características "X_uni_media" y las etiquetas "Y_encoded" en conjuntos de entrenamiento, validación y prueba. La división debe ser aleatoria y estratificada, manteniendo las proporciones de las clases en cada conjunto. Utiliza los nombres únicos de los materiales (unique_material_names_fe) para la estratificación. Las salidas deben ser "X_train_uni_media", "X_val_uni_media", "X_test_uni_media" para las características, "Y_train", "Y_val", "Y_test" para las etiquetas, y los índices originales de las muestras en cada conjunto (X_train_idx, X_val_idx, X_test_idx). Configura el conjunto de entrenamiento con 50 muestras, el de validación con 10 y no utilices conjunto de prueba (0 muestras)".

BLOQUE 7. Definición y compliación de modelos de *Deep Learning* (RNN, LSTM, GRU)

Este bloque define la arquitectura de los modelos de Redes Neuronales Recurrentes (RNN), Long Short-Term Memory (LSTM) y Gated Recurrent Unit (GRU). Se construye una función que permite crear y compilar estos modelos con configuraciones específicas (capa RNN, capas densas, dropout). Los modelos se compilan usando el optimizador Adam y la función de pérdida "categorical_crossentropy", apropiada para la clasificación multiclase. Las configuraciones utilizadas son las estándares, las usadas en el estado del arte. Las entradas son: el número de clases, la longitud de la secuencia de características y la forma de los datos de entrenamiento. La salida es: un diccionario "models_univariate" que contiene los tres modelos de Keras compilados (RNN_Media, LSTM_Media, GRU_Media).

Prompt empleado:
"Define y compila modelos de *Deep Learning* para clasificación utilizando arquitecturas RNN, LSTM y GRU. Crea una función build_rnn_model que construya estos modelos con una capa recurrente, capas densas con activación "relu", y una capa de salida "softmax" para clasificación multiclase. Cada modelo debe usar una tasa de dropout de 0.2, 64 unidades para la capa recurrente (RNN, LSTM, GRU) y 32 unidades para la capa densa. Compila los modelos con el optimizador Adam y la función de pérdida "categorical_crossentropy". Las entradas para la función deben ser el tipo de modelo ("SimpleRNN", "LSTM", "GRU"), la forma de entrada (input_shape), y el número de clases (num_classes). Utiliza las variables num_classes y sequence_length_fe (de las celdas anteriores) para establecer las formas de entrada correctas para datos univariados de "Media". La salida debe ser un diccionario models_univariate que contenga los tres modelos compilados: SimpleRNN_Media, LSTM_Media y GRU_Media".

BLOQUE 8. Aprendizaje de los modelos de *Deep Learning*

Este bloque se encarga de aprender cada uno de los modelos de *Deep Learning* definidos. Durante el aprendizaje, se utiliza un "callback" para guardar la mejor versión del modelo (basada en la menor pérdida en validación) y también la versión del modelo de la última época (epoch). Las entradas son: el

diccionario de modelos compilados, los conjuntos de datos de entrenamiento y validación, y la ruta de salida del experimento. Las salidas son: un historial de aprendizaje para cada modelo y los archivos .keras de los modelos entrenados (versiones "_best" y "_last") guardados en un subdirectorio "trained_models".

Prompt empleado:

"Entrena los modelos de *Deep Learning* (RNN, LSTM, GRU) previamente definidos. Durante el entrenamiento, utiliza callbacks para guardar la mejor versión del modelo (basada en la menor pérdida de validación) y también la versión del modelo de la ùltima época. La función de entrenamiento debe tomar como entradas el diccionario de modelos compilados, los conjuntos de datos de entrenamiento y validación, y la ruta de salida del experimento. Como salidas, la función debe devolver el historial de aprendizaje para cada modelo y guardar los modelos entrenados como archivos .keras (con sufijos "_best" y "_last") en un subdirectorio llamado "trained_models" dentro de la ruta de salida del experimento".

BLOQUE 9.1. Configuración del directorio de evaluación

Este bloque simplemente configura y crea el directorio donde se almacenarán todos los resultados de la evaluación del modelo, como gráficos de rendimiento y tablas de predicciones. Utiliza la ruta del experimento previamente definida. La entrada es: la ruta de salida del experimento. La salida es: la creación del directorio "evaluation_plots" dentro de la carpeta del experimento.

Prompt empleado:

"Configura el directorio para guardar los resultados de la evaluación de los modelos de *Deep Learning*."

BLOQUE 9.2. Preparación previa para guardado del aprendizaje de los modelos de *Deep Learning*

Este bloque constituye una etapa fundamental de preparación, estableciendo un conjunto de funciones auxiliares diseñadas para la evaluación exhaustiva y la visualización clara de los resultados de los modelos entrenados. Esta fase es crítica para transformar los datos crudos de entrenamiento y predicción en información interpretable y actionable.

Prompt empleado:

"Define un conjunto de funciones auxiliares para la evaluación y visualización de los modelos de *Deep Learning*. Necesito funciones para seleccionar los conjuntos de datos correctos (entrenamiento, validación, prueba) según el tipo de modelo, para generar gráficos de la evolución de la pérdida y la precisión durante el entrenamiento, para crear tablas detalladas de las predicciones de los modelos (incluyendo probabilidades por clase, clases reales y predichas) guardadas en formato *CSV*, y una función para acortar los nombres largos de los materiales en las visualizaciones y tablas para mayor claridad."

BLOQUE 9.3. Evaluación detallada de los modelos aprendidos y generación de informes

Este bloque realiza una evaluación exhaustiva de cada modelo aprendido (tanto la versión "_best.keras" como "_last.keras"). Para cada modelo y para cada conjunto de datos (entrenamiento, validación y prueba), carga el modelo, realiza predicciones y genera los siguientes resultados: gráficos de rendimiento (pérdida y precisión vs. época) y tablas *CSV* detalladas con las probabilidades de predicción y las clases reales/predichas para visualizar el rendimiento de clasificación. Todas estas salidas se guardan en el directorio de evaluación. Las entradas son: los modelos entrenados, los

historiales de aprendizaje, todos los conjuntos de datos (X e Y para train, val, test), los índices de las muestras, los nombres de los materiales, las etiquetas de clase y el directorio de evaluación. Las salidas son: múltiples archivos *PNG* con gráficos de rendimiento y archivos *CSV* con las tablas de predicción para cada modelo y conjunto de datos.

Prompt empleado:

"Genera el código para evaluar exhaustivamente los modelos de *Deep Learning* entrenados. Para cada modelo (tanto la versión "_best.keras" como "_last.keras"), y para cada conjunto de datos (entrenamiento, validación y prueba), el código debe cargar el modelo, realizar predicciones y generar los siguientes resultados: gráficos de rendimiento (pérdida y precisión vs. Época, aquí solo al conjunto de datos de entrenamiento y de validación) y tablas *CSV* detalladas con las probabilidades de predicción y las clases reales/predichas para cada conjunto de datos. Asegura que todas estas salidas se guarden en el directorio de evaluación especificado".

En la siguiente página, en la Figura 3-16, se muestra el diagrama de flujo de los bloques de códigos desarrollados para esta segunda y última fase.

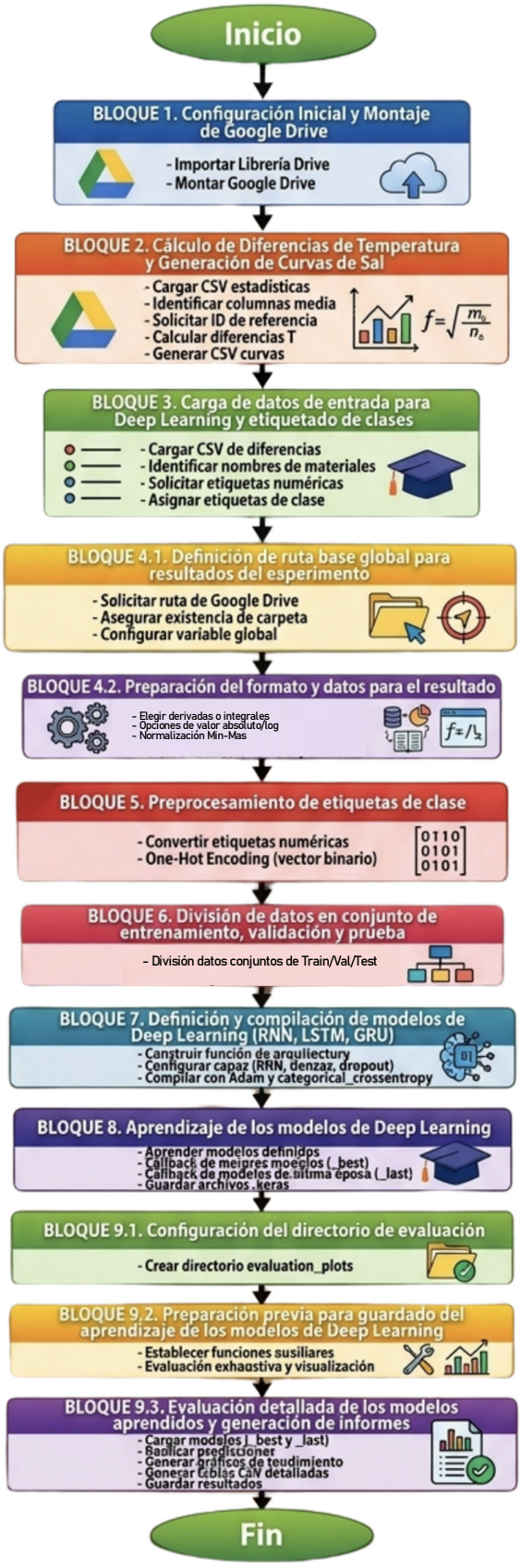


Figura 3-16. Diagrama de flujo de la segunda etapa de Predicción de Sales (Elaborado con inteligencia artificial Gemini)

4 RESULTADOS Y DISCUSIÓN

4.1 Caracterización mecánica de las muestras

Para la toma de datos de ultrasonidos y dureza superficial de las muestras se tiene en cuenta la siguiente referencia representada en la Figura 4-1.

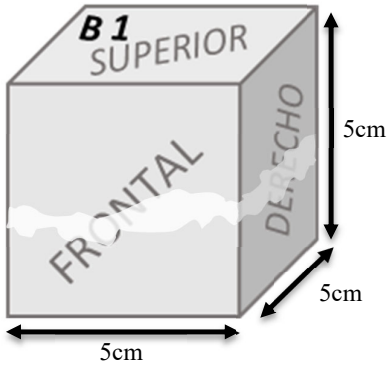


Figura 4-1. Referencia para la toma de datos de las muestras (Elaboración propia)

Hay dos caras de estudio, la superior y la inferior. Siendo estas superficies las elegidas para la caracterización debido a que, al ser contaminada la muestra en un plano paralelo intermedio a la cara superior e inferior de la muestra, es el sentido óptimo para la medición de la prueba de ultrasonidos y dureza superficial.

4.1.1 Propiedades de las muestras previo a la adición de sales

En primer lugar, se ha llevado a cabo la adquisición de medidas de velocidad ultrasónica y dureza en las muestras en su estado inicial, es decir, antes de someterlas al proceso de contaminación. Estas mediciones se han establecido como referencia para su posterior comparación con los resultados obtenidos tras la exposición a disoluciones salinas. Asimismo, cabe destacar que dichas medidas se han realizado únicamente sobre las muestras identificadas con el número "1", dado que todas las probetas presentan las mismas características y condiciones iniciales (ver Tablas 4-1 y 4-2).

Tipo de material	Muestra	Lado	Tiempo de vuelo (TOF) [µs]	Velocidad [m/s]
Granito de Madrid	1	Superior-Inferior	9,8	5102
Granito Azul Alpendorada	A1	Superior-Inferior	10,7	4673
Caliza Bateig	B1	Superior-Inferior	14,1	3546
Arenisca Dorada Urbión	D1	Superior-Inferior	27,3	1832
Arenisca Sierra de la Demanda	S1	Superior-Inferior	18,9	2646

Tipo de material	Muestra	Lado	Tiempo de vuelo (TOF) [µs]	Velocidad [m/s]
Arenisca Villamayor	V1	Superior-Inferior	21,5	2326

Tabla 4-1. Medición de ultrasonidos en las muestras previo a la contaminación con sales (Elaboración propia)

Tipo de material	Muestra	Lado	Dureza por rebote [HLD] (media)	Promedio	Desviación típica
Granito de Madrid	1	Superior	827,6	813,9	± 13,7
		Inferior	800,2		
Granito Azul Alpendorada	A1	Superior	813,2	821,9	± 8,7
		Inferior	830,6		
Caliza Bateig	B1	Superior	459,6	457,7	± 1,9
		Inferior	455,8		
Arenisca Dorada Urbión	D1	Superior	502,4	496,5	± 5,9
		Inferior	490,6		
Arenisca Sierra de la Demanda	S1	Superior	275,6	271,4	± 4,2
		Inferior	267,2		
Arenisca Villamayor	V1	Superior	361,8	347,6	± 14,2
		Inferior	333,4		

Tabla 4-2. Medición de la prueba de dureza en las muestras previo a la contaminación con sales (Elaboración propia)

4.1.2 Propiedades de las muestras tras la adición de sales

Tras la adición de sales, se ha llevado a cabo la evaluación de las propiedades mecánicas de las muestras mediante ensayos de velocidad ultrasónica y dureza superficial, observándose un comportamiento claramente diferenciado en función del tipo de material al comparar los resultados con los obtenidos previamente a la contaminación. En los materiales más compactos, los valores de velocidad ultrasónica se mantienen elevados y próximos a los iniciales, al igual que la dureza superficial, lo que pone de manifiesto su baja susceptibilidad frente a los procesos de degradación inducidos por la cristalización de sales. No obstante, se aprecia un incremento en la dispersión de los resultados de dureza

respecto a las mediciones iniciales, lo que sugiere una afectación superficial heterogénea, sin comprometer de manera significativa la integridad interna del material.

Por el contrario, en los materiales de carácter más poroso, las diferencias respecto al estado previo a la contaminación son notablemente más acusadas. Se observa una disminución generalizada de la velocidad ultrasónica, indicativa de un deterioro interno asociado a la formación de microfisuras y al aumento de la porosidad efectiva. Este comportamiento se acompaña de una reducción significativa de la dureza superficial, reflejando una pérdida de cohesión en las capas más externas del material. Asimismo, la comparación entre los valores iniciales y finales evidencia un incremento en la dispersión de los resultados, especialmente en los ensayos de dureza, lo que pone de manifiesto la naturaleza heterogénea del deterioro.

En los casos más severos, el grado de degradación alcanzado ha impedido la correcta realización de determinadas mediciones debido a la elevada irregularidad superficial y a la pérdida de material, lo que dificulta el adecuado contacto de los equipos de ensayo. Estas situaciones se han indicado en las tablas mediante el símbolo "(-)", correspondiente a aquellas mediciones que no han podido realizarse debido a la irregularidad de la superficie, la cual impedía la correcta ejecución de los ensayos de ultrasonido y de dureza superficial mediante durómetro. En conjunto, los resultados muestran una clara coherencia entre la evolución de los parámetros medidos y el estado físico de las muestras tras la contaminación, evidenciando que los mayores descensos en velocidad ultrasónica y dureza se corresponden con los materiales que presentan un deterioro más acusado, lo que confirma la estrecha relación entre las propiedades petrofísicas de los materiales y su susceptibilidad frente a los procesos de alteración inducidos por sales (ver Tabla 4-3 y Tabla 4-4).

Tipo de material	Muestra	Lado	Tiempo de vuelo (TOF) [µs]	Velocidad [m/s]
Granito de Madrid	1	Superior-Inferior	9,9	5051
	2	Superior-Inferior	11,4	4386
	3	Superior-Inferior	9,9	5051
	4	Superior-Inferior	8,9	5618
	5	Superior-Inferior	9,4	5319
Granito Azul Alpendorada	A1	Superior-Inferior	10,7	4673
	A2	Superior-Inferior	9,9	5051
	A3	Superior-Inferior	8,4	5952
	A4	Superior-Inferior	9	5556

Tipo de material	Muestra	Lado	Tiempo de vuelo (TOF) [µs]	Velocidad [m/s]
	A5	Superior-Inferior	9,4	5319
	B1	Superior-Inferior	14,9	3356
	B2	Superior-Inferior	-	-
Caliza Bateig	B3	Superior-Inferior	13,3	3759
	B4	Superior-Inferior	13,7	3649
	B5	Superior-Inferior	16,1	3106
	D1	Superior-Inferior	20,9	2392
	D2	Superior-Inferior	-	-
Arenisca Dorada Urbión	D3	Superior-Inferior	21,9	2283
	D4	Superior-Inferior	16,6	3012
	D5	Superior-Inferior	21,8	2294
	S1	Superior-Inferior	18,1	2762
	S2	Superior-Inferior	-	-
Arenisca de la Sierra de la Demanda	S3	Superior-Inferior	19,8	2525
	S4	Superior-Inferior	15,4	3247
	S5	Superior-Inferior	20,7	2415
Arenisca Villanueva	V1	Superior-Inferior	30,7	1629

Tipo de material	Muestra	Lado	Tiempo de vuelo (TOF) [µs]	Velocidad [m/s]
	V2	Superior-Inferior	-	-
	V3	Superior-Inferior	28,1	1779
	V4	Superior-Inferior	-	-
	V5	Superior-Inferior	30,5	1639

Tabla 4-3. Medición de ultrasonidos en las muestras tras la contaminación con sales (Elaboración propia)

Tipo de material	Muestra	Lado	Dureza por rebote [HLD] (media)	Promedio	Desviación típica
Granito de Madrid	1	Superior	781	730,4	± 51,4
		Inferior	679		
	2	Superior	788,6	787	± 6,6
		Inferior	775,4		
	3	Superior	648,2	700,4	± 52,2
		Inferior	752,6		
	4	Superior	850,4	728,3	± 122,1
		Inferior	606,2		
	5	Superior	769,2	767,9	± 1,3
		Inferior	766,6		
Granito Azul Alpendorada	A1	Superior	703,2	759	± 56
		Inferior	815,2		
	A2	Superior	800,4	818,9	± 18,5
		Inferior	837,4		

Tipo de material	Muestra	Lado	Dureza por rebote [HLD] (media)	Promedio	Desviación típica
Caliza Bateig	A3	Superior	650,6	698	± 47,4
		Inferior	745,4		
	A4	Superior	820,8	750,9	± 69,9
		Inferior	681		
	A5	Superior	813,6	810,6	± 3
		Inferior	807,6		
	B1	Superior	257,6	257,8	± 0,2
		Inferior	258		
	B2	Superior	303,6	312,9	± 9,3
		Inferior	322,2		
	B3	Superior	321,4	334,2	± 12,8
		Inferior	347		
	B4	Superior	331,4	279,3	± 52,1
		Inferior	227,2		
	B5	Superior	202,4	242,5	± 40,1
		Inferior	282,6		
Arenisca Dorada Urbión	D1	Superior	251,2	295,2	± 44
		Inferior	339,2		
	D2	Superior	-	-	-
		Inferior	-		
	D3	Superior	249	308,8	± 59,8
		Inferior	368,6		

Tipo de material	Muestra	Lado	Dureza por rebote [HLD] (media)	Promedio	Desviación típica
Arenisca Sierra de la Demanda	D4	Superior	409,2	401	± 8,2
		Inferior	392,8		
	D5	Superior	271,2	289,8	± 18,6
		Inferior	308,4		
	S1	Superior	377,6	345,7	± 31,9
		Inferior	313,8		
	S2	Superior	202	290,5	± 88,5
		Inferior	379		
	S3	Superior	323,2	357,9	± 34,7
		Inferior	392,6		
	S4	Superior	443	378,5	± 64,5
		Inferior	314		
	S5	Superior	279	366,8	± 87,8
		Inferior	454,6		
Arenisca Villanueva	V1	Superior	203	202,6	± 0,4
		Inferior	202,2		
	V2	Superior	192	230	± 38
		Inferior	268		
	V3	Superior	236,4	259,2	± 22,8
		Inferior	282		
	V4	Superior	240,2	231,5	± 8,7

Tipo de material	Muestra	Lado	Dureza por rebote [HLD] (media)	Promedio	Desviación típica
		Inferior	222,8		
		Superior	233,6		
	V5			258,1	± 24,5
		Inferior	282,6		

Tabla 4-4. Medición de la prueba de dureza en las muestras tras la contaminación con sales (Elaboración propia)

4.2 Contaminación de las muestras con sales

Durante los tres primeros ciclos no se han observado cambios significativos en la mayor parte de las muestras. En la Figura 4-2 se muestra el estado de las muestras contaminadas con la disolución salina deNa_2SO_4 tras la finalización del tercer ciclo, pues son las únicas muestras que comienzan a mostrar cambios visuales. En esta etapa aún no se aprecia una cristalización salina claramente desarrollada en la mayoría de los materiales. No obstante, comienzan a observarse ligeras acumulaciones de material en bordes y aristas, así como variaciones de tonalidad en algunas superficies, indicativas del ascenso capilar de la disolución salina a través de la red porosa del material.

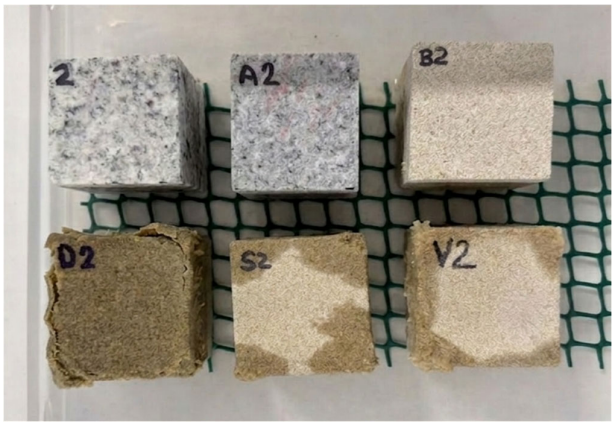


Figura 4-2. Muestras contaminadas con Na_2SO_4 tras el tercer ciclo (Elaboración propia)

En el cuarto ciclo comienzan a apreciarse de forma más evidente los primeros signos de alteración asociados a la cristalización de sales. Las muestras de granito mantienen un comportamiento estable, sin cambios significativos más allá de ligeras variaciones cromáticas superficiales. Por el contrario, en las areniscas, especialmente la Dorada Urbión y la de Villamayor, ya se observa una incipiente acumulación de sales en forma de eflorescencias superficiales, localizadas principalmente en aristas y caras expuestas. La arenisca de Sierra de la Demanda muestra un comportamiento intermedio, con depósitos menos desarrollados pero ya perceptibles. En el caso de la caliza Bateig , se detecta una leve aparición de depósitos blanquecinos, indicativos del inicio del proceso de migración salina (ver Figura 4-3).

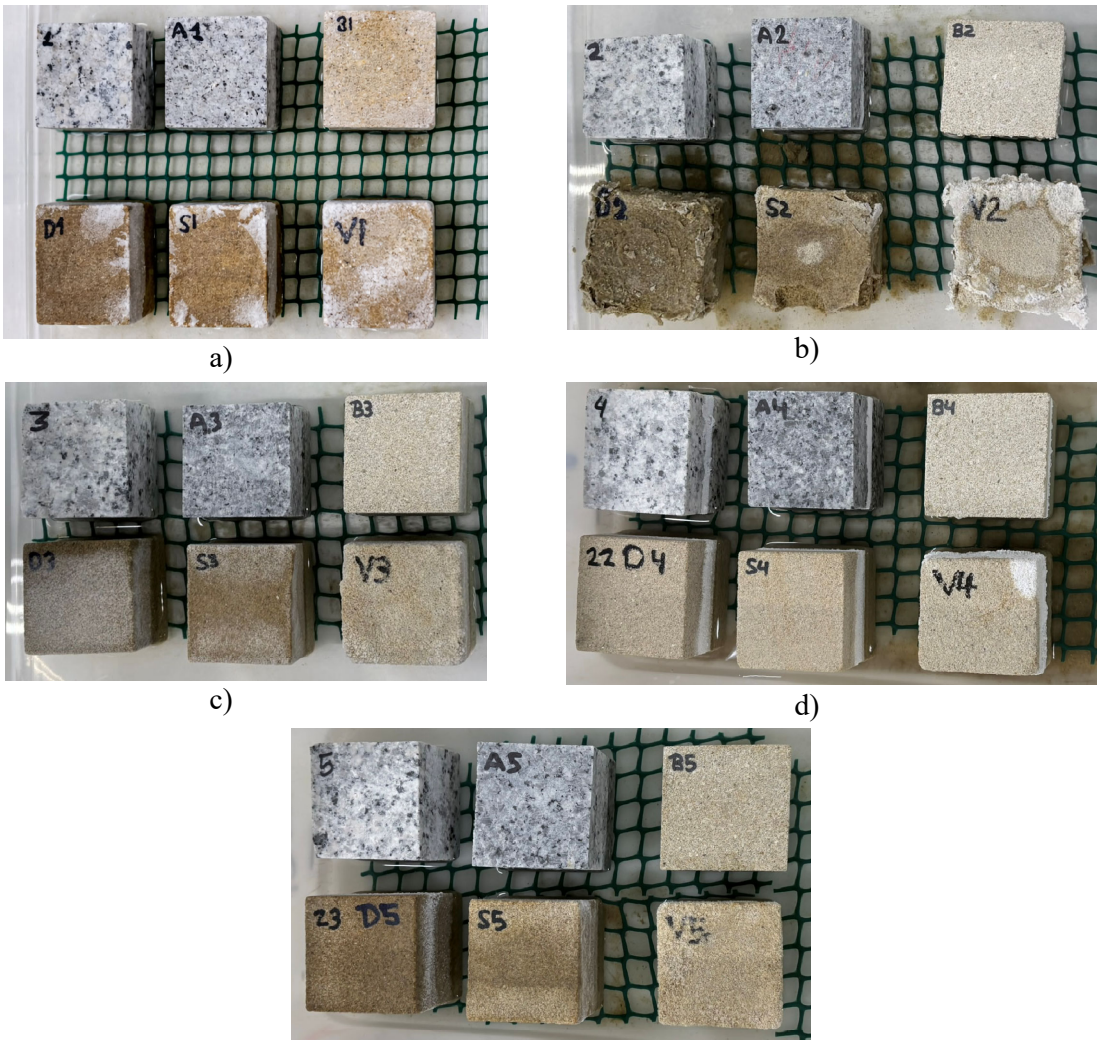
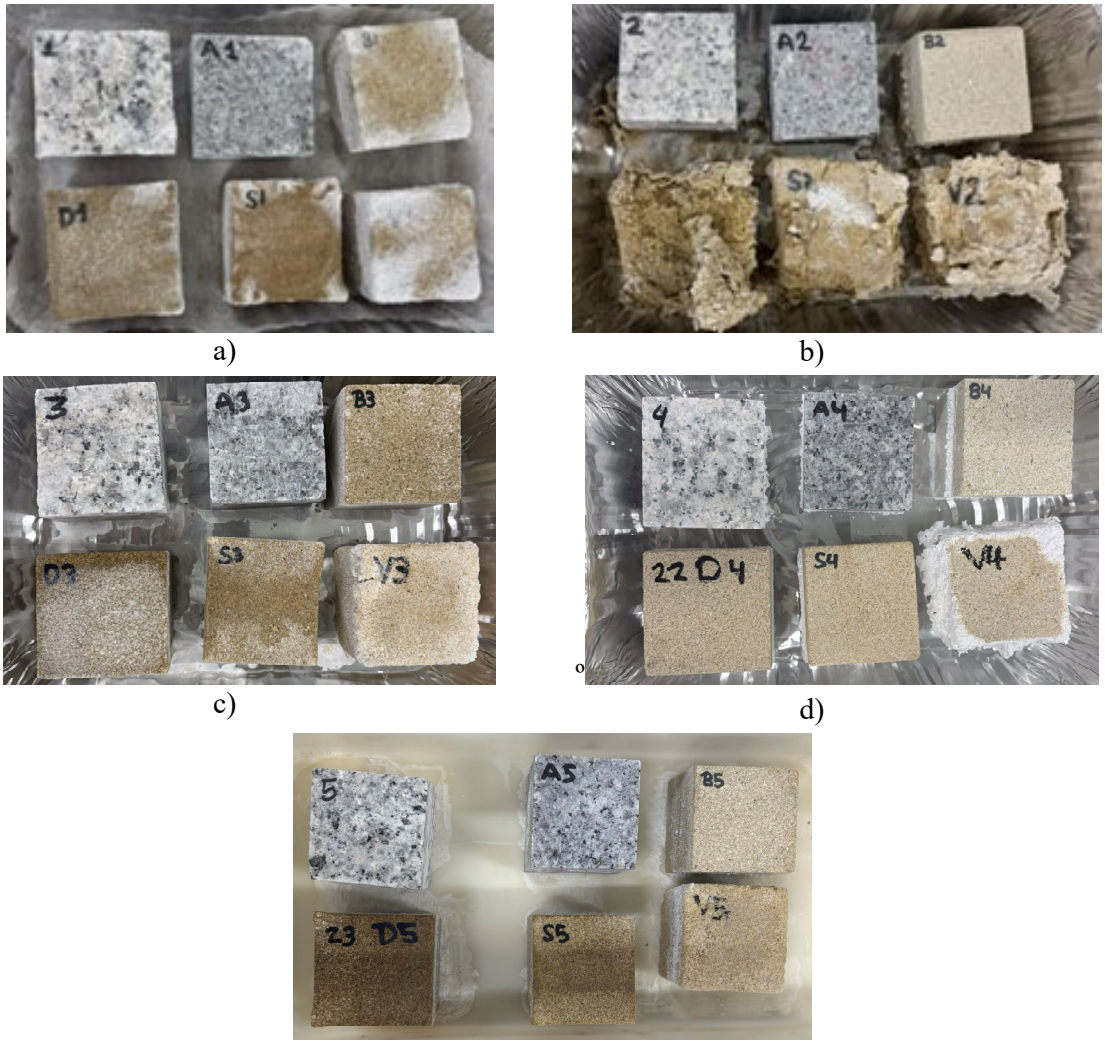


Figura 4-3. Muestras tras el cuarto ciclo de contaminación con sales solubles
a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na_2SO_4
c) Muestras contaminadas con $NaNO_3$; d) Muestras contaminadas con $MgSO_4$
e) Muestras contaminadas con NaCl
(Elaboración propia)

Durante el quinto ciclo, el proceso de deterioro se intensifica de manera notable en los materiales más porosos. Las areniscas presentan una evolución clara hacia formas más desarrolladas de eflorescencias, destacando especialmente la arenisca Dorada Urbión, donde se aprecia una acumulación significativa de sales con inicio de disgregación superficial. La arenisca de Villamayor muestra también un incremento considerable en la cantidad de depósitos, con zonas donde la cohesión superficial comienza a verse comprometida. En la arenisca de Sierra de la Demanda se observa una progresión homogénea del fenómeno, aunque menos agresiva. La caliza Bateig incrementa la presencia de depósitos, aunque sin evidenciar aún daños estructurales importantes. Los granitos continúan mostrando una alta resistencia al deterioro, limitándose los efectos a pequeñas acumulaciones superficiales sin pérdida de material. Este ciclo evidencia claramente la influencia de la porosidad en la susceptibilidad al daño por sales (ver Figura 4-4)



a) b)

c) d)

e)

Figura 4-4. Muestras tras el quinto ciclo de contaminación con sales solubles
a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na_2SO_4
c) Muestras contaminadas con $NaNO_3$; d) Muestras contaminadas con $MgSO_4$
e) Muestras contaminadas con NaCl
(Elaboración propia)

En el sexto ciclo se consolida el deterioro en los materiales más vulnerables, apreciándose ya fenómenos de disgregación más avanzados. La arenisca Dorada Urbión presenta una degradación severa, con pérdida evidente de material y formación de estructuras descohesionadas, lo que indica la acción de criptoflorescencias en el interior del material. La arenisca de Villamayor sigue una evolución similar, aunque con menor intensidad, mostrando zonas con acumulación masiva de sales y comienzo de desmoronamiento. La arenisca de Sierra de la Demanda mantiene un comportamiento más estable, con crecimiento de eflorescencias pero sin una pérdida estructural tan acusada. En la caliza Bateig contaminada con sulfato de sodio se observa un comportamiento singular, caracterizado por la aparición de un corte prácticamente recto que provoca una rotura súbita del material; este fenómeno se asocia a la cristalización interna de las sales, que genera elevadas tensiones en planos de debilidad y desencadena un fallo estructural brusco y de gran magnitud. Los granitos, una vez más, apenas muestran alteraciones, confirmando su baja porosidad y elevada resistencia frente a estos procesos (ver Figura 4-5).

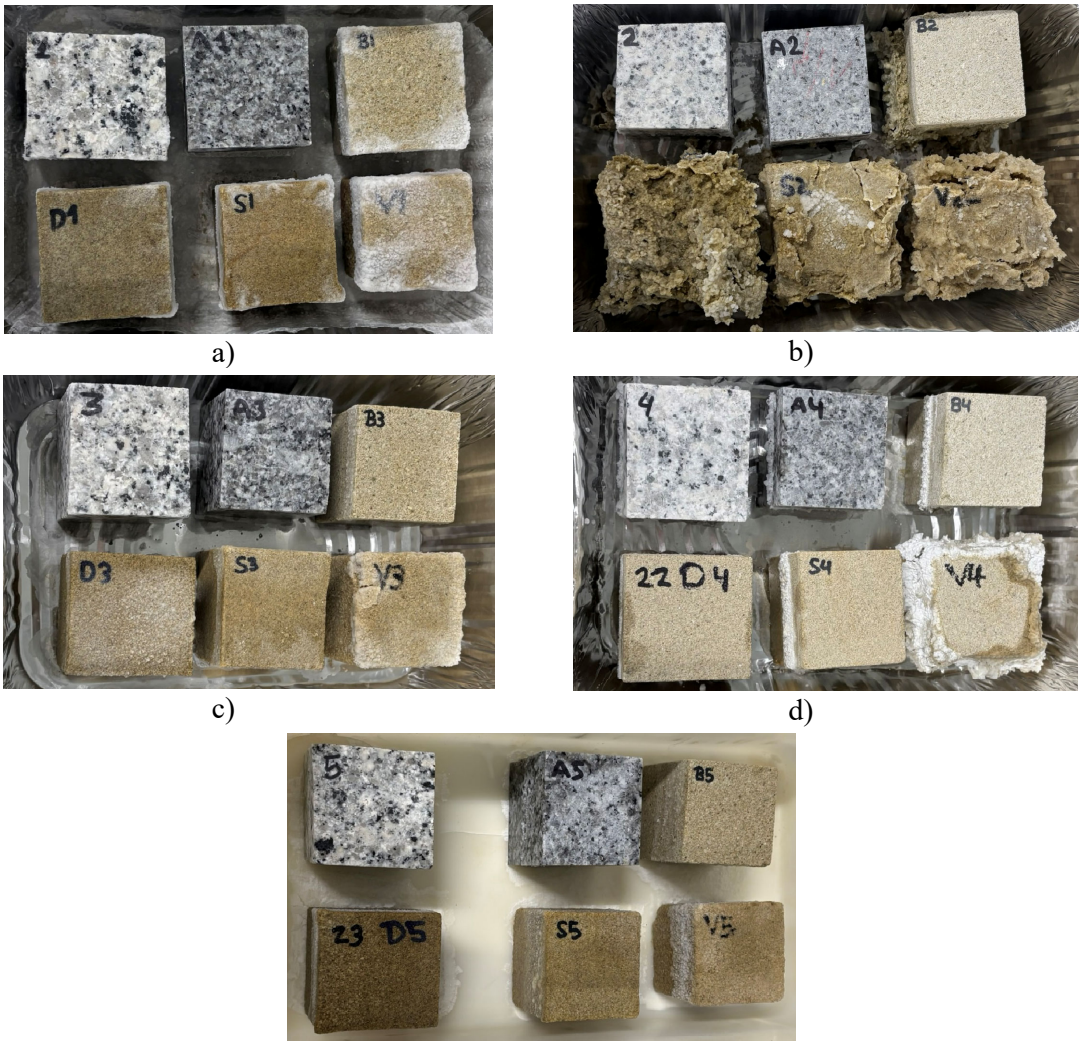


Figura 4-5. Muestras tras el sexto ciclo de contaminación con sales solubles
a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na_2SO_4
c) Muestras contaminadas con $NaNO_3$; d) Muestras contaminadas con $MgSO_4$
e) Muestras contaminadas con NaCl
(Elaboración propia)

En el séptimo ciclo se observa una evolución diferenciada entre materiales, acentuándose las tendencias previamente identificadas. La arenisca Dorada Urbión alcanza un estado avanzado de degradación, con una disgregación generalizada que afecta a gran parte del volumen de la muestra. La arenisca de Villamayor presenta un incremento en la acumulación de sales y un deterioro progresivo de su cohesión, aunque sin alcanzar el grado extremo observado en la anterior. La arenisca de Sierra de la Demanda continúa mostrando un comportamiento más moderado, con eflorescencias visibles pero controladas. La caliza Bateig evidencia un aumento en los depósitos superficiales, especialmente en forma de costras en los bordes, aunque mantiene su integridad estructural. Los granitos siguen mostrando una notable estabilidad, con alteraciones prácticamente despreciables (ver Figura 4-6).

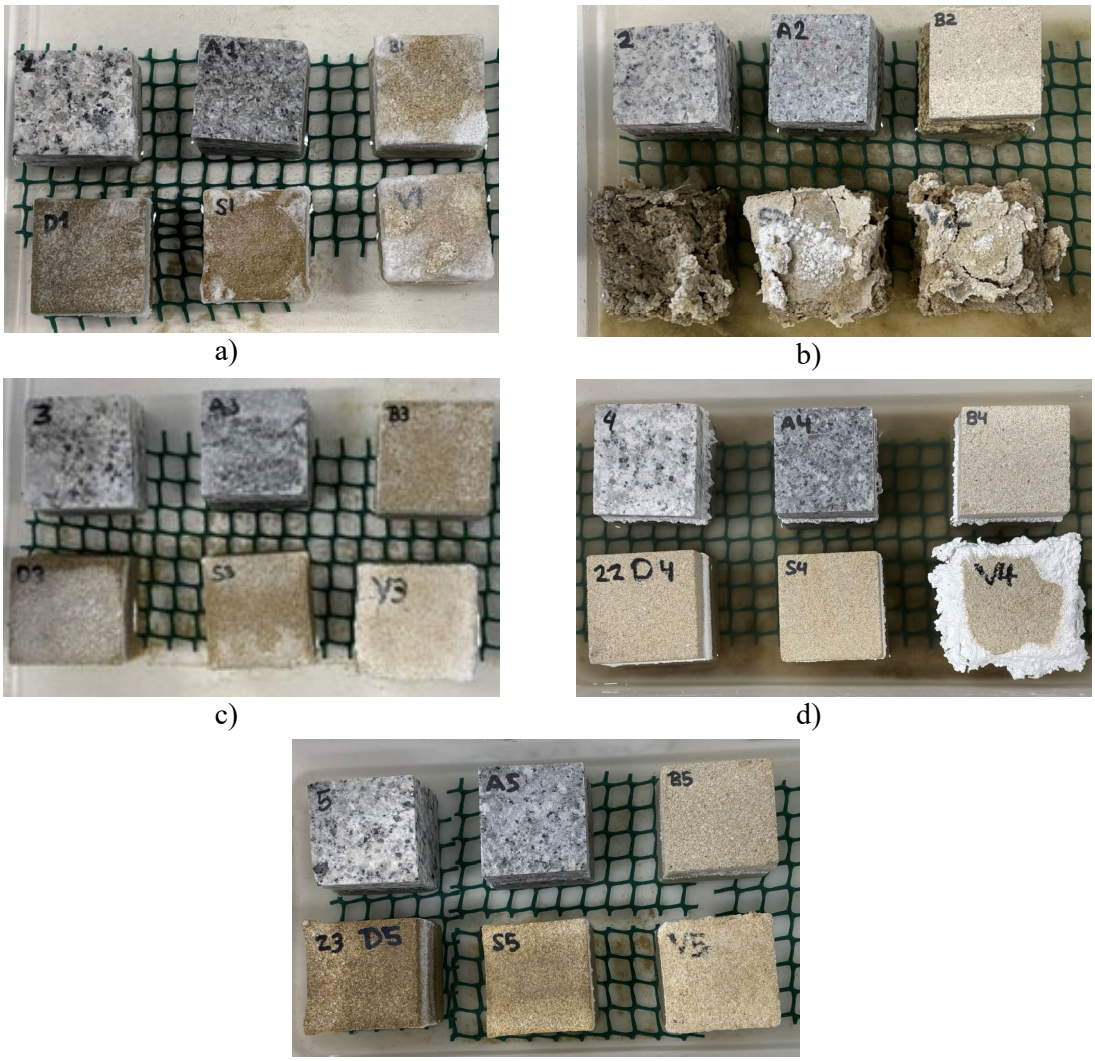


Figura 4-6. Muestras tras el séptimo ciclo de contaminación con sales solubles
a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na_2SO_4
c) Muestras contaminadas con $NaNO_3$; d) Muestras contaminadas con $MgSO_4$
e) Muestras contaminadas con NaCl
(Elaboración propia)

En el octavo ciclo se alcanza el máximo grado de deterioro observable en las muestras más sensibles, sobre todo con Na_2SO_4 como sal contaminante seguido de $MgSO_4$ y, a la par, KCl y $NaNO_3$ y por último con NaCl. La arenisca Dorada Urbión presenta una descomposición prácticamente total de su estructura superficial, con una pérdida masiva de cohesión que evidencia la acción continuada de la cristalización interna de sales. La arenisca de Villamayor muestra también un estado avanzado de degradación, con importantes acumulaciones salinas y zonas claramente desagregadas. La arenisca de Sierra de la Demanda, aunque afectada, mantiene una mayor estabilidad relativa, con depósitos superficiales pero sin colapso estructural significativo. La caliza Bateig presenta un desarrollo notable de eflorescencias, especialmente en bordes y zonas de evaporación, aunque conserva su integridad global. Finalmente, los granitos continúan sin mostrar deterioro relevante, confirmando su idoneidad frente a ambientes salinos. Este ciclo final pone de manifiesto la evolución completa del proceso de alteración, desde la aparición de eflorescencias hasta la pérdida estructural en los materiales más vulnerables (ver Figura 4-7).



a)



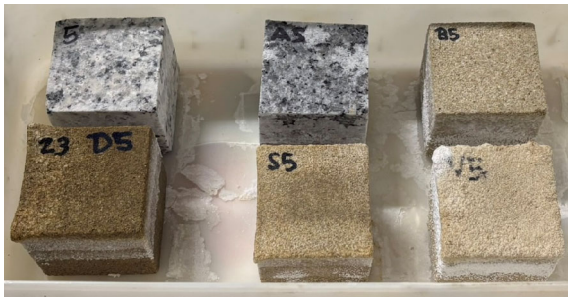
b)



c)



d)



e)

Figura 4-7. Muestras tras el octavo ciclo de contaminación con sales solubles
a) Muestras contaminadas con KCl; b) Muestras contaminadas con Na2SO4
c) Muestras contaminadas con $NaNO_3$; d) Muestras contaminadas con $MgSO_4$
e) Muestras contaminadas con NaCl
(Elaboración propia)

A continuación, se presentan las imágenes correspondientes a las distintas muestras tras su exposición a las disoluciones salinas, donde se muestran, en cada figura, probetas del mismo tipo de material sometidas a diferentes sales, lo que permite realizar una comparación directa del efecto de cada agente sobre un mismo material. De este modo, se puede apreciar de forma cualitativa cuál de las sales produce una mayor o menor afección en cada caso. En general, se observa que los materiales más compactos presentan una menor alteración, manteniendo su integridad superficial, mientras que los materiales más porosos experimentan un deterioro más acusado, evidenciado por la aparición de eflorescencias, pérdida de cohesión y desagregación. Asimismo, se aprecian diferencias en la intensidad y morfología de los depósitos salinos en función del tipo de sal empleada, lo que pone de manifiesto su influencia en los procesos de degradación. (ver desde la Figura 4-8 hasta la Figura 4-13)

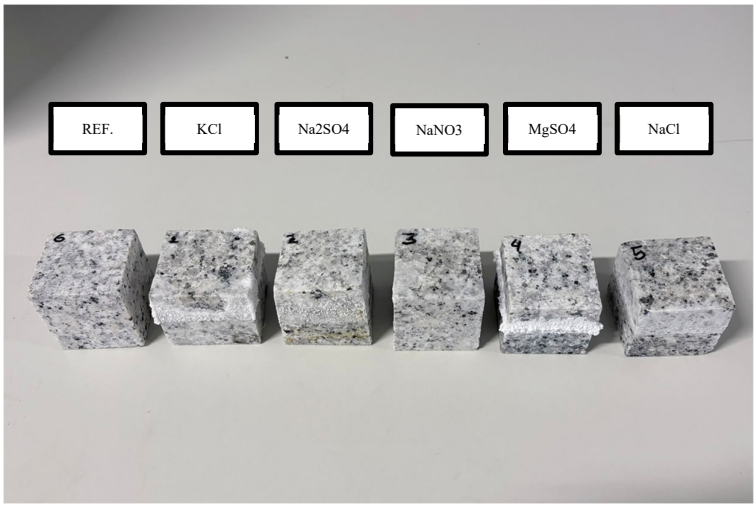


Figura 4-8. Muestras de Granito de Madrid contaminadas con las distintas soluciones salinas (Elaboración propia)

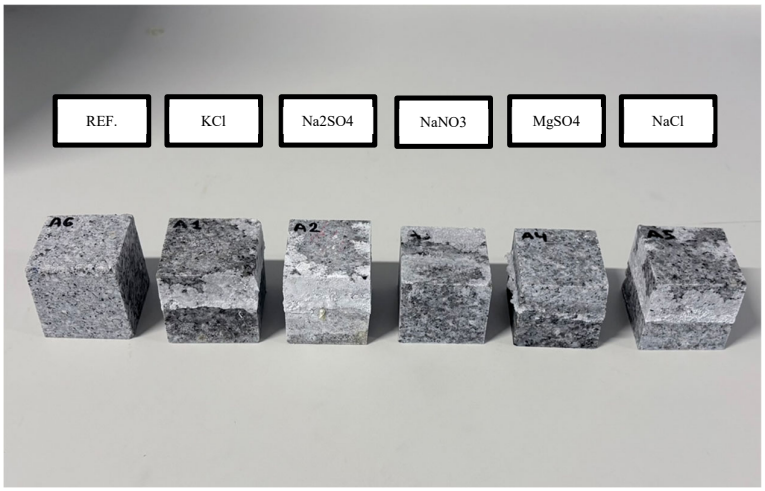


Figura 4-9. Muestras de Granito Azul Alpendorada contaminadas con las distintas soluciones salinas (Elaboración propia)

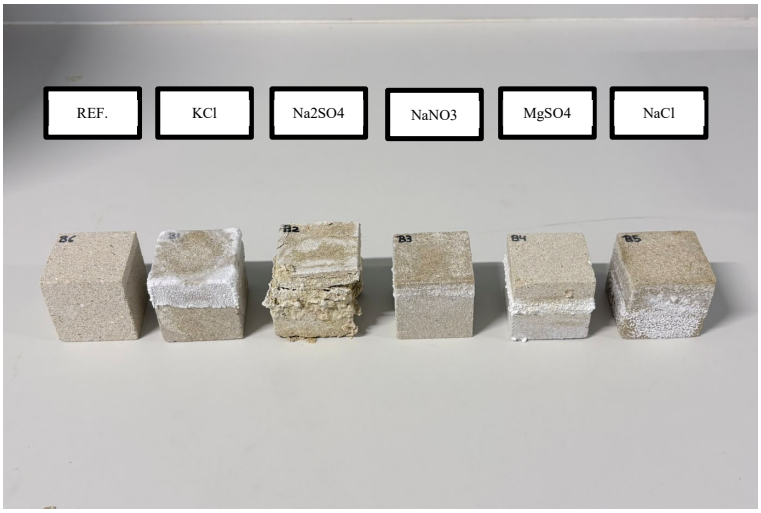


Figura 4-10. Muestras de Caliza Bateig contaminadas con las distintas soluciones salinas (Elaboración propia)

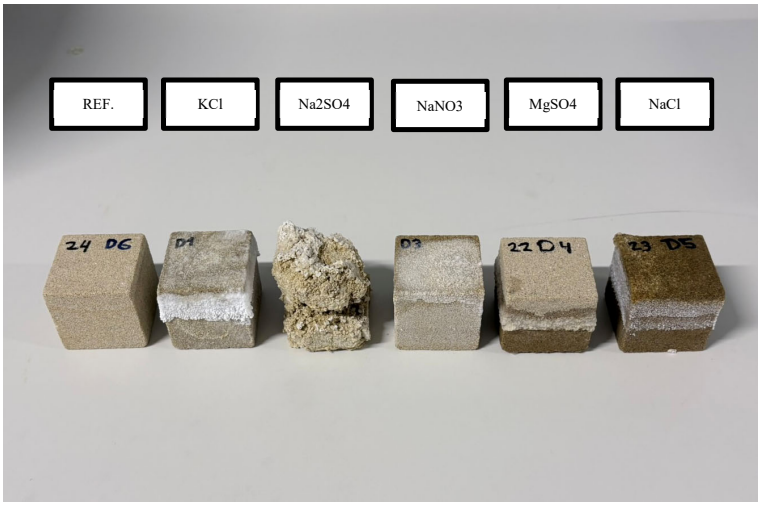


Figura 4-11. Muestras de Arenisca Dorada Urbión contaminadas con las distintas soluciones salinas (Elaboración propia)

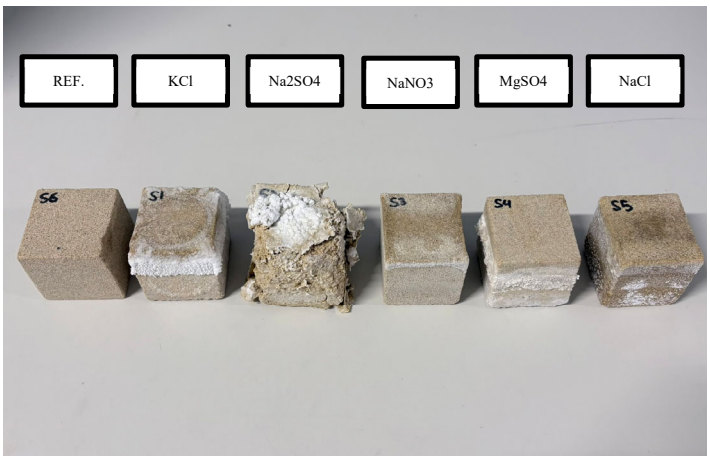


Figura 4-12. Muestras de Arenisca Sierra de la Demanda contaminadas con las distintas soluciones salinas (Elaboración propia)

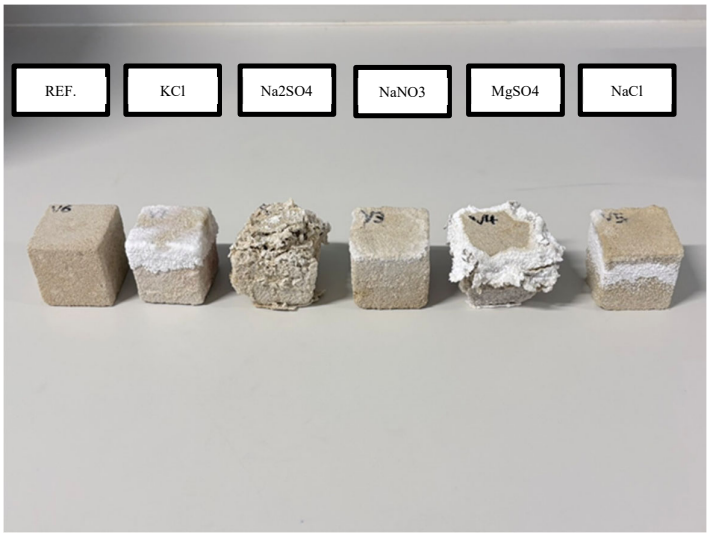


Figura 4-13. Muestras de Arenisca Villanueva contaminadas con las distintas soluciones salinas (Elaboración propia)

4.3 Análisis termográfico

En el presente apartado, dedicado al análisis termográfico, se ha procedido a la recopilación y tratamiento de los datos térmicos obtenidos a partir de las grabaciones realizadas durante los ciclos de calentamiento y enfriamiento de las diferentes muestras estudiadas. Dichas grabaciones presentan una duración aproximada de dos horas, intervalo temporal que permite observar con suficiente resolución la evolución térmica de los materiales sometidos a las distintas condiciones experimentales.

A partir de estas grabaciones, se han extraído los datos numéricos en formato *.csv*, lo que posibilita trabajar con un elevado volumen de información térmica de alta precisión. En las representaciones obtenidas, la disposición de las muestras sigue un orden sistemático: en el extremo izquierdo se sitúa la muestra 6, empleada como muestra de referencia, la cual no presenta contaminación por sales. A continuación, se muestran sucesivamente las muestras contaminadas con KCl, Na_2SO_4, $NaNO_3$, $MgSO_4$ y NaCl, manteniendo este mismo orden para cada uno de los materiales analizados: las muestras de granito, la muestra de caliza y las tres muestras de arenisca.

Con el objetivo de ilustrar el procedimiento seguido para la adquisición y procesamiento de los datos, se incluyen a continuación capturas de pantalla del software FLIR Tools+, herramienta mediante la cual se han analizado las grabaciones termográficas y se ha realizado la exportación de los datos térmicos al formato *.csv*. Este procedimiento permite disponer de una base de datos extensa y estructurada, adecuada para su posterior utilización en los modelos de aprendizaje profundo (deep learning) empleados.

En conclusión, se presenta una representación gráfica en cada caso de estudio de:

1) la evolución de la temperatura media de las muestras a lo largo del tiempo, acompañada de su correspondiente intervalo de confianza (media ± desviación típica).

2) la evolución de la suma temperatura media más la desviación típica de las muestras a lo largo del tiempo (media + desviación típica).

3) la evolución de la diferencia de temperatura media menos la desviación típica de las muestras a lo largo del tiempo (media - desviación típica).

siguien

Con ello, se permite evaluar de manera comparativa el comportamiento térmico de cada tipo de muestra contaminada frente a la muestra de referencia. Esta visualización facilita la identificación de posibles divergencias en la dinámica térmica inducidas por la presencia de las distintas sales, constituyendo una base cuantitativa para el análisis posterior mediante las técnicas avanzadas de modelado seleccionadas. Para los ciclos de enfriamiento véase desde la Figura 4-14 hasta la Figura 4-25. Para los ciclos de calentamiento véase desde la Figura 4-26 hasta la Figura 4-37.

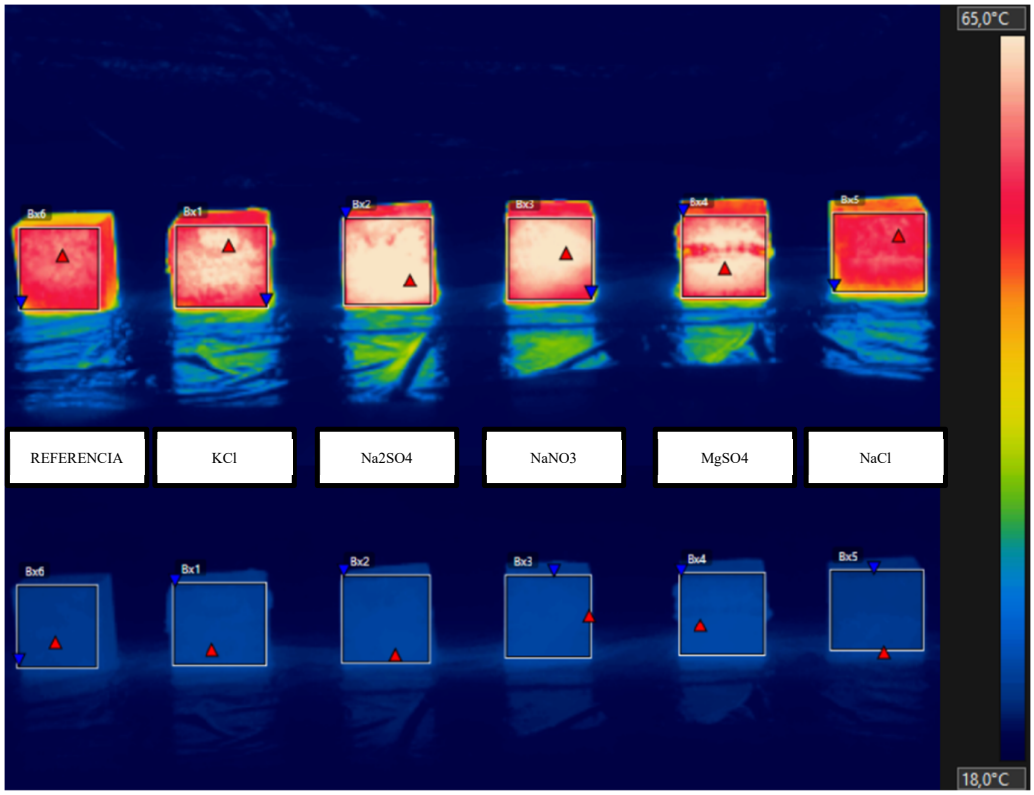


Figura 4-14. Captura inicial y final del enfriamiento de las muestras de Granito de Madrid (Elaboración propia)

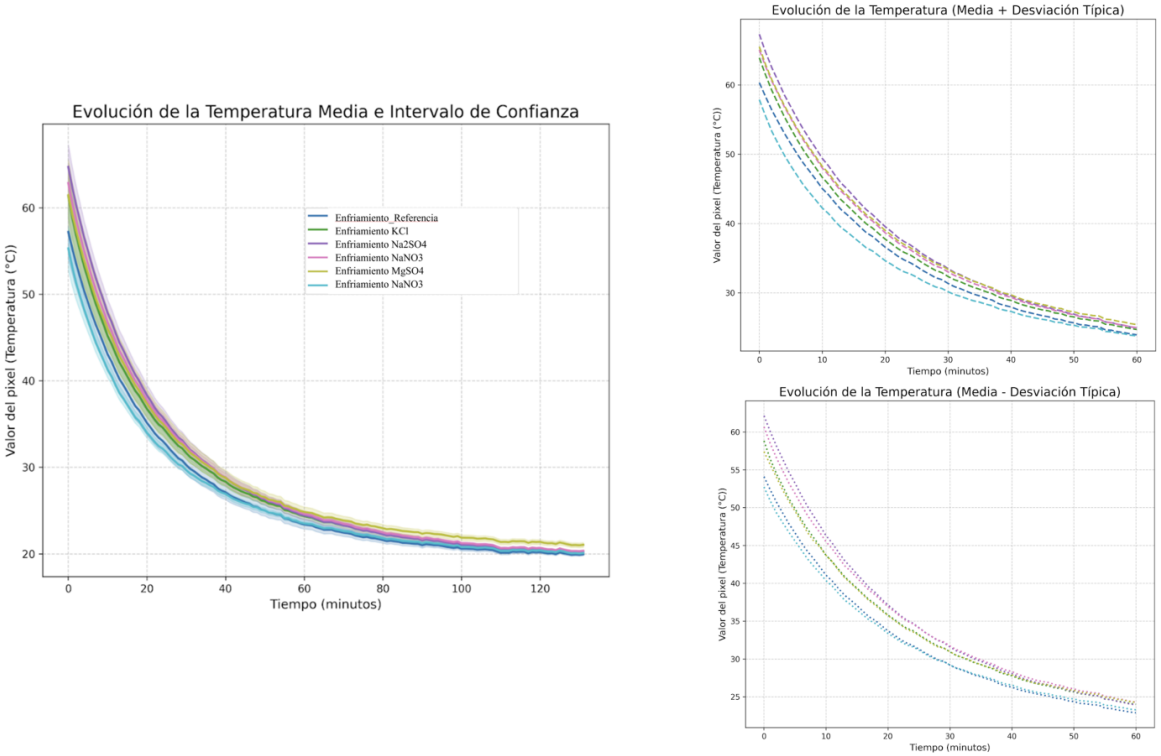


Figura 4-15. Representaciones gráficas del enfriamiento de las muestras de Granito de Madrid (Elaboración propia)

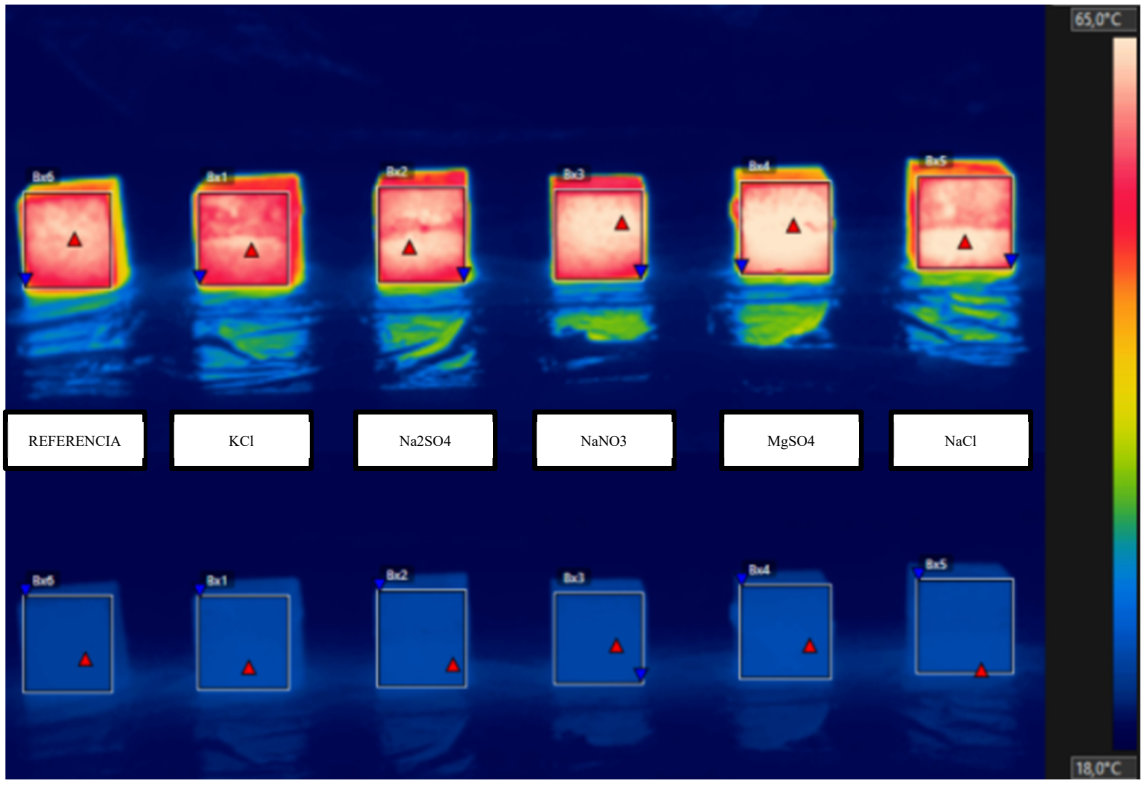


Figura 4-16. Captura inicial y final del enfriamiento de las muestras de Granito Azul Alpendorada (Elaboración propia)

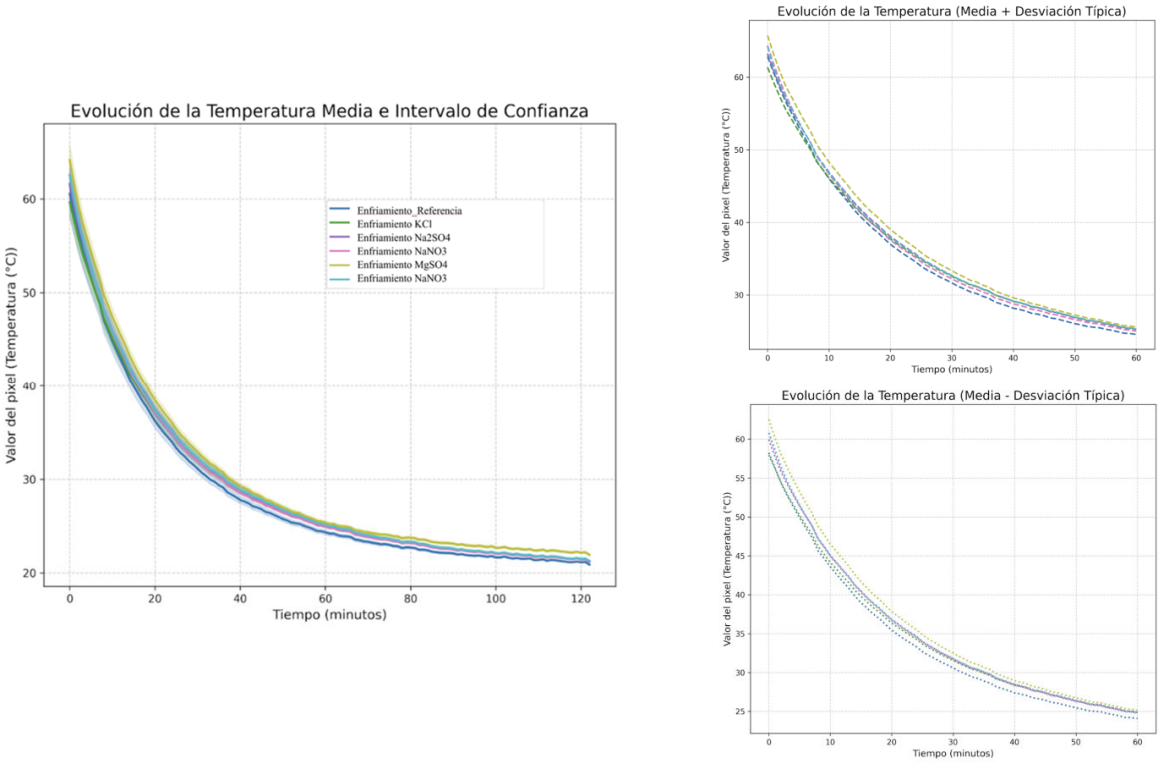


Figura 4-17. Representaciones gráficas del enfriamiento de las muestras de Granito Azul Alpendorada (Elaboración propia)

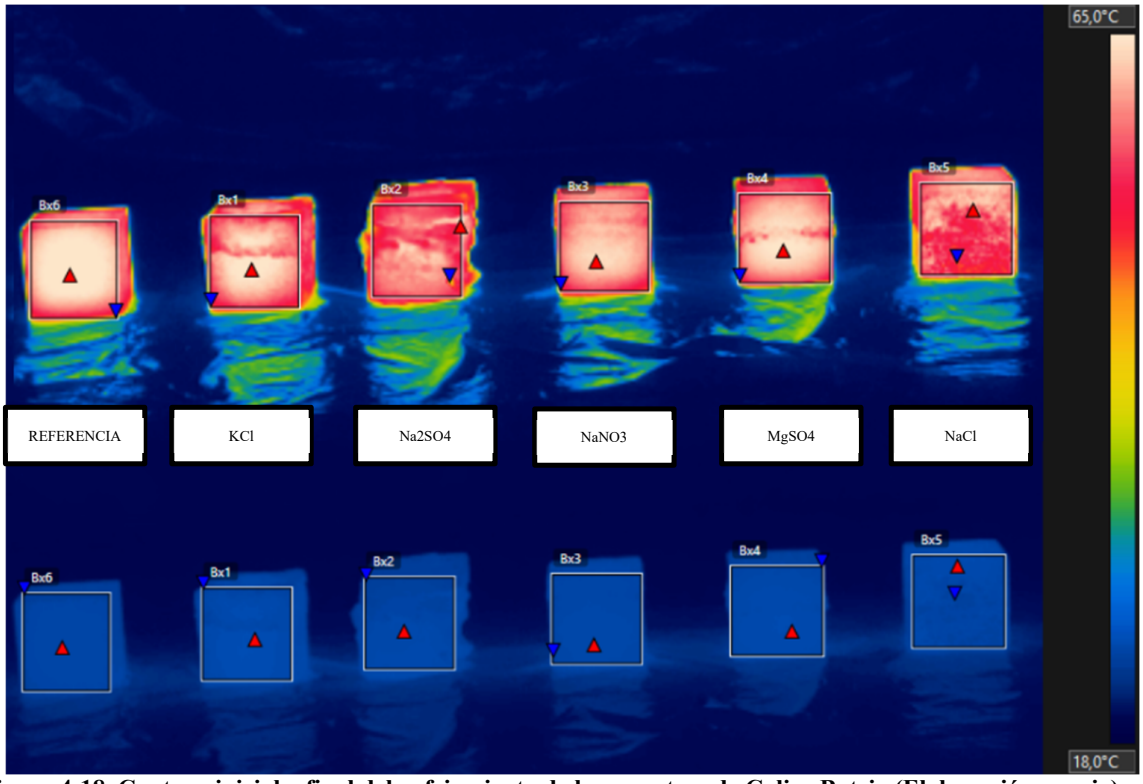


Figura 4-18. Captura inicial y final del enfriamiento de las muestras de Caliza Bateig (Elaboración propia)

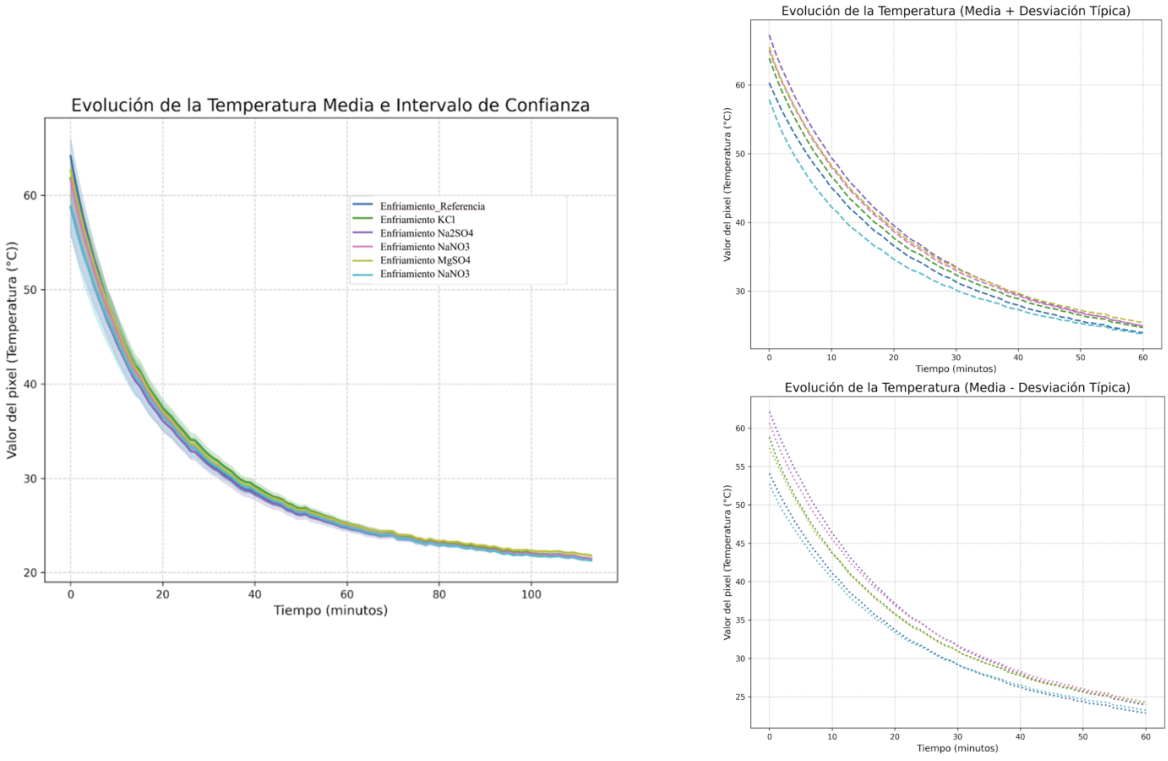


Figura 4-19. Representaciones gráficas del enfriamiento de las muestras de Caliza de Bateig (Elaboración propia)

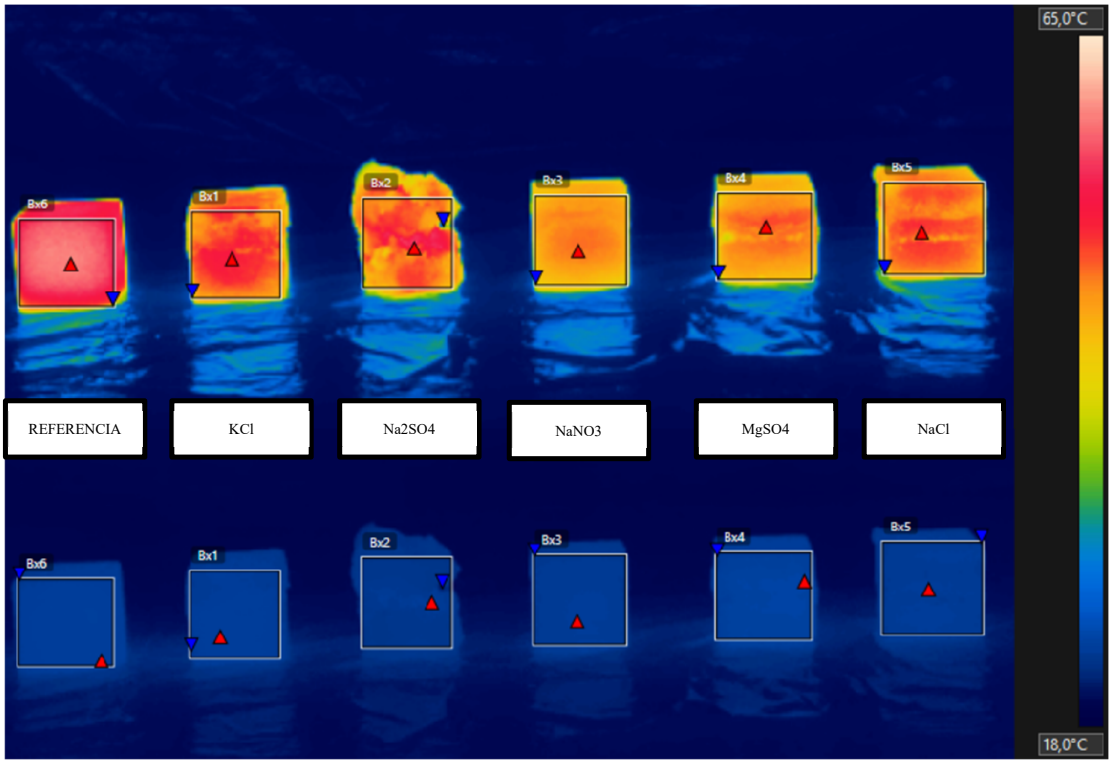


Figura 4-20. Captura inicial y final del enfriamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia)

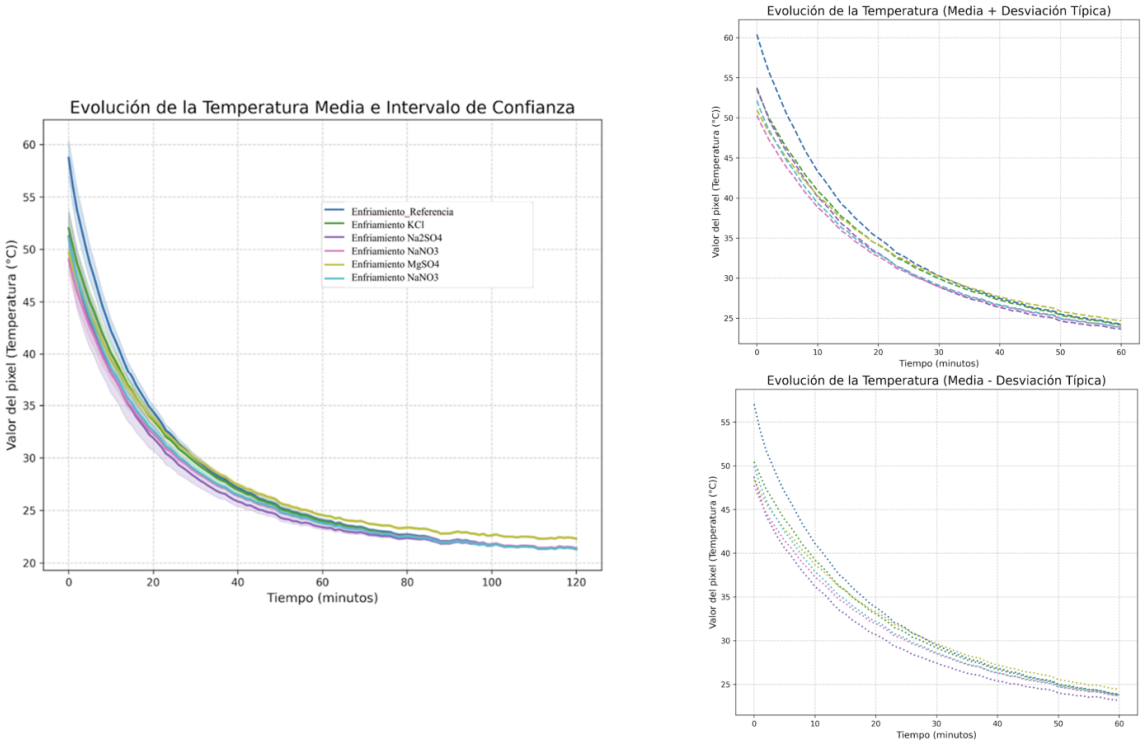


Figura 4-21. Representaciones gráficas del enfriamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia)

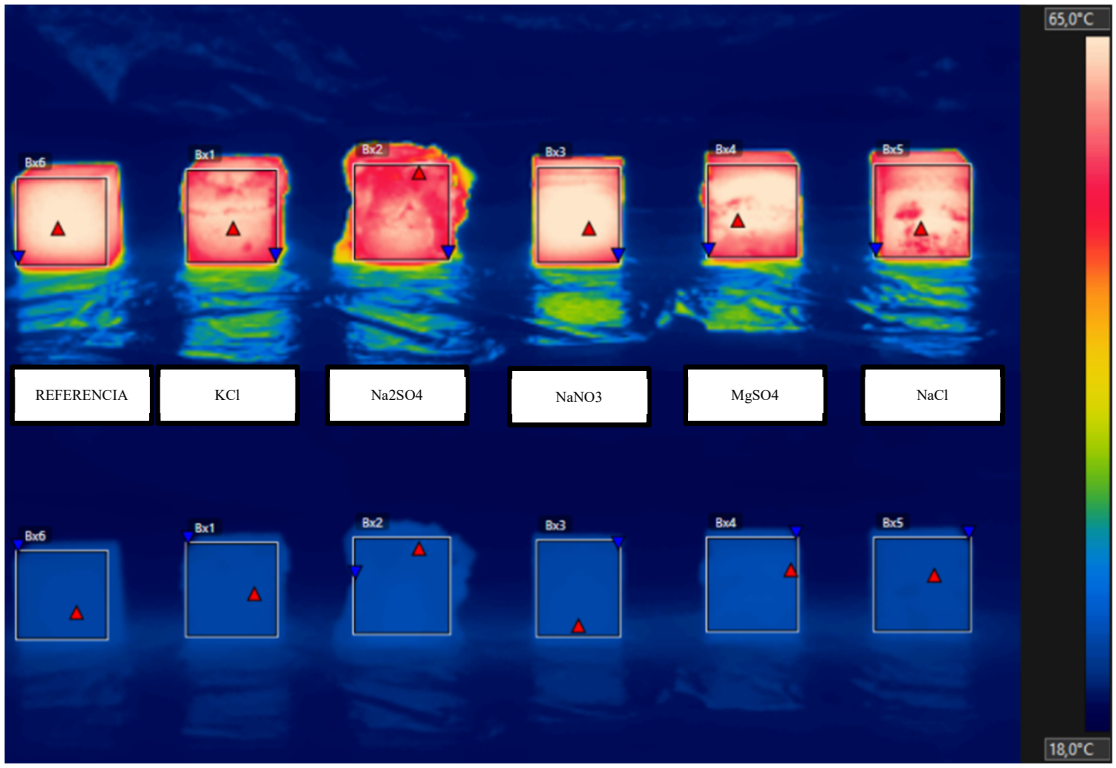


Figura 4-22. Captura inicial y final del enfriamiento de las muestras de Arenisca de Sierra de la Demanda (Elaboración propia)

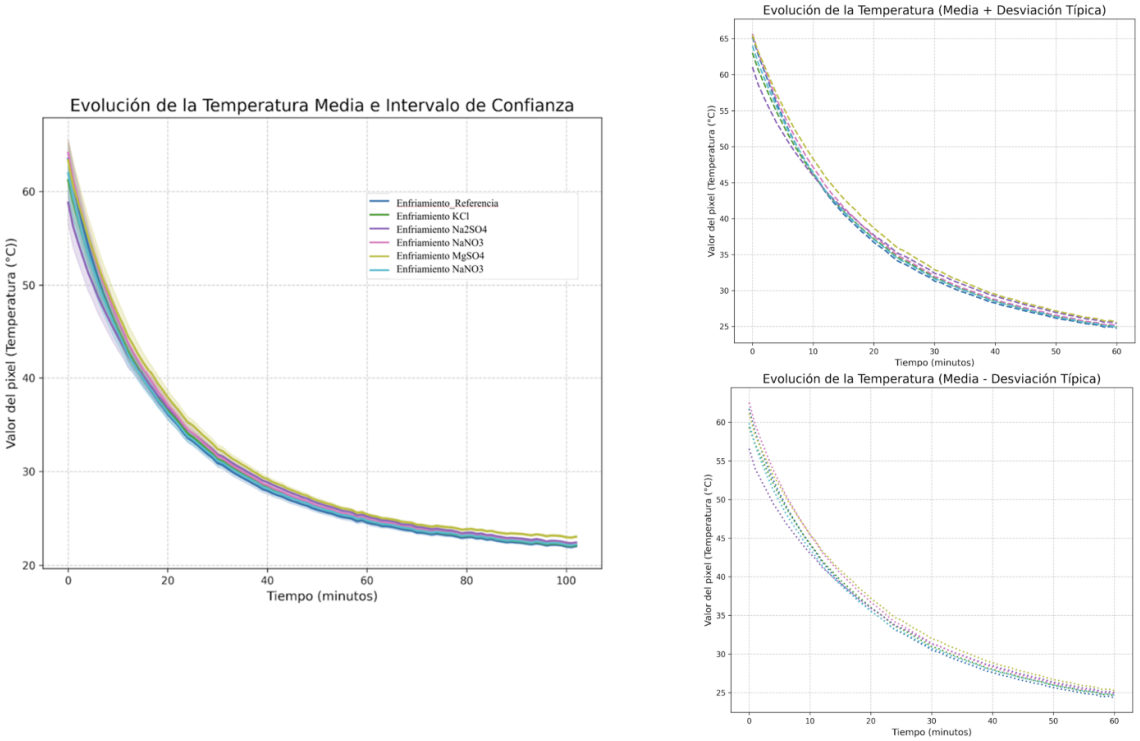


Figura 4-23. Representaciones gráficas del enfriamiento de las muestras de Arenisca de Sierra de la Demanda (Elaboración propia)

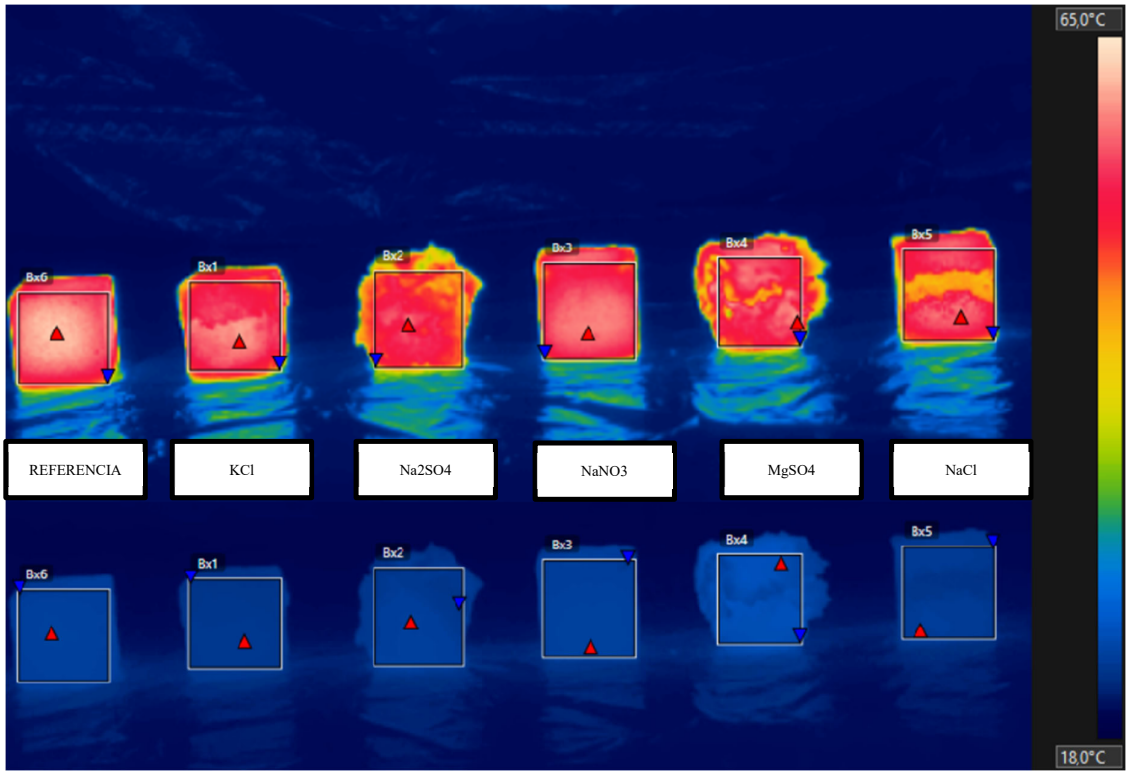


Figura 4-24. Captura inicial y final del enfriamiento de las muestras de Arenisca Villanueva (Elaboración propia)

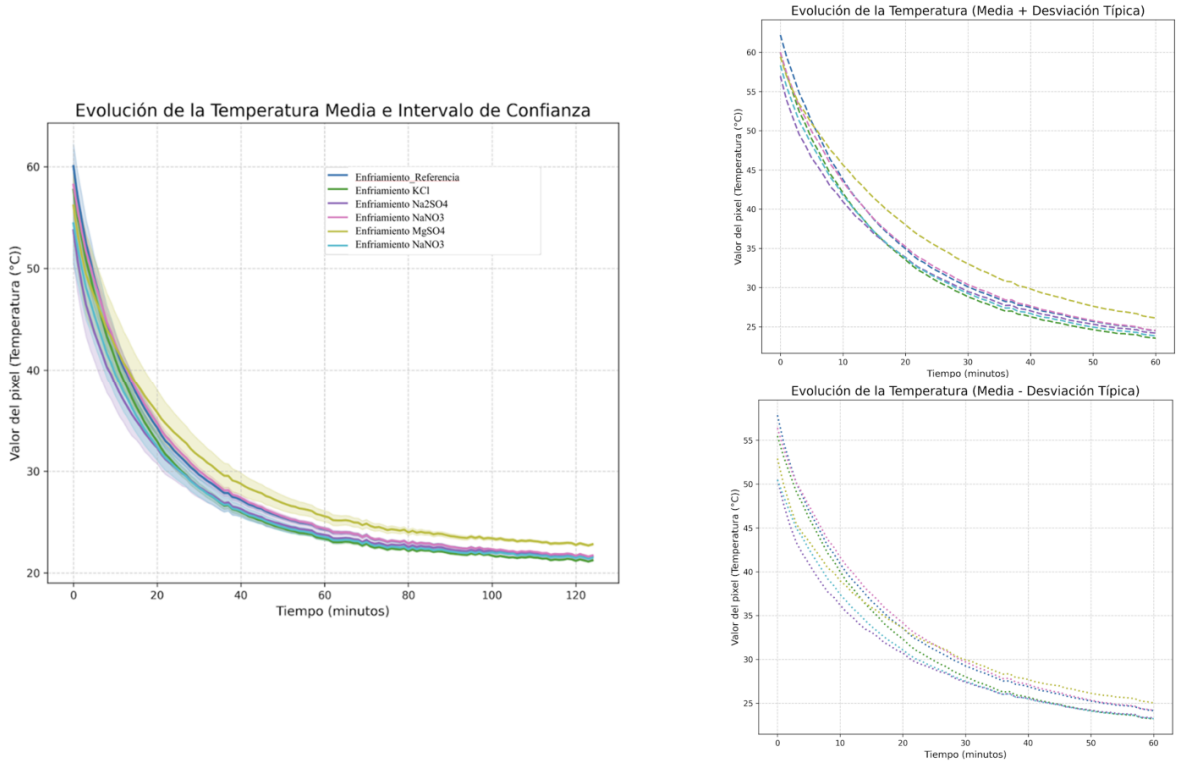


Figura 4-25. Representaciones gráficas del enfriamiento de las muestras de Arenisca Villanueva (Elaboración propia)

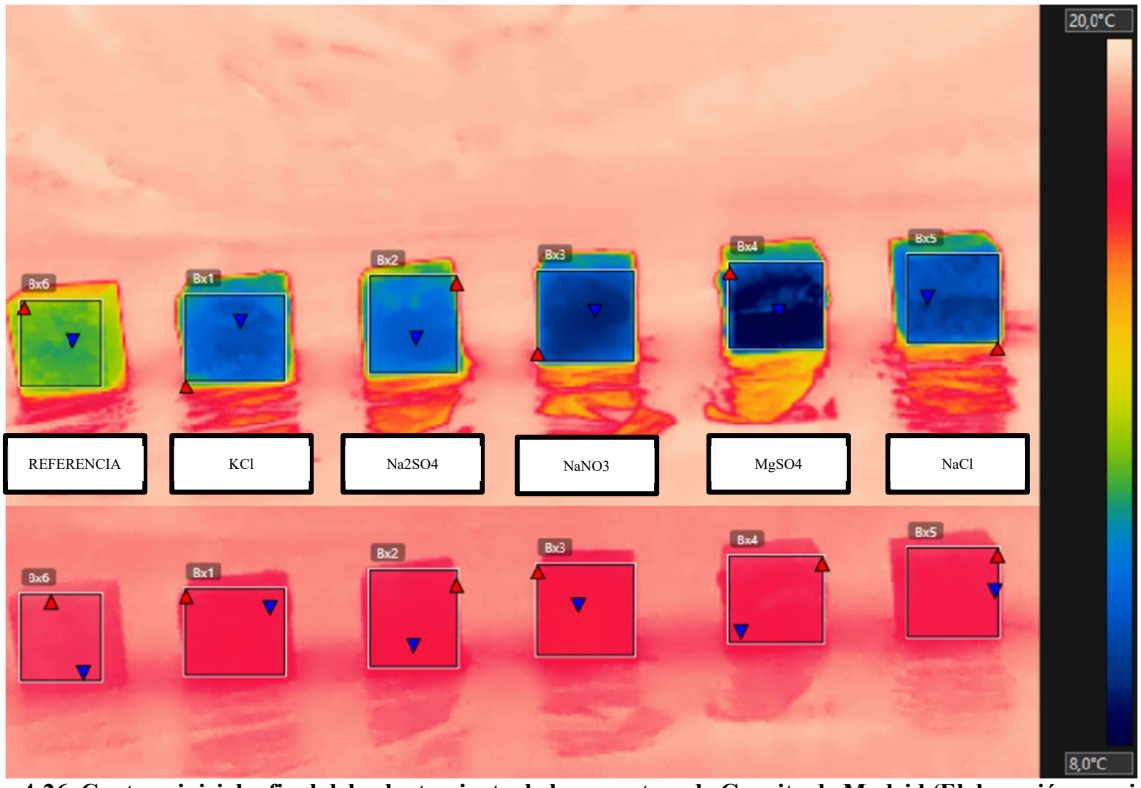


Figura 4-26. Captura inicial y final del calentamiento de las muestras de Granito de Madrid (Elaboración propia)

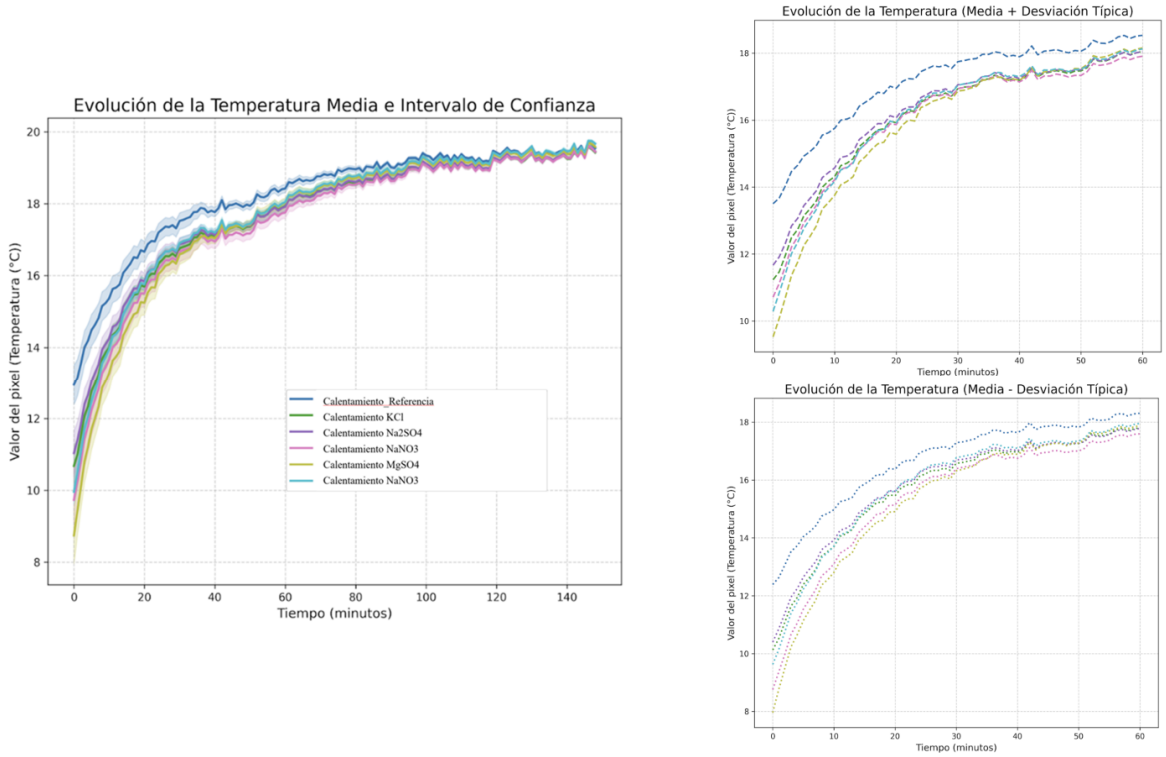


Figura 4-27. Representaciones gráficas del calentamiento de las muestras de Granito de Madrid (Elaboración propia)

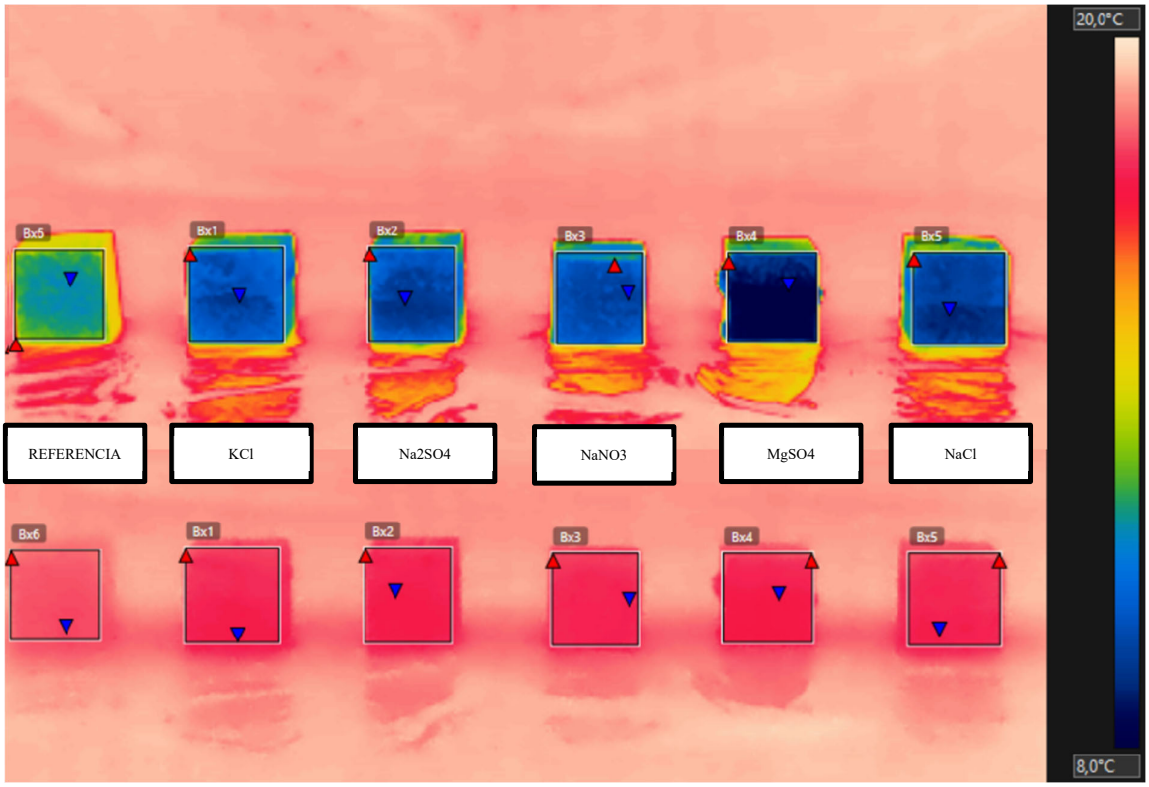


Figura 4-28. Captura inicial y final del calentamiento de las muestras de Granito Azul Alpendorada (Elaboración propia)

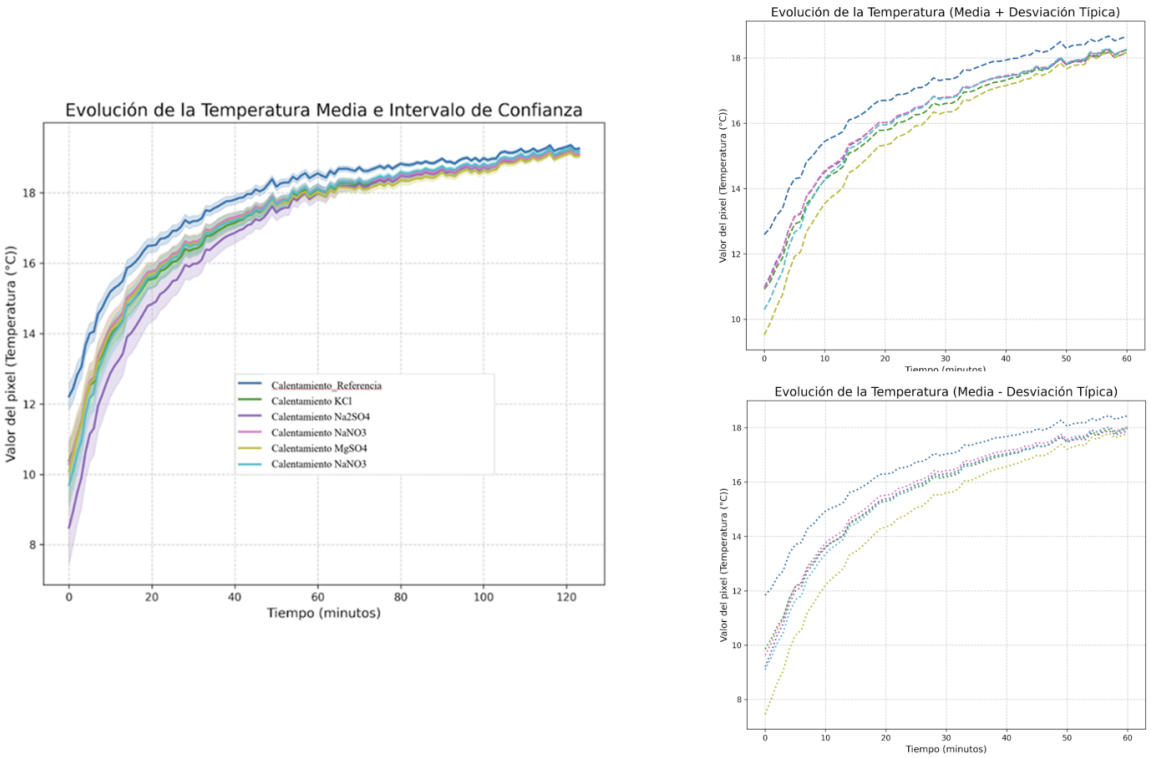


Figura 4-29. Representaciones gráficas del calentamiento de las muestras de Granito Azul Alpendorada (Elaboración propia)

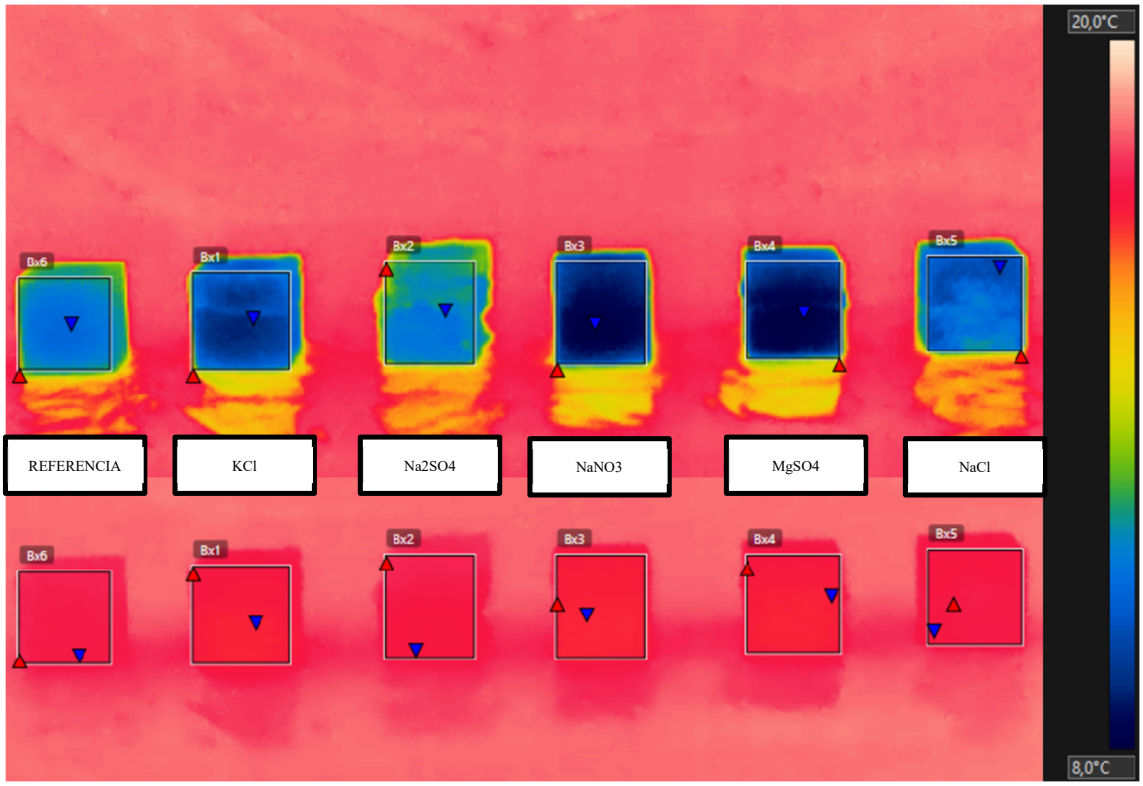


Figura 4-30. Captura inicial y final del calentamiento de las muestras de Caliza Bateig (Elaboración propia)

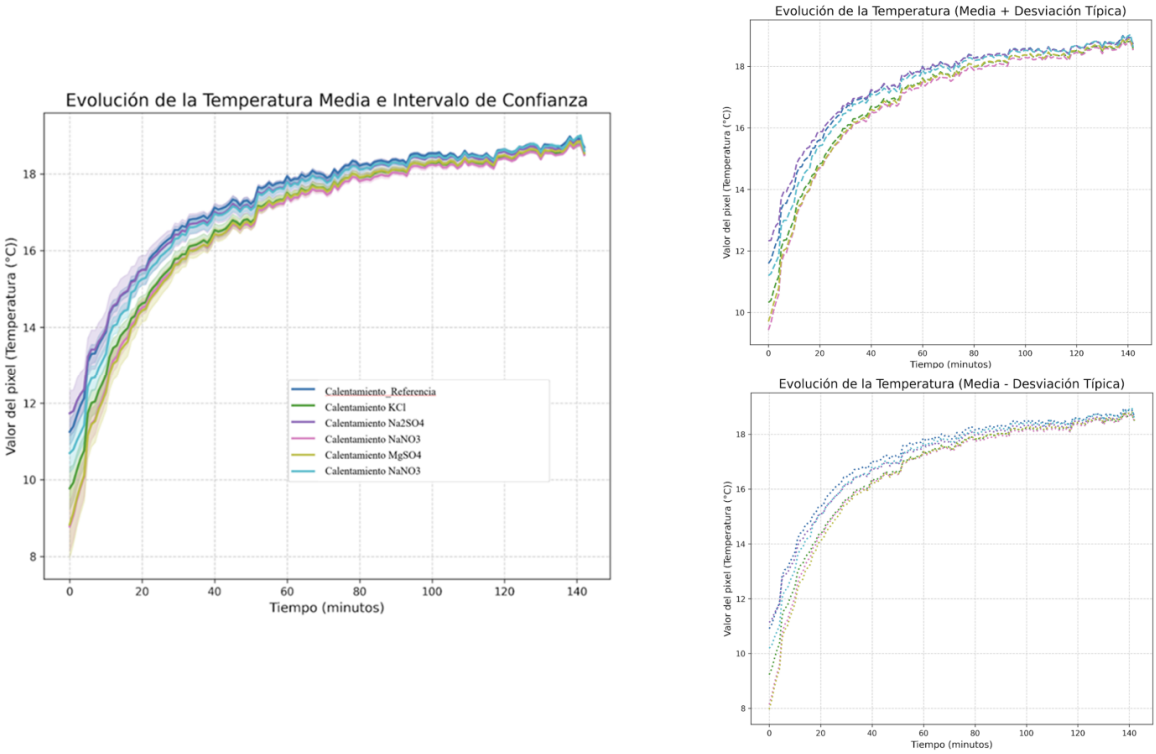


Figura 4-31. Representaciones gráficas del calentamiento de las muestras de Caliza Bateig (Elaboración propia)

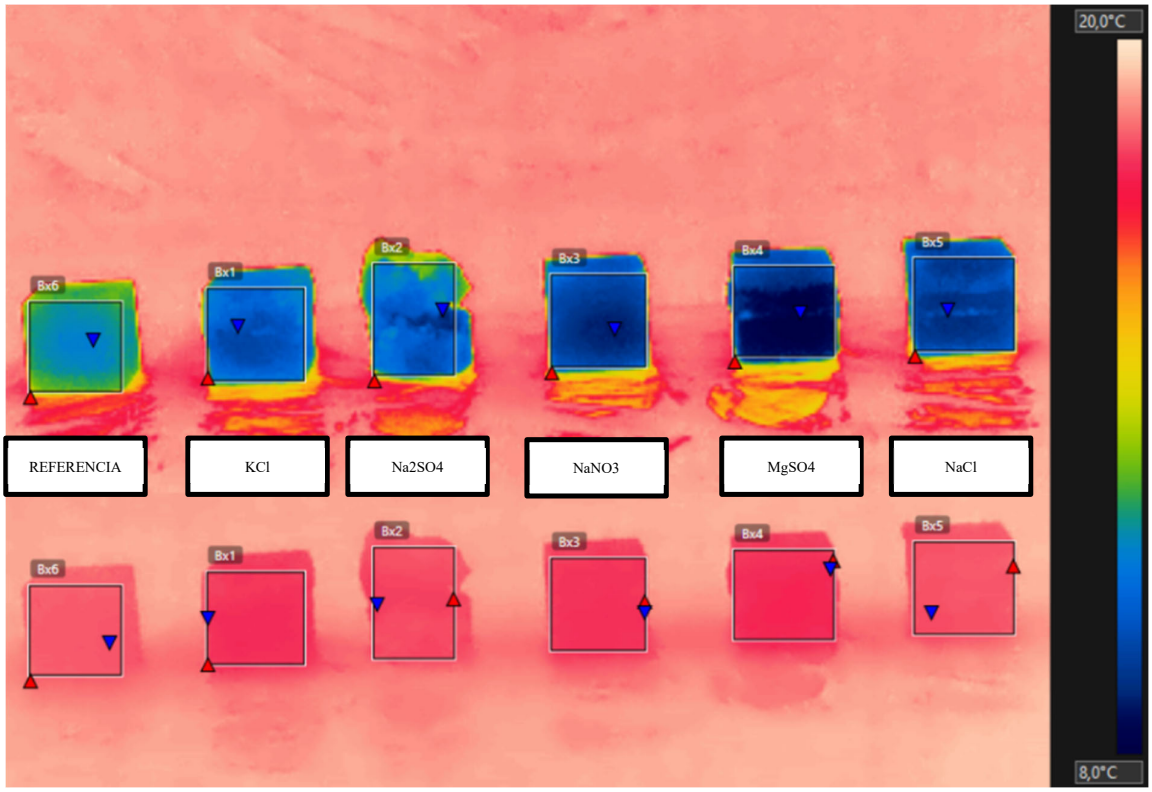


Figura 4-32. Captura inicial y final del calentamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia)

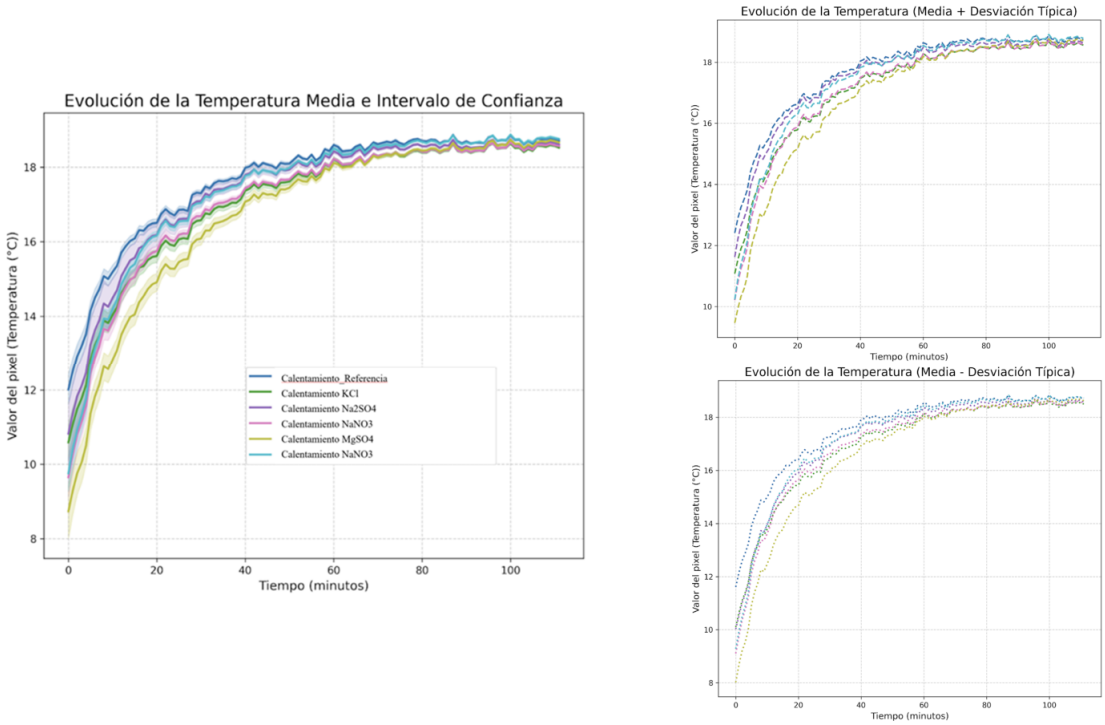


Figura 4-33. Representaciones gráficas del calentamiento de las muestras de Arenisca Dorada Urbión (Elaboración propia)

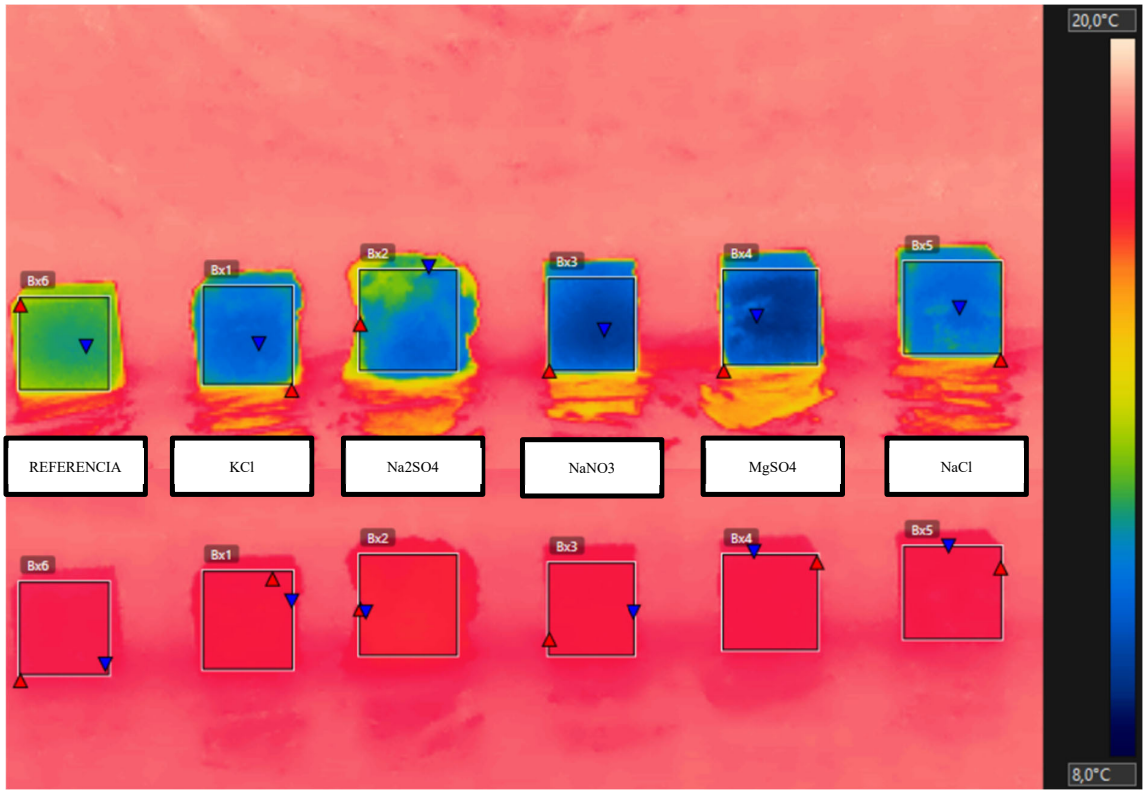


Figura 4-34. Captura inicial y final del calentamiento de las muestras de Arenisca Sierra de la Demanda (Elaboración propia)

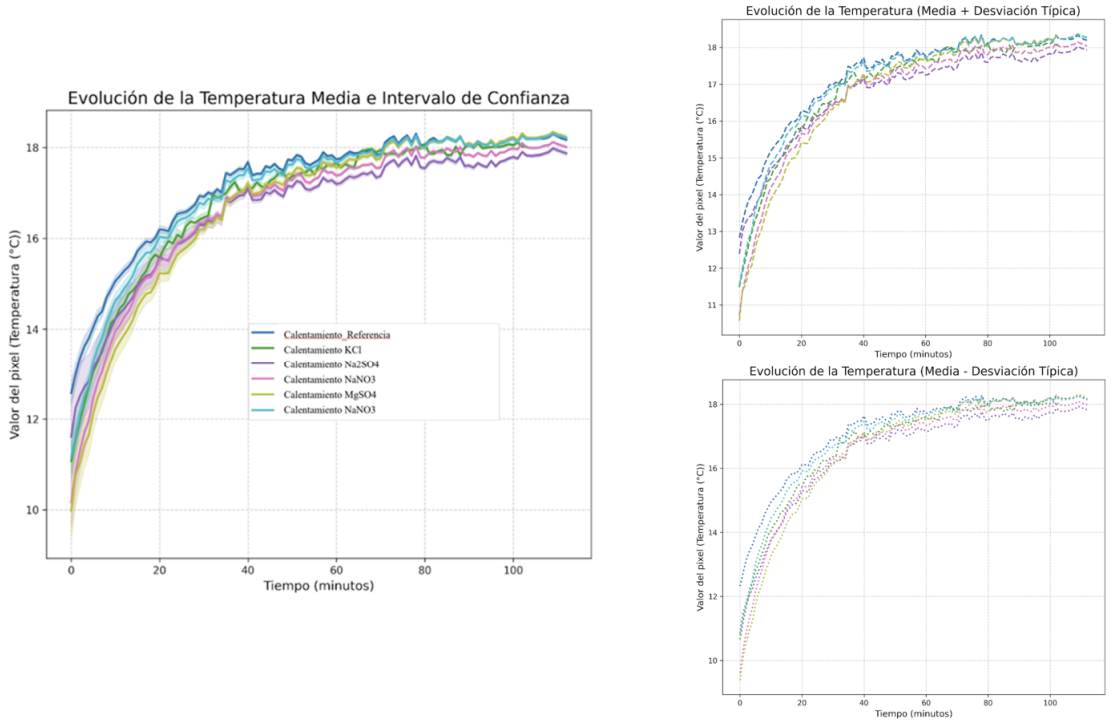


Figura 4-35. Representaciones gráficas del calentamiento de las muestras de Arenisca Sierra de la Demanda (Elaboración propia)

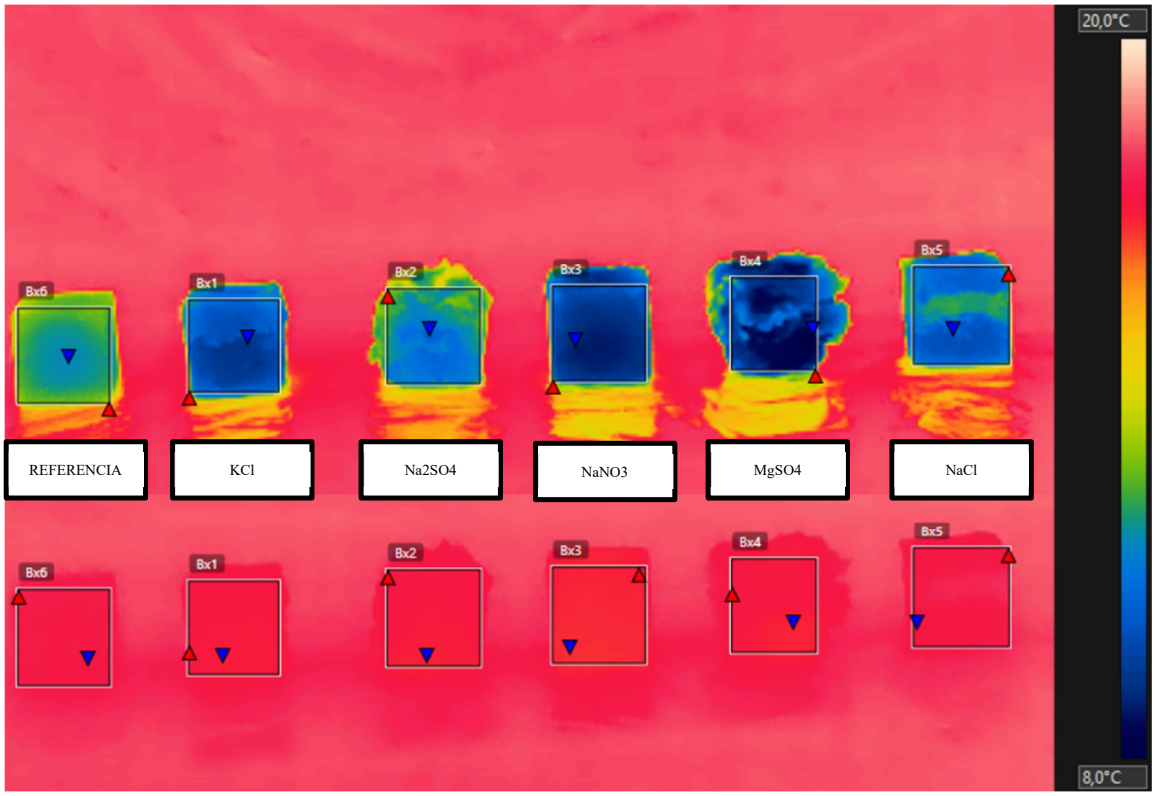


Figura 4-36. Captura inicial y final del calentamiento de las muestras de Arenisca Villanueva (Elaboración propia)

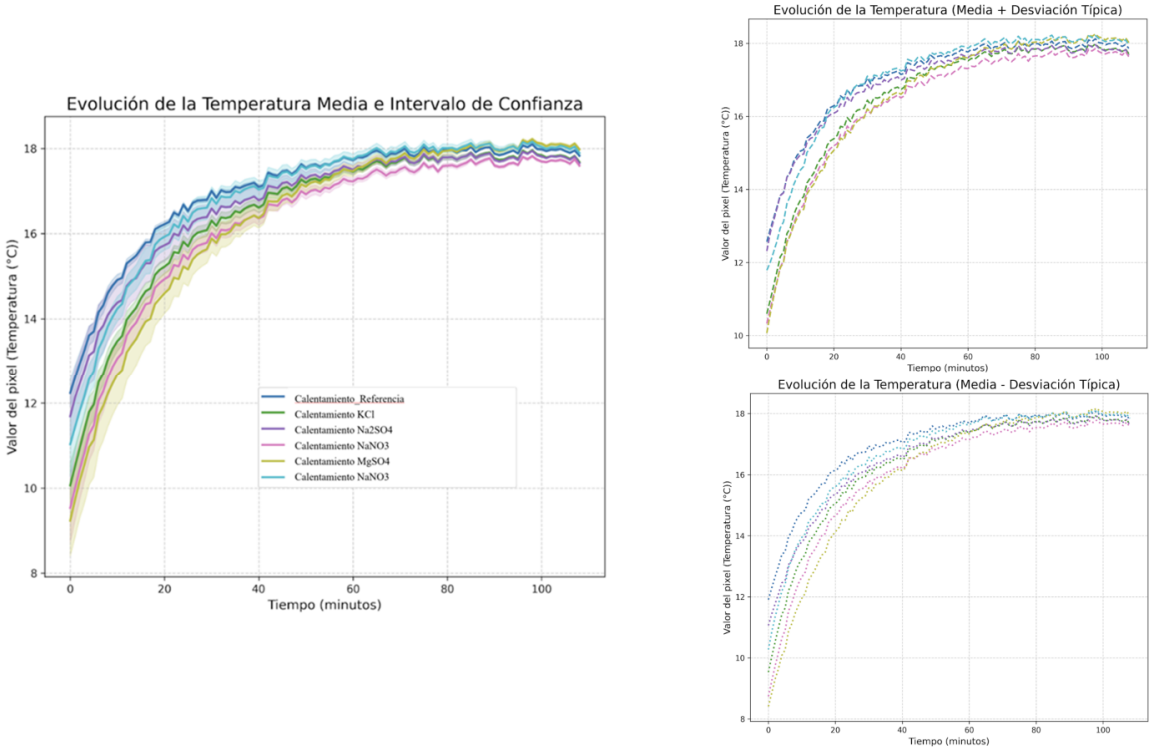


Figura 4-37. Representaciones gráficas del calentamiento de las muestras de Arenisca Villanueva (Elaboración propia)

4.4 Análisis de datos con los modelos *Deep Learning*

4.4.1 Primera etapa. Detección de Materiales

Siguiendo la metodología de preprocesado y modelización descrita, se han evaluado distintos modelos de *deep learning*, los definidos en el apartado 3.5 (RNN, LSTM, GRU), para la clasificación de los materiales a partir de sus curvas térmicas mostradas en el apartado 4.3, transformadas a ratios de temperatura frente a tiempo, definiendo previamente el intervalo de temperatura determinado. Para ello, se ha separado de forma aleatoria 30 de las curvas térmicas para el conjunto de entrenamiento y 6 para el conjunto de validación, tanto para el ensayo de enfriamiento como de calentamiento. Como último escenario, se ha juntado los dos escenarios previos, designando 60 curvas térmicas para el conjunto de entrenamiento y 12 para el los rconjunto de validación.

Los resultados obtenidos muestran un comportamiento altamente satisfactorio en la mayoría de las escenas analizadas, alcanzándose valores máximos de hasta 12 aciertos (100% de aciertos) en el tercer escenario y seleccionado a la vez los 3 parámetros de entrada, es decir, el escenario que abarca tanto las curvas de enfriamiento como las de calentamiento e introduciendo como curvas las de temperatura media, temperatura media más desviación típica y temperatura media menos desviación típica. Este hecho indica una elevada capacidad de discriminación entre materiales e incluso con distintas condiciones, es decir, mismo material con condiciones sanas o con un tipo de sal como contaminante.

En particular, los modelos correspondientes a intervalos temporales de y minutos presentan un rendimiento consistente y estable, independientemente de la configuración empleada, es decir, con o sin diferencia de temperaturas en términos absoluto.

El mejor resultado global se obtiene con el modelo GRU y para el caso de minutos con aplicación de valor absoluto, alcanzando la clasificación completa de las muestras. Este comportamiento refleja una adecuada captación de la dinámica térmica característica de cada material, también reflejado por las gráficas de pérdida y precisión en el conjunto de entrenamiento y de validación a lo largo del aprendizaje, tiendo a cero en la pérdida y a uno en precisión (ver Figura 4-38 y Tabla 4-5).

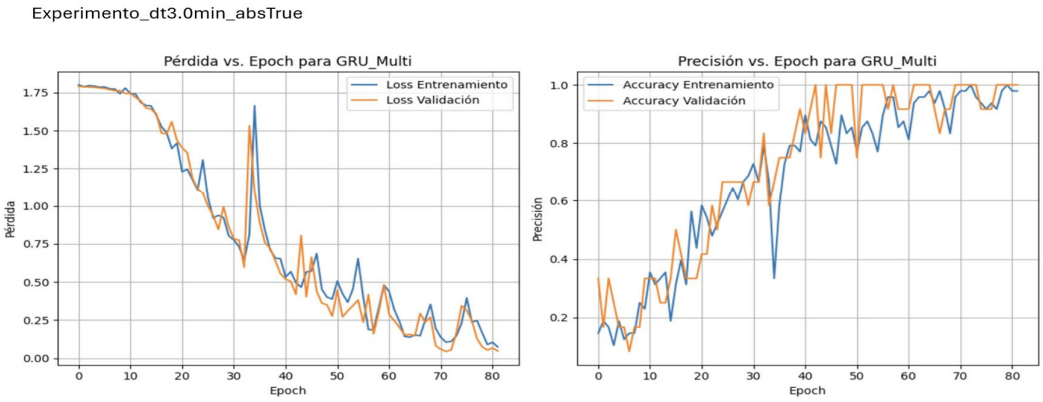


Figura 4-38. Gráficas de validación y entrenamiento en la predicción de Materiales con valores absolutos y dt=3 (Elaboración propia)

Material	P0	P1	P2	P3	P4	P5	Real	Pred
Calentamiento_S4	0.0	0.02	0.0	0.0	0.98	0.0	4	4

Material	P0	P1	P2	P3	P4	P5	Real	Pred
Enfriamiento_A2	0.0	0.98	0.0	0.0	0.01	0.0	1	1
Calentamiento_S5	0.0	0.01	0.0	0.01	0.99	0.0	4	4
Calentamiento_D4	0.0	0.0	0.0	0.99	0.01	0.0	3	3
Enfriamiento_1	1.0	0.0	0.0	0.0	0.0	0.0	0	0
Calentamiento_V3	0.0	0.03	0.0	0.06	0.0	0.91	5	5
Enfriamiento_A5	0.01	0.94	0.0	0.0	0.05	0.0	1	1
Enfriamiento_V3	0.0	0.0	0.0	0.01	0.0	0.99	5	5
Enfriamiento_B2	0.15	0.0	0.82	0.0	0.02	0.0	2	2
Enfriamiento_D1	0.0	0.0	0.0	0.91	0.0	0.09	3	3
Enfriamiento_3	1.0	0.0	0.0	0.0	0.0	0.0	0	0
Calentamiento_B4	0.0	0.0	1.0	0.0	0.0	0.0	2	2

Tabla 4-5. Tabla de validación en la predicción de Materiales con valores absolutos y dt=3 (Elaboración propia)

Tras el análisis de los resultados anteriores, cabe señalar que únicamente se presentan tablas correspondientes al conjunto de validación, no habiéndose considerado un conjunto de test independiente. Esta decisión responde a la limitada disponibilidad de datos, lo que ha llevado a incorporar las muestras inicialmente destinadas al testeo dentro del conjunto de entrenamiento, con el objetivo de maximizar la información disponible y mejorar la capacidad de aprendizaje del modelo.

Este ajuste no afecta a la estructura ni al funcionamiento del modelo, dado que el conjunto de test se emplea habitualmente como una herramienta adicional de comprobación del rendimiento y no interviene directamente en el proceso de entrenamiento. No obstante, dada la escasez de datos, se ha considerado más adecuado priorizar el entrenamiento con el mayor número posible de muestras.

En cualquier caso, tal y como se puede observar en los códigos incluidos en los Anexos IV y V, la funcionalidad correspondiente al conjunto de test se mantiene implementada, lo que permitirá, en futuras ampliaciones con un mayor volumen de datos, realizar una validación completa mediante la separación clásica en conjuntos de entrenamiento, validación y test.

Adicionalmente, se han identificado otras configuraciones con resultados igualmente elevados. En particular, el caso correspondiente a $\Delta t = 5$minutos sin aplicación de valor absoluto presenta un comportamiento robusto, con un alto número de aciertos y una evolución del entrenamiento estable (ver Figura 4-39).

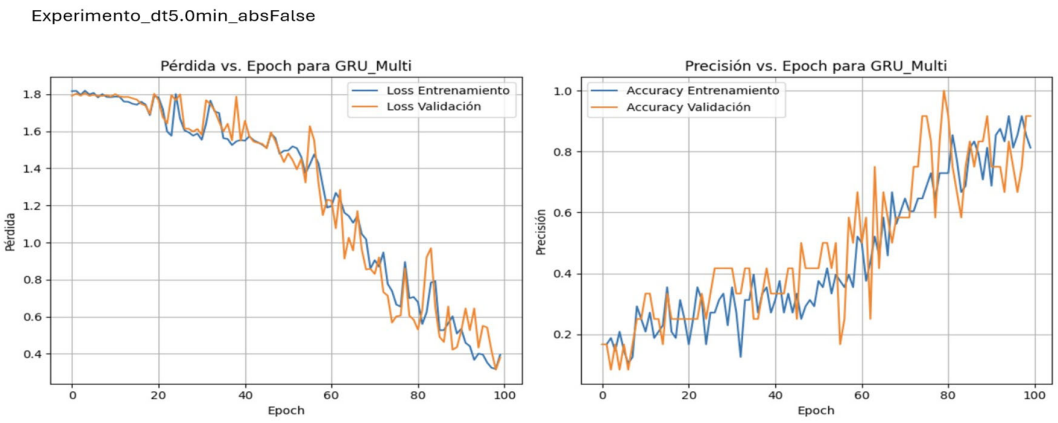


Figura 4-39. Gráficas de validación y entrenamiento en la predicción de Materiales sin valores absolutos y dt=5 (Elaboración propia)

Por otro lado, el caso de $\Delta t = 3$minutos sin valor absoluto, aunque alcanza un número elevado de aciertos, presenta una evolución menos estable del modelo, con mayores oscilaciones en la función de pérdida y en la precisión a lo largo del entrenamiento (ver Figura 4-40).

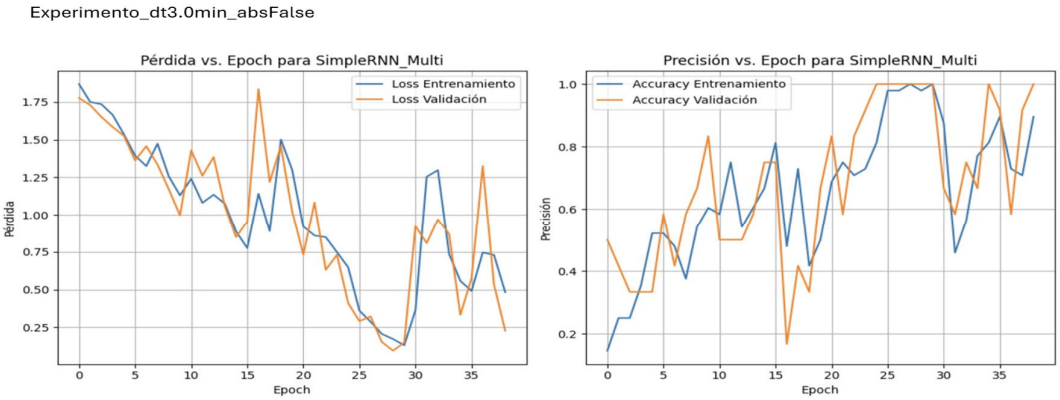


Figura 4-40. Gráficas de validación y entrenamiento en la predicción de Materiales sin valores absolutos y dt=3 (Elaboración propia)

En contraste, configuraciones con intervalos temporales más reducidos, como $\Delta t = 1$minuto, muestran un rendimiento significativamente inferior, evidenciando una menor capacidad de generalización del modelo (ver Figura 4-41).

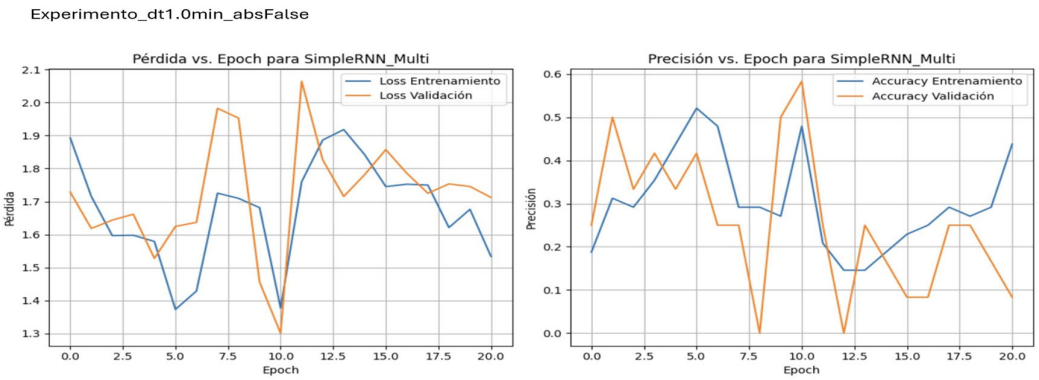


Figura 4-41. Gráficas de validación y entrenamiento en la predicción de Materiales sin valores absolutos y dt=1 (Elaboración propia)

En conjunto, los resultados confirman la elevada fiabilidad del modelo en la clasificación de materiales, así como la influencia del intervalo temporal en el rendimiento obtenido.

4.4.2 Segunda etapa. Detección de sales

Siguiendo el procedimiento de preprocesado y modelización descrito en el apartado 3.5, se ha abordado la clasificación de las sales a partir de las secuencias de imágenes térmicas procesadas mediante distintos modelos de *deep learning*. A diferencia de la predicción de materiales, los resultados obtenidos en este caso presentan una mayor dispersión y un rendimiento global inferior, lo que pone de manifiesto la mayor complejidad inherente a este problema (véase desde la Figura 4-42 hasta la Figura 4-51).

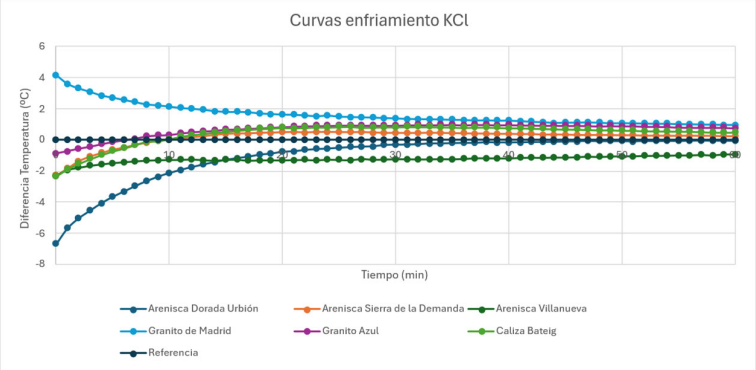


Figura 4-42. Representación gráfica del enfriamiento de KCl en las diferentes muestras (Elaboración propia)

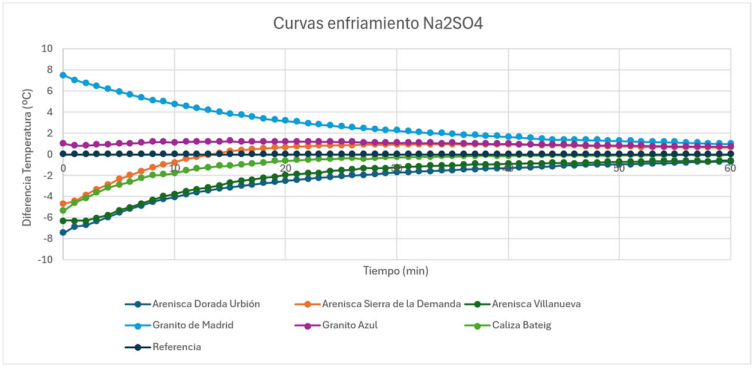


Figura 4-43. Representación gráfica del enfriamiento de Na_2SO_4 en las diferentes muestras (Elaboración propia)

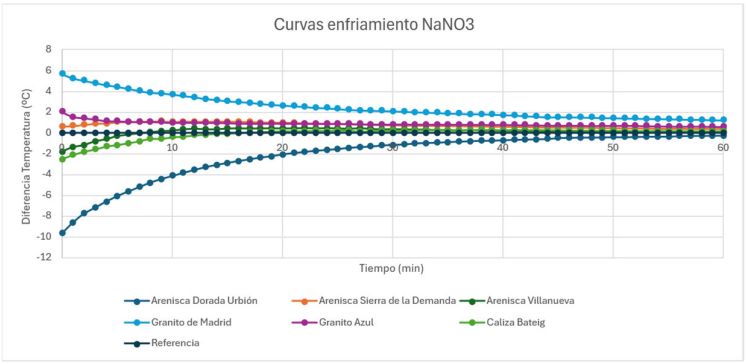


Figura 4-44. Representación gráfica del enfriamiento de $NaNO_3$ en las diferentes muestras (Elaboración propia)

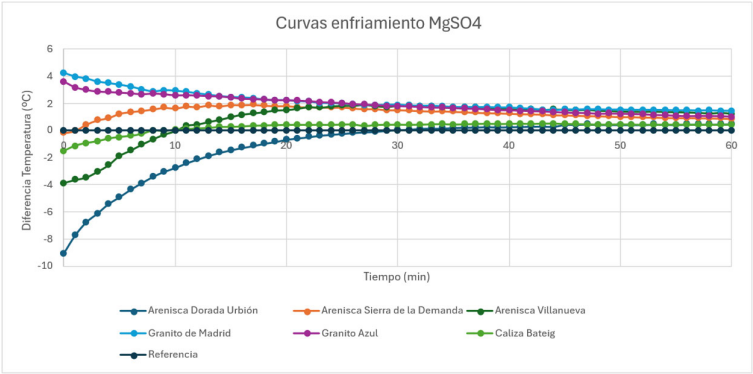


Figura 4-45. Representación gráfica del enfriamiento de $MgSO_4$ en las diferentes muestras (Elaboración propia)

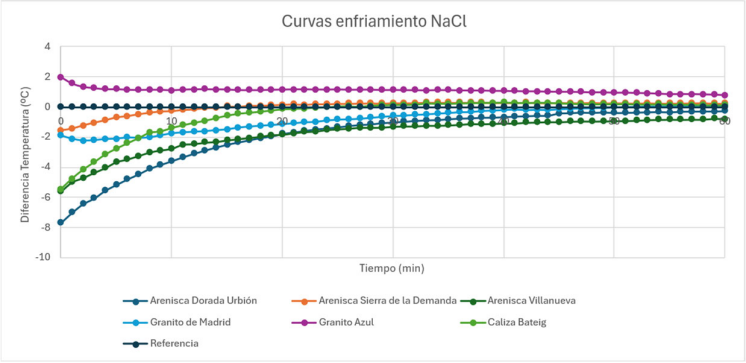


Figura 4-46. Representación gráfica del enfriamiento de NaCl en las diferentes muestras (Elaboración propia)

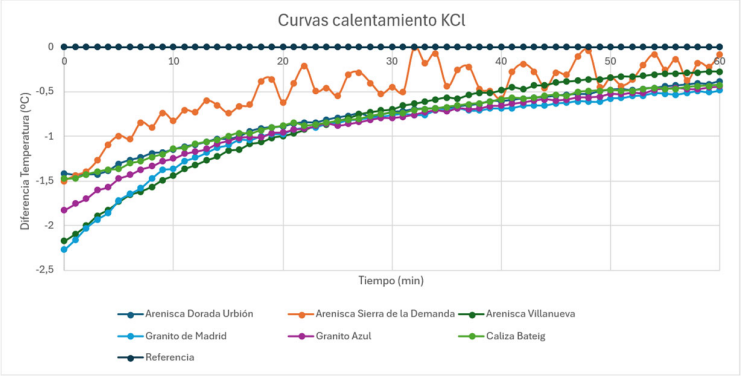


Figura 4-47. Representación gráfica del calentamiento de KCl en las diferentes muestras (Elaboración propia)

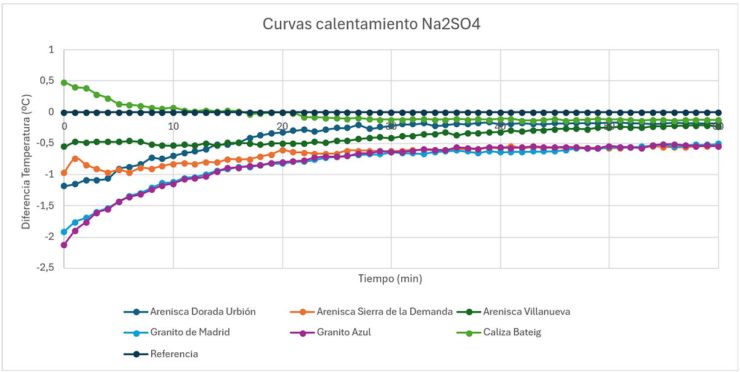


Figura 4-48. Representación gráfica del calentamiento de Na_2SO_4 en las diferentes muestras (Elaboración propia)

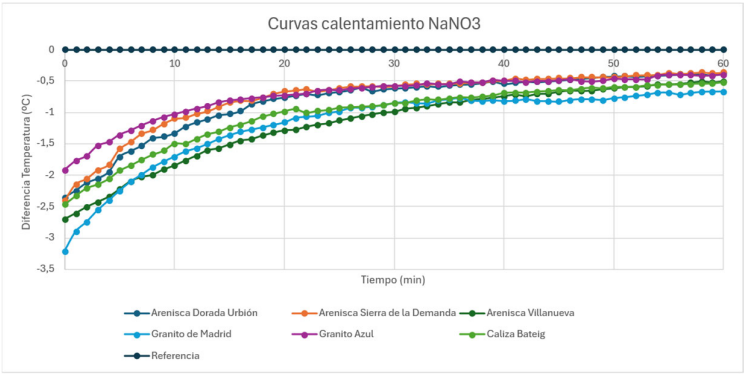


Figura 4-49. Representación gráfica del calentamiento de $NaNO_3$ en las diferentes muestras (Elaboración propia)

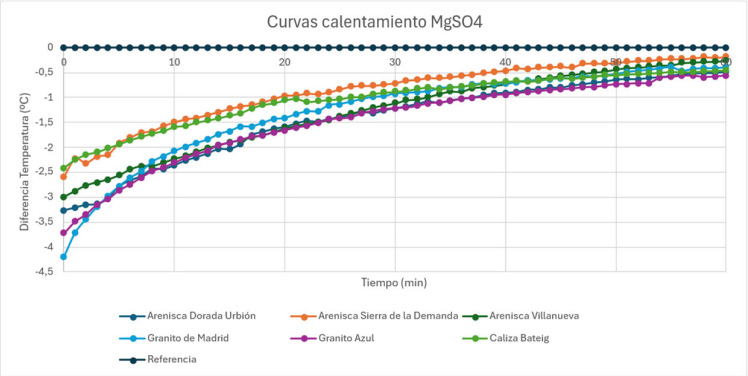


Figura 4-50. Representación gráfica del calentamiento de $MgSO_4$ en las diferentes muestras (Elaboración propia)

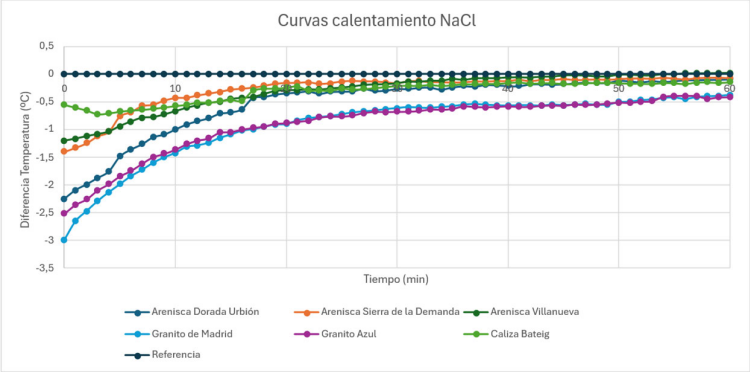


Figura 4-51. Representación gráfica del calentamiento de NaCl en las diferentes muestras (Elaboración propia)

En términos cuantitativos, el mayor número de aciertos se obtiene para la configuración basada en la derivada de la imagen térmica con un intervalo temporal de $\Delta t = 5$minutos, aplicación de valor absoluto y sin transformación logarítmica, alcanzando un total de 8 aciertos (ver Figura 4-52 y Tabla 4-6).

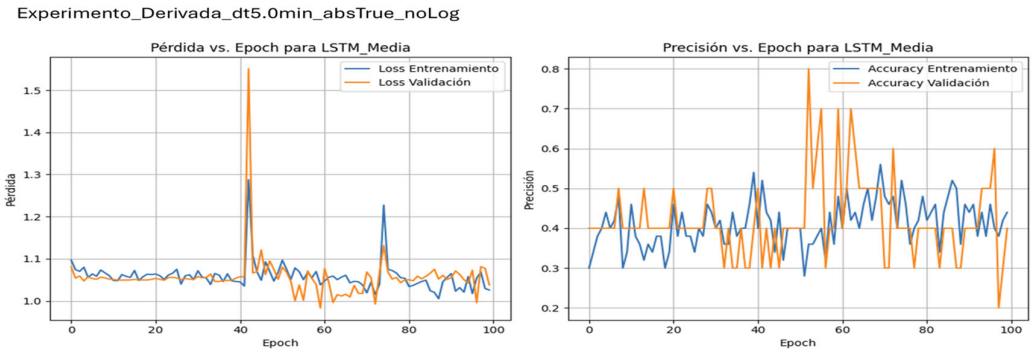


Figura 4-52. Gráficas de validación y entrenamiento en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=5 y sin logaritmo (Elaboración propia)

Material	P0	P1	P2	Real	Pred
Enfriamiento_V4 vs Referencia	0.37	0.46	0.17	1	1
Calentamiento_A3 vs Referencia	0.40	0.40	0.20	2	0

Material	P0	P1	P2	Real	Pred
Enfriamiento_D5 vs Referencia	0.40	0.39	0.21	0	0
Calentamiento_D1 vs Referencia	0.41	0.40	0.19	0	0
Enfriamiento_B3 vs Referencia	0.38	0.45	0.17	2	1
Enfriamiento_V4 Referencia	0.38	0.39	0.23	1	1
Enfriamiento_S4 vs Referencia	0.39	0.43	0.18	1	1
Enfriamiento_A1 vs Referencia	0.40	0.39	0.20	0	0
Enfriamiento_B2 vs Referencia	0.39	0.42	0.19	1	1
Calentamiento_V1 vs Referencia	0.40	0.38	0.22	0	0

Tabla 4-6. Tabla de validación en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=5 y sin logaritmo (Elaboración propia)

No obstante, un análisis más detallado del comportamiento del modelo revela que este resultado debe interpretarse con cautela. En esta configuración, el modelo tiende a asignar probabilidades similares a distintas clases, de modo que, aunque la predicción final coincide en varios casos con la clase correcta, la confianza asociada a dicha predicción es reducida. Este comportamiento sugiere que el modelo no está discriminando de forma clara entre clases, sino que su acierto se aproxima en algunos casos a una decisión con bajo nivel de certeza, lo que limita su fiabilidad desde un punto de vista práctico.

Por este motivo, se considera que la configuración correspondiente a minutos, con aplicación de valor absoluto y transformación logarítmica en la diferencia de temperatura, representa el caso más adecuado desde el punto de vista global, a pesar de presentar un número ligeramente inferior de aciertos en el conjunto de validación (7 sobre 10 aciertos, 70%). En este caso, el modelo muestra una separación más definida entre clases (fallando en la clase de salida 0, aunque acierta en uno de los cuatros casos de esa clase) y una evolución más estable durante el aprendizaje (terminando en torno a 0,6 y 1 en la pérdida de entrenamiento y validación, respectivamente, y cerca de 0,8 y 0,7 en la precisión de entrenamiento y validación, respectivamente), lo que indica una mayor capacidad de generalización (ver Figura 4-53 y Tabla 4-7).

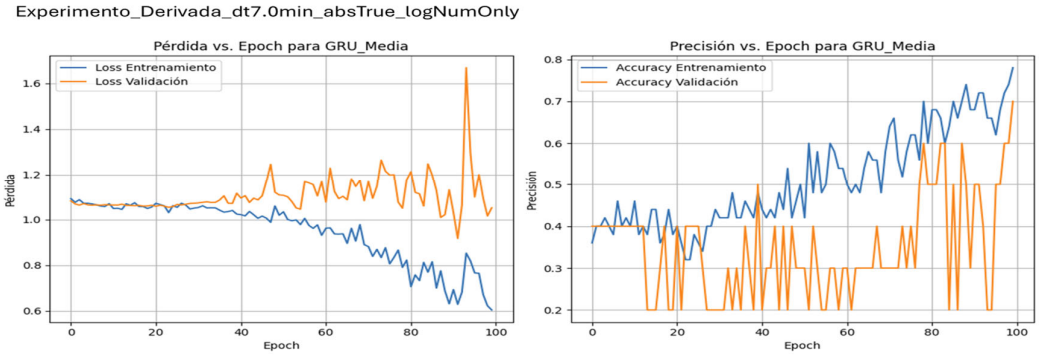


Figura 4-53. Gráficas de validación y entrenamiento en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=7 y con logaritmo (Elaboración propia)

Material	P0	P1	P2	Real	Pred
Enfriamiento_V4 vs Referencia	0.48	0.50	0.02	1	1
Calentamiento_A3 vs Referencia	0.22	0.26	0.52	2	2
Enfriamiento_D5 vs Referencia	0.13	0.56	0.30	0	1
Calentamiento_D1 vs Referencia	0.28	0.68	0.04	0	1
Enfriamiento_B3 vs Referencia	0.33	0.18	0.49	2	2
Calentamiento_V4 vs Referencia	0.12	0.88	0.01	1	1
Enfriamiento_S4 vs Referencia	0.43	0.45	0.11	1	1
Enfriamiento_A1 vs Referencia	0.05	0.93	0.02	0	1
Enfriamiento_B2 vs Referencia	0.28	0.55	0.17	1	1
Calentamiento_V1 vs Referencia	0.53	0.29	0.17	0	0

Tabla 4-7. Tabla de validación en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=7 y con logaritmo (Elaboración propia)

Por otro lado, la configuración correspondiente a un intervalo temporal reducido ($\Delta t = 1$) presenta un rendimiento inferior, con un total de 6 aciertos, lo que evidencia, en términos generales, una menor capacidad de discriminación del modelo en este caso. No obstante, un análisis más detallado de las probabilidades asociadas a las predicciones revela un comportamiento relevante: en aquellos casos en los que el modelo acierta, la diferencia entre la probabilidad asignada a la clase correcta y la del resto de clases es significativamente elevada. Este hecho indica que, a pesar de obtener un menor número total de aciertos, el modelo muestra una mayor confianza en sus predicciones correctas, lo que sugiere una capacidad de discriminación más clara en dichos casos. En consecuencia, esta configuración no puede considerarse deficiente, sino más bien un modelo con un comportamiento relativamente sólido, aunque limitado en términos de generalización global (ver Figura 4-54 y Tabla 4-8).

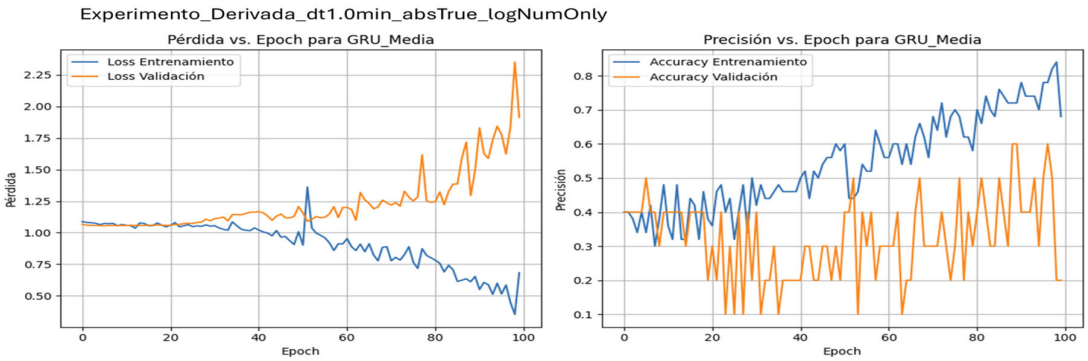


Figura 4-54. Gráficas de validación y entrenamiento en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=1 y con logaritmo (Elaboración propia)

Material	P0	P1	P2	Real	Pred
Enfriamiento_V4 vs Referencia	0.64	0.12	0.23	1	0
Calentamiento_A3 vs Referencia	0.04	0.42	0.54	2	2
Enfriamiento_D5 vs Referencia	0.02	0.67	0.31	0	1
Calentamiento_D1 vs Referencia	0.10	0.87	0.02	0	1
Enfriamiento_B3 vs Referencia	0.18	0.20	0.62	2	2
Calentamiento_V4 vs Referencia	0.30	0.68	0.02	1	1
Enfriamiento_S4 vs Referencia	0.08	0.85	0.08	1	1
Enfriamiento_A1 vs Referencia	0.14	0.48	0.38	0	1
Enfriamiento_B2 vs Referencia	0.24	0.58	0.18	1	1
Calentamiento_V1 vs Referencia	0.61	0.31	0.08	0	0

Tabla 4-8. Tabla de validación en la predicción de Sales aplicando ratios de temperatura, con valores absolutos, dt=1 y con logaritmo (Elaboración propia)

En lo que respecta a las transformaciones basadas en áreas acumuladas, los resultados obtenidos son significativamente inferiores, alcanzando un máximo de 5 aciertos, lo que confirma la menor eficacia de este tipo de representación en el contexto analizado (ver Figura 4-55 y Tabla 4-9).

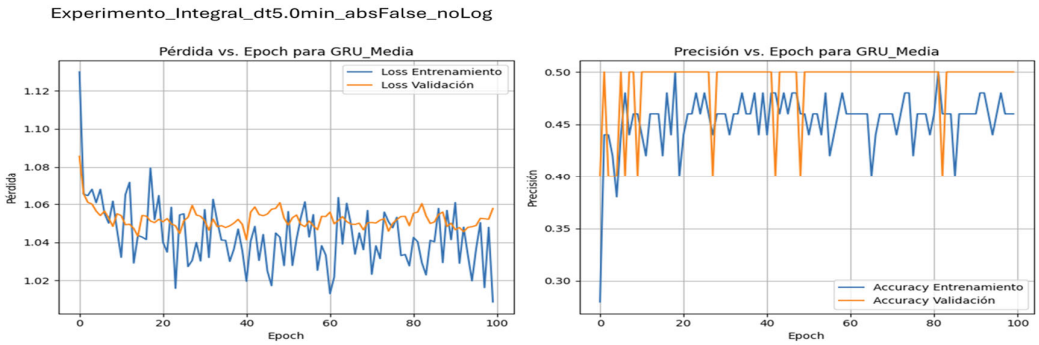


Figura 4-55. Gráficas de validación y entrenamiento en la predicción de Sales aplicando áreas acumuladas, sin valores absolutos, dt=5 y sin logaritmo (Elaboración propia)

Material	P0	P1	P2	Real	Pred
Enfriamiento_V4 vs Referencia	0.38	0.36	0.26	1	0
Calentamiento_A3 vs Referencia	0.38	0.37	0.25	2	0
Enfriamiento_D5 vs Referencia	0.38	0.37	0.25	0	0
Calentamiento_D1 vs Referencia	0.38	0.37	0.26	0	0

Material	P0	P1	P2	Real	Pred
Enfriamiento_B3 vs Referencia	0.38	0.37	0.26	2	0
Calentamiento_V4 vs Referencia	0.38	0.37	0.25	1	0
Enfriamiento_S4 vs Referencia	0.36	0.36	0.28	1	1
Enfriamiento_A1 vs Referencia	0.38	0.36	0.25	0	0
Enfriamiento_B2 vs Referencia	0.38	0.37	0.25	1	0
Calentamiento_V1 vs Referencia	0.38	0.37	0.25	0	0

Tabla 4-9. Tabla de validación en la predicción de Sales aplicando áreas acumuladas, sin valores absolutos, dt=1 y con logaritmo (Elaboración propia)

Un aspecto relevante que se observa en las gráficas asociadas al aprendizaje de los modelos es la presencia de un comportamiento de **sobreajuste (*overfitting*)**, especialmente acusado en comparación con los resultados obtenidos en la predicción de materiales. Este fenómeno se manifiesta en una divergencia entre las curvas de entrenamiento y validación, donde el modelo mejora su rendimiento sobre los datos de entrenamiento mientras empeora o se estabiliza en los datos de validación.

Este comportamiento puede atribuirse principalmente a la **escasez de datos disponibles**, tanto en número de muestras como en diversidad experimental. A diferencia del caso de los materiales, donde las diferencias térmicas son más marcadas, en la clasificación de sales las variaciones entre clases son más sutiles, lo que exige un mayor volumen de datos para que el modelo pueda aprender patrones representativos de forma robusta. La limitada cantidad de datos favorece que el modelo memorice características específicas del conjunto de entrenamiento en lugar de generalizar adecuadamente.

En este sentido, la obtención de resultados más fiables y consistentes requeriría la ampliación del conjunto de datos, incorporando un mayor número de muestras y condiciones experimentales, lo que permitiría mejorar la capacidad de generalización del modelo y reducir los efectos del sobreajuste.

En conjunto, los resultados ponen de manifiesto la mayor complejidad de la predicción de sales frente a la de materiales, así como la fuerte dependencia del rendimiento del modelo respecto al volumen de datos y al preprocesado de la señal. En la Figura 4-56 se recoge el porcentaje máximo de acierto, considerándose más representativo el segundo mejor resultado en la predicción de sales, ya que el primero no presenta una evolución coherente entre entrenamiento y validación. Asimismo, para dt = 7 se observa un buen rendimiento en entrenamiento debido al mayor volumen de datos, aunque acompañado de un claro sobreajuste al no reproducirse en validación; no obstante, las predicciones presentan una distribución de probabilidades más diferenciada entre clases, lo que sugiere que el modelo es capaz de captar cierta estructura en los datos y no se comporta de manera aleatoria.

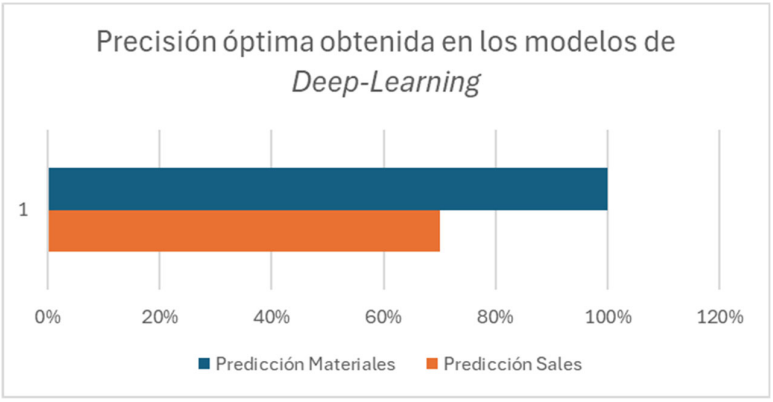


Figura 4-56. Precisión óptima obtenida en los modelos de *Deep-Learning* (Elaboración propia)

5 CONCLUSIONES Y LÍNEAS FUTURAS

5.1 Conclusiones

El presente trabajo ha permitido evaluar la viabilidad del uso combinado de la termografía infrarroja y técnicas de *deep learning* para la identificación de materiales y la detección de compuestos salinos en sistemas constructivos.

En lo que respecta a la clasificación de materiales, los resultados obtenidos han sido altamente satisfactorios, alcanzándose niveles de acierto elevados y consistentes. Esto confirma que la respuesta térmica de los materiales contiene información suficiente para su discriminación, especialmente cuando se aplican transformaciones adecuadas de la señal. En este sentido, los intervalos temporales intermedios ($\Delta t = 3$ y $\Delta t = 5$) han demostrado ser los más adecuados para capturar la dinámica térmica, destacando particularmente el caso de $\Delta t = 3$ con valor absoluto por su robustez.

Por otro lado, la predicción de sales presenta una mayor complejidad, tal y como cabía esperar. No obstante, se ha logrado un avance significativo respecto a trabajos previos, pasando de un 41 % de acierto a aproximadamente un 70 % en la configuración más adecuada ($\Delta t = 7$). Este resultado no solo implica una mejora cuantitativa, sino también cualitativa, ya que el modelo muestra una mayor capacidad de discriminación real entre clases, evitando predicciones ambiguas. Aunque otras configuraciones presentan un mayor número de aciertos, estos no reflejan una separación efectiva entre clases, por lo que $\Delta t = 7$ se considera la opción más fiable y representativa del comportamiento del modelo. Asimismo, este trabajo incorpora un valor añadido fundamental al permitir la identificación previa del material, lo que facilita y mejora notablemente el proceso posterior de análisis.

Se ha identificado además la presencia de sobreajuste (*overfitting*) en los modelos de predicción de sales, atribuible principalmente a la limitada cantidad de datos disponibles, lo que pone de manifiesto la necesidad de ampliar el conjunto de datos en futuras investigaciones para mejorar la capacidad de generalización.

De forma complementaria, los resultados obtenidos en la caracterización mecánica mediante ensayos de ultrasonidos y dureza superficial presentan una clara coherencia con las observaciones visuales y con el comportamiento esperado de cada material frente a la acción de las sales. Los materiales más porosos muestran una mayor degradación, reflejada tanto en la reducción de sus propiedades mecánicas como en su estado superficial, mientras que los materiales más compactos mantienen una mayor integridad. Esta concordancia refuerza la validez global del estudio y la fiabilidad del procedimiento experimental desarrollado.

Desde un punto de vista aplicado, este trabajo presenta un notable potencial en el ámbito del diagnóstico no destructivo en edificación. En un escenario real, sería posible analizar un muro mediante termografía infrarroja, obteniendo su respuesta térmica sin necesidad de intervención directa sobre la estructura. A partir de estos datos, el modelo permitiría identificar el tipo de material constitutivo. Posteriormente, mediante el uso de una muestra de referencia del mismo material analizada en condiciones controladas, se podría obtener su comportamiento térmico en ausencia de sales. La comparación entre ambas respuestas, junto con el procedimiento de sustracción de señales desarrollado, permitiría aislar la contribución térmica asociada a la presencia de compuestos salinos. En este sentido, cabe destacar que el modelo no identifica directamente la sal completa, sino el anión asociado, que es el principal responsable de los procesos de degradación observados, ya que determina los mecanismos de cristalización, expansión y daño en el material. Este enfoque permitiría, por tanto, no solo detectar la presencia de contaminación, sino también caracterizar el tipo de agente degradante y anticipar posibles patologías estructurales (ver Figura 5-1).

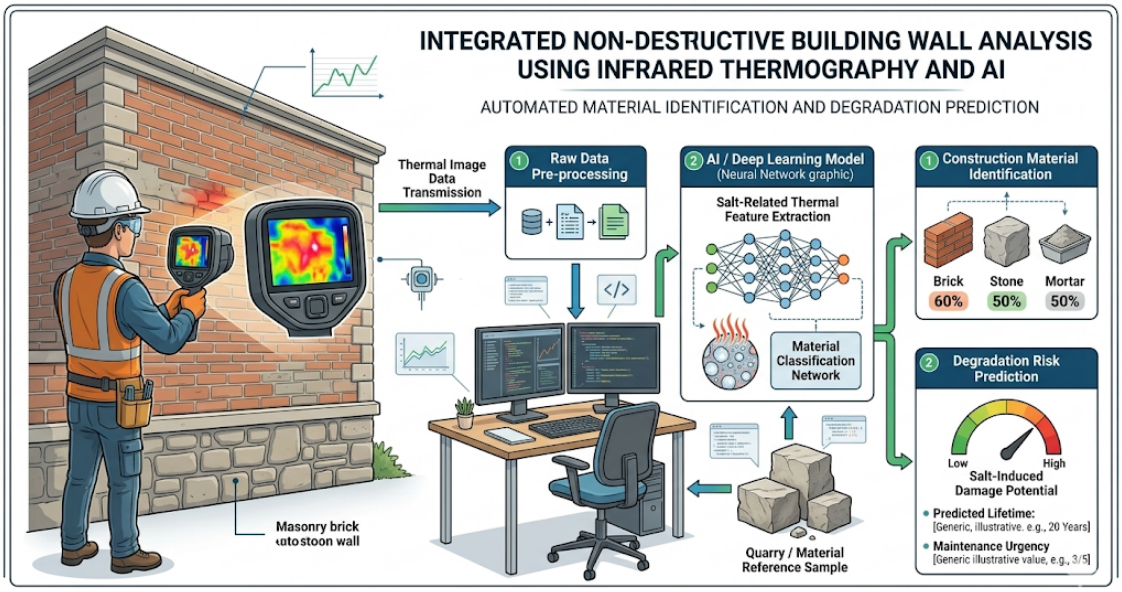


Figura 5-1. Caso práctico para la predicción de patología en una estructura

(Elaborado con inteligencia artificial Gemini)

En conjunto, el trabajo demuestra de forma clara el potencial de la integración entre técnicas de inspección no destructiva y *deep learning*, así como el avance significativo alcanzado en la capacidad de análisis y predicción, sentando una base sólida para futuras aplicaciones en el ámbito de la ingeniería y la conservación de edificaciones.

5.2 Líneas futuras

A partir de los resultados obtenidos, se identifican diversas líneas de trabajo que permitirían mejorar y ampliar el alcance del estudio.

En primer lugar, resulta fundamental la ampliación del conjunto de datos, tanto en número de muestras como en variedad de condiciones experimentales. La incorporación de un mayor volumen de datos permitiría mejorar la capacidad de generalización de los modelos y reducir los efectos de sobreajuste observados en la predicción de sales.

Asimismo, sería de interés explorar nuevas arquitecturas de *deep learning*, incluyendo modelos híbridos o el uso de redes convolucionales aplicadas directamente sobre secuencias termográficas, lo que podría permitir una mejor extracción de características espaciales y temporales.

Otra línea de mejora consiste en el desarrollo de técnicas de preprocesado más avanzadas, orientadas a reducir el ruido de la señal y a resaltar las diferencias entre clases, como el uso de filtros adaptativos o técnicas de análisis en el dominio frecuencial.

Del mismo modo, podría profundizarse en el estudio de la influencia de las condiciones ambientales, como la temperatura o la humedad relativa, en la respuesta térmica de las muestras, con el fin de mejorar la robustez de los modelos en escenarios reales.

Finalmente, como aplicación práctica, sería especialmente relevante la integración de estos modelos en sistemas de inspección automatizados, que permitan la evaluación en tiempo real del estado de estructuras, facilitando la toma de decisiones en tareas de mantenimiento y conservación.

6 BIBLIOGRAFÍA

[1] Structural-Safety, «Elliot Lake Mall Inquiry». Accedido: 27 de marzo de 2026. [En línea]. Disponible en: https://www.designingbuildings.co.uk/wiki/Elliot_Lake_Mall_Inquiry
[2] «Blatchford: At Elliot Lake's Algo Centre Mall, the wisdom of Benjamin Franklin went wanting | Ottawa Citizen». Accedido: 27 de marzo de 2026. [En línea]. Disponible en: https://ottawacitizen.com/news/national/blatchford-at-elliot-lakes-algo-centre-mall-the-wisdom-of-benjamin-franklin-went-wanting
[3] J. M. Cordero de la Puente, J. (advisor) Feijoo Conde, y I. (advisor) Garrido González, «Influencia de la presencia de sales en la respuesta técnica de materiales de construcción», abr. 2024, Accedido: 27 de marzo de 2026. [En línea]. Disponible en: https://calderon.cud.uvigo.es/handle/123456789/779
[4] I. de Rosario Amado y J. F. Conde, *Alteración de las rocas y otros materiales de construcción en los monumentos: técnicas de conservación*. Servizo de Publicacións da Universidade de Vigo, 2016.
[5] F. J. Alonso y R. M. Esbert, «ANÁLISIS DEL DETERIORO EN LOS MATERIALES PÉTREOS DE EDIFICACIÓN», n.º 3, 2006.
[6] F. L. Rodríguez, V. R. Rodriguez, J. S. C. Astorqui, I. T. Gomez, y P. U. D. Mingo, «MANUAL DE PATOLOGÍA DE LA EDIFICACIÓN».
[8] M.-A. Bello-López, «Los procesos químicos como factores de alteración de materiales petreos empleados en construcción», *RE*, vol. 11, pp. 39-44, ene. 1970, doi: 10.15581/020.11.35023.
[9] M. Kämmlein y H. Stollhofen, «Pore-fluid-dependent controls of matrix and bulk thermal conductivity of mineralogically heterogeneous sandstones», *Geothermal Energy*, vol. 7:13, p. 2019, may 2019, doi: 10.1186/s40517-019-0129-4.
[10] J. L. Prada y A. Valenciano, «Procesos de alteración de materiales pétreos en edificios de interés histórico».
[11] «Metodología aplicable al estudio de la alteración de rocas graníticas usadas en la construcción».
[12] R. Flatt *et al.*, «Predicting salt damage in practice: A theoretical insight into laboratory tests.», *RILEM Technical Letters*, vol. 2, pp. 108-118, dic. 2017, doi: 10.21809/rilemtechlett.2017.41.
[13] Geocaching, «Geocaching - The Official Global GPS Cache Hunt Site». Accedido: 18 de febrero de 2026. [En línea]. Disponible en: http://www.geocaching.com/
[14] H. Cañola, F. Granda-Ramírez, y J. Arroyave-Rojas, «Emulsión asfáltica como alternativa de reducción de la presencia de eflorescencias en morteros de pega», *Revista UIS Ingenierías*, vol. 20, n.º 1, pp. 103-114, 2021, doi: 10.18273/revuin.v20n1-2021009.
[15] C. Groot, «Repair mortars for historic masonry; Effects of the binder choice on durability».

[16] R. Fort, J. Feijoo, M. J. Varas–Muriel, M. A. Navacerrada, M. M. Barbero-Barrera, y D. De La Prida, «Appraisal of non-destructive in situ techniques to determine moisture- and salt crystallization-induced damage in dolostones», *Journal of Building Engineering*, vol. 53, p. 104525, ago. 2022, doi: 10.1016/j.jobe.2022.104525.
[17] J. Gosalbez *et al.*, *Application of non-destructive evaluation*. [En línea]. Disponible en: https://www.iteam.upv.es/wp-content/uploads/pdf_articles/51.pdf
[18] *Diagnosis of Heritage Buildings by Non-Destructive Techniques*. Elsevier, 2024. doi: 10.1016/C2022-0-02361-8.
[19] B. Zapico-Blanco, J. D. Rodríguez-Mariscal, y M. Solis Muñiz, «Ultrasonic Evaluation of Mechanical Properties in Heritage Buildings Constructed with Córdoba Freestone», *Heritage*, vol. 8, n.º 11, p. 462, nov. 2025, doi: 10.3390/heritage8110462.
[20] R. Usamentiaga, P. Venegas, J. Guerediaga, L. Vega, J. Molleda, y F. Bulnes, «Infrared Thermography for Temperature Measurement and Non-Destructive Testing», *Sensors*, vol. 14, n.º 7, pp. 12305-12348, jul. 2014, doi: 10.3390/s140712305.
[21] T. Astarita, G. Cardone, G. M. Carlomagno, y C. Meola, «A survey on infrared thermography for convective heat transfer measurements», *Optics & Laser Technology*, vol. 32, n.º 7-8, pp. 593-610, oct. 2000, doi: 10.1016/S0030-3992(00)00086-4.
[22] G. M. Carlomagno y G. Cardone, «Infrared thermography for convective heat transfer measurements», *Exp Fluids*, vol. 49, n.º 6, pp. 1187-1218, dic. 2010, doi: 10.1007/s00348-010-0912-2.
[23] L. Muzika, J. Tesař, M. Švantner, J. Skála, y P. Honnerová, «Comparison of Methods for Emissivity Influence Suppression on Thermographic Data», *Buildings*, vol. 13, n.º 1, p. 69, dic. 2022, doi: 10.3390/buildings13010069.
[24] T. L. Bergman, *Fundamentals of Heat and Mass Transfer*. John Wiley & Sons, 2011.
[25] C. Hall y W. D. Hoff, *Water transport in brick, stone and concrete*, Third edition. Boca Raton London New York: CRC Press, 2022.
[26] P.-M. Robitaille y S. J. Crothers, «"The Theory of Heat Radiation" Revisited: A Commentary on the Validity of Kirchhoff's Law of Thermal Emission and Max Planck's Claim of Universality», vol. 11, n.º 2, 2015.
[27] C. A. Balaras y A. A. Argiriou, «Infrared thermography for building diagnostics», *Energy and Buildings*, vol. 34, n.º 2, pp. 171-183, feb. 2002, doi: 10.1016/S0378-7788(01)00105-0.
[28] D. L. Balageas, «Termografía Infrarroja : una técnica multifacética para la Evaluación No Destructiva (END)».
[29] A. Rogalski, «Infrared detectors: an overview».
[30] N. Rajic, «Principal Component Thermography».
[31] F. J. Madruga, C. Ibarra-Castanedo, O. M. Conde, J. M. López-Higuera, y X. Maldague, «Infrared thermography processing based on higher-order statistics», *NDT & E International*, vol. 43, n.º 8, pp. 661-666, nov. 2010, doi: 10.1016/j.ndteint.2010.07.002.
[32] S. J. Russell y P. Norvig, *Artificial intelligence: a modern approach*, 13. [Dr.]. en Prentice Hall series in artificial intelligence. Upper Saddle River, NJ: Prentice Hall, 1995.
[33] T. Mitchell, «Machine - Learning - Tom Mitchell», mar. 2026, Accedido: 17 de marzo de 2026. [En línea]. Disponible en: http://localhost:8080/xmlui/handle/1/6078
[34] Ian Goodfellow, Yoshua Bengio, y Aaron Courville, *Deep Learning*.
[35] «CS 230 - Recurrent Neural Networks Cheatsheet». Accedido: 17 de marzo de 2026. [En línea]. Disponible en: https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks/
[36] Z. C. Lipton, J. Berkowitz, y C. Elkan, «A Critical Review of Recurrent Neural Networks for Sequence Learning», 17 de octubre de 2015, *arXiv*: arXiv:1506.00019. doi: 10.48550/arXiv.1506.00019.
[37] W. Feng, N. Guan, Y. Li, X. Zhang, y Z. Luo, *Audio visual speech recognition with multimodal recurrent neural networks*. 2017, p. 688. doi: 10.1109/IJCNN.2017.7965918.
[38] Jürgen Schmidhuber, *Long Short-Term Memory*.

[39] S. Mohsen, A. Elkaseer, y S. Scholz, «Industry 4.0-Oriented Deep Learning Models for Human Activity Recognition», *IEEE Access*, vol. PP, pp. 1-1, nov. 2021, doi: 10.1109/ACCESS.2021.3125733.
[40] J. Chung, C. Gulcehre, K. Cho, y Y. Bengio, «Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling», 11 de diciembre de 2014, *arXiv*: arXiv:1412.3555. doi: 10.48550/arXiv.1412.3555.
[41] K. Abdelli, H. Griesser, y S. Pachnicke, *A Machine Learning-based Framework for Predictive Maintenance of Semiconductor Laser for Optical Communication*. 2022. doi: 10.48550/arXiv.2211.02842.
[42] M. A. García-del-Cura, C. P. Soba, D. Benavente, y A. Bernabéu, «Dioritas y rocas graníticas del Sistema Central usadas en el adoquinado histórico de la ciudad de Madrid. Valoración según la normativa europea de pavimentos», 2008.
[43] M. García Rodríguez, M. Gómez Heras, M. Alvarez De Buergo, R. Fort González, y J. D. Centeno Carrillo, «Caracterización de agrietamientos poligonales sobre granito en la Pedriza de Manzanares y en Cenicientos, Madrid (Sistema Central)», *M+A. Revista Electrónica de Medioambiente*, vol. 15, n.º 1, pp. 22-36, jun. 2014, doi: 10.5209/rev_MARE.2014.v15.n1.45567.
[44] L. Sousa, P. Pita, J. Carvalho, y J. Lourenço, «Granitos Ornamentais do Norte de Portugal - Caraterísticas, Potencialidades e Constrangimentos».
[45] M. A. Villasol, «Conservación de la Piedra Bateig: análisis de la eficacia de consolidantes e hidrofugantes basados en nanopartículas».
[46] A. Gómez-Marfil y M. J. Varas-Muriel, «Demanda Gold sandstone (Burgos): effect of consolidation and ageing tests on its petrographic and petrophysical properties», *Materiales de Construcción*, vol. 73, n.º 349, pp. e309-e309, mar. 2023, doi: 10.3989/mc.2023.296622.
[47] C. Vielba Cuerpo, «La arenisca de Villamayor en revestimientos de fachada», PhD Thesis, Universidad Politécnica de Madrid, 2001. doi: 10.20868/UPM.thesis.143.
[48] J. Ordaz, «Características físicas y alterabilidad de la piedra de Villamayor (Salamanca)», *Mater. constr.*, vol. 33, n.º 190-191, pp. 85-95, sep. 1983, doi: 10.3989/mc.1983.v33.i190-191.973.
[49] E. Doehne, «Salt weathering: a selective review», *SP*, vol. 205, n.º 1, pp. 51-64, ene. 2002, doi: 10.1144/GSL.SP.2002.205.01.05.
[50] C. T. Oguchi y S. Yu, «A review of theoretical salt weathering studies for stone heritage», *Prog Earth Planet Sci*, vol. 8, n.º 1, p. 32, may 2021, doi: 10.1186/s40645-021-00414-x.
[51] N. Tsui, R. J. Flatt, y G. W. Scherer, «Crystallization damage by sodium sulfate», *Journal of Cultural Heritage*, vol. 4, n.º 2, pp. 109-115, abr. 2003, doi: 10.1016/S1296-2074(03)00022-0.
[52] E. Ruiz-Agudo, F. Mees, P. Jacobs, y C. Rodriguez-Navarro, «The role of saline solution properties on porous limestone salt weathering by magnesium and sodium sulfates», *Environ Geol*, vol. 52, n.º 2, pp. 269-281, mar. 2007, doi: 10.1007/s00254-006-0476-x.
[53] M. P. Bracciale, S. Sammut, J. Cassar, M. L. Santarelli, y A. Marrocchi, «Molecular Crystallization Inhibitors for Salt Damage Control in Porous Materials: An Overview», *Molecules*, vol. 25, n.º 8, p. 1873, abr. 2020, doi: 10.3390/molecules25081873.
[54] M. A. Khodabandeh y N. Rozgonyi-Boissinot, «The Effect of Salt Weathering and Water Absorption on the Ultrasonic Pulse Velocities of Highly Porous Limestone», *Period. Polytech. Civil Eng.*, mar. 2022, doi: 10.3311/PPci.18647.
[55] M. Montoto San Miguel, «Técnicas no destructivas aplicadas a la conservación del patrimonio histórico».
[56] R. Huete-Fuertes, P. Rubio-de-Hita, y C. Rodríguez-Liñán, «Aplicación de los métodos de ultrasonidos a la verificación de materiales de construcción», *RE*, vol. 13, pp. 9-22, ene. 1970, doi: 10.15581/020.13.34995.
[57] F. J. A. Murillo y J. G. Aguilar, «Caracterización petrofísica del material pétreo: Ventajas e inconvenientes de la emisión sonora frente a otros métodos.».
[58] A. Basu y A. Aydin, «A method for normalization of Schmidt hammer rebound values», *International Journal of Rock Mechanics and Mining Sciences*, vol. 41, n.º 7, pp. 1211-1214, oct. 2004, doi: 10.1016/j.ijrmms.2004.05.001.

[59] B. Li *et al.*, «Salt migration and capillary absorption characteristics of cement mortar partially immersed in NaCl solution», *Journal of Building Engineering*, vol. 64, p. 105605, abr. 2023, doi: 10.1016/j.jobe.2022.105605.
[60] M. Y. Çelik y M. Sert, «An assessment of capillary water absorption changes related to the different salt solutions and their concentrations ratios in the Döğer tuff (Afyonkarahisar-Turkey) used as building stone of cultural heritages», *Journal of Building Engineering*, vol. 35, p. 102102, mar. 2021, doi: 10.1016/j.jobe.2020.102102.
[61] «Building Applications». Accedido: 27 de marzo de 2026. [En línea]. Disponible en: https://www.flir.com/en-us/home/
[62] Y.-C. Yeh, W.-J. Wang, y C. W. Chiou, «Cardiac arrhythmia diagnosis method using linear discriminant analysis on ECG signals», *Measurement*, vol. 42, n.º 5, pp. 778-789, jun. 2009, doi: 10.1016/j.measurement.2009.01.004.
[63] F. Hou, Y. Zhang, Y. Zhou, M. Zhang, B. Lv, y J. Wu, «Review on Infrared Imaging Technology», *Sustainability*, vol. 14, n.º 18, p. 11161, sep. 2022, doi: 10.3390/su141811161.
[64] «Técnicas no destructivas aplicadas a la conservación del patrimonio histórico».
[65] «User's manual FLIR Exx series».
[66] A. A. Yagüe, *Los Materiales graníticos hercínicos del sistema central español*. IGME, 1975.
[67] C. de Córdoba, «Actualidad profesional energía & minas».
[68] J. Carvalho *et al.*, «A Pedra Natural Portuguesa – Proposta de harmonização de Designações Comerciais».

ANEXO I: IMPLICACIONES SOCIALES, Y/O ECONÓMICAS, Y/O AMBIENTALES

El presente Trabajo Fin de Grado, centrado en la identificación de patologías en materiales de construcción mediante técnicas de *deep learning* aplicadas a datos termográficos, presenta diversas implicaciones de carácter social, económico y ambiental, que se exponen a continuación.

- **Implicaciones sociales:** desde el punto de vista social, este trabajo contribuye a la mejora de la seguridad en las edificaciones mediante el desarrollo de herramientas capaces de detectar de forma temprana patologías estructurales. La identificación precoz de procesos de degradación asociados a la presencia de sales solubles permite anticipar fallos estructurales potencialmente graves, reduciendo riesgos para los usuarios y facilitando la toma de decisiones en materia de mantenimiento.

  Asimismo, la aplicación de técnicas no destructivas como la termografía infrarroja permite realizar inspecciones sin alterar el material, lo que resulta especialmente relevante en el ámbito de la conservación del patrimonio construido. En este sentido, el trabajo favorece el desarrollo de metodologías más seguras, eficientes y respetuosas con las estructuras analizadas.

- **Implicaciones económicas:** en el ámbito económico, el desarrollo de modelos basados en *deep learning* orientados al diagnóstico de patologías permite optimizar los procesos de inspección y mantenimiento de estructuras. La detección temprana de daños reduce la necesidad de intervenciones correctivas de gran envergadura, que suelen implicar elevados costes.

  Además, la automatización del análisis de datos termográficos contribuye a disminuir los tiempos de evaluación y los recursos necesarios, mejorando la eficiencia de los procesos de diagnóstico. Este enfoque favorece la implementación de estrategias de mantenimiento predictivo, lo que permite una gestión más eficiente del ciclo de vida de las infraestructuras.

- **Implicaciones ambientales:** desde el punto de vista ambiental, el uso de técnicas de inspección no destructivas evita la extracción de muestras y la alteración de los materiales, reduciendo el impacto asociado a métodos tradicionales. Esto contribuye a minimizar la generación de residuos y a preservar la integridad de las estructuras analizadas.

  Asimismo, la mejora en la detección de procesos de degradación permite prolongar la vida útil de los materiales de construcción, reduciendo la necesidad de sustitución y, por tanto, el consumo de recursos naturales. En consecuencia, este trabajo se alinea con principios de sostenibilidad y conservación del patrimonio, promoviendo una gestión más responsable de los materiales y de las infraestructuras.

ANEXO II: REFLEXIONES ÉTICAS Y SOCIALES

El desarrollo del presente Trabajo Fin de Grado, basado en la aplicación de técnicas de inteligencia artificial al diagnóstico de patologías estructurales, implica una serie de consideraciones éticas y sociales que deben ser tenidas en cuenta.

En primer lugar, es fundamental señalar que los modelos de *deep learning* desarrollados en este trabajo constituyen herramientas de apoyo a la toma de decisiones, y no sistemas sustitutivos del criterio técnico. La interpretación de los resultados obtenidos debe realizarse siempre bajo la supervisión de un profesional cualificado, especialmente en un ámbito como el de la seguridad estructural, donde las decisiones pueden tener consecuencias relevantes.

Por otro lado, la fiabilidad de los modelos depende directamente de la calidad y cantidad de los datos utilizados en su entrenamiento. Tal y como se ha observado en el desarrollo del trabajo, la disponibilidad limitada de datos condiciona la capacidad de generalización de los modelos, lo que puede dar lugar a incertidumbre en las predicciones. Desde una perspectiva ética, resulta imprescindible reconocer estas limitaciones y evitar una sobreestimación de la capacidad del modelo.

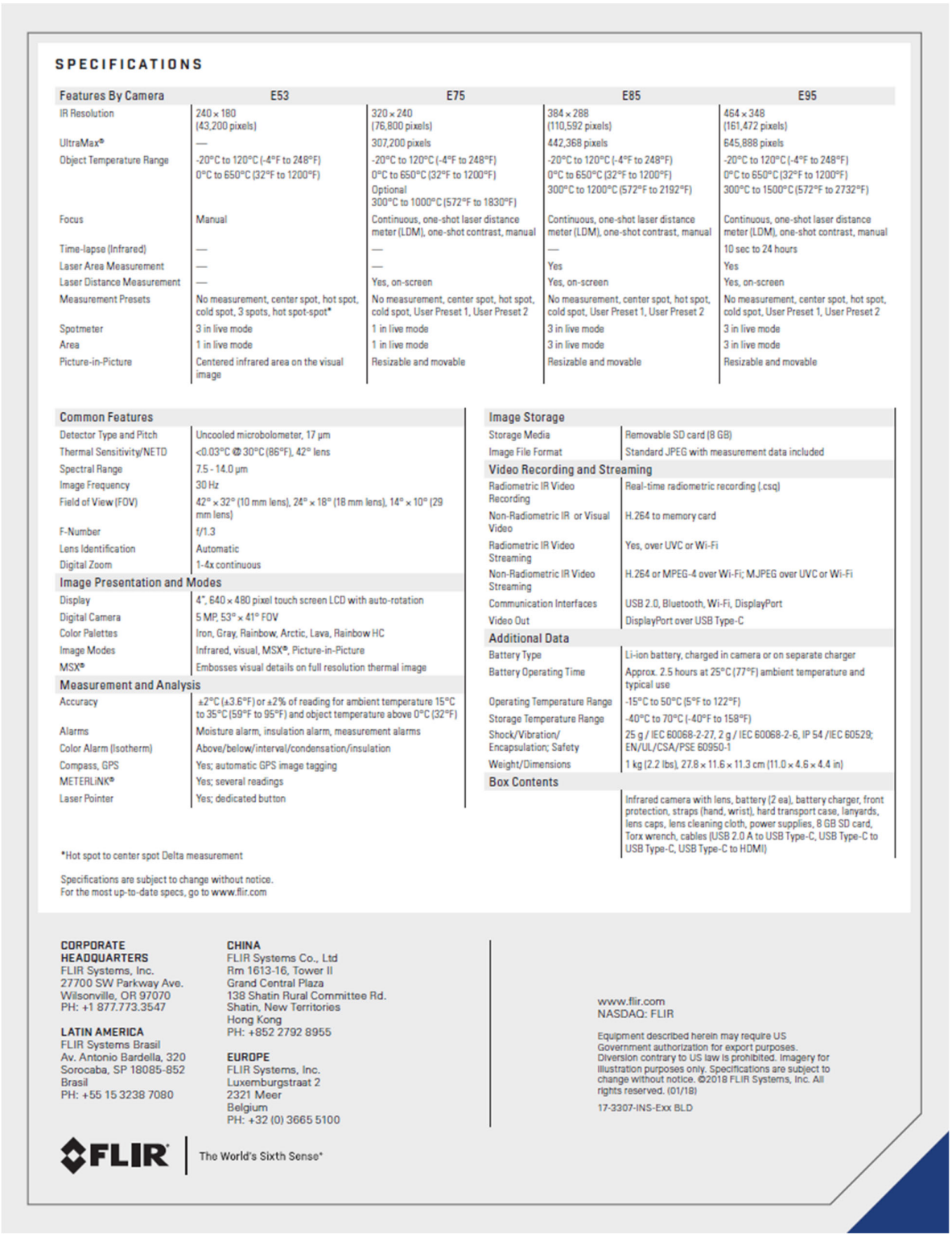
Asimismo, debe considerarse la responsabilidad asociada al uso de estas herramientas. En aplicaciones reales, la decisión final recae siempre en el técnico responsable, siendo el modelo una herramienta complementaria que facilita el análisis, pero no lo sustituye.

Desde el punto de vista social, la incorporación de estas tecnologías permite mejorar la eficiencia y precisión en los procesos de inspección, contribuyendo a una mayor seguridad en las infraestructuras. No obstante, también implica la necesidad de una adecuada formación por parte de los profesionales, que deben comprender tanto el funcionamiento como las limitaciones de los sistemas basados en inteligencia artificial.

Finalmente, el trabajo se ha desarrollado siguiendo criterios de transparencia y reproducibilidad, documentando los procedimientos y modelos empleados, lo que permite su revisión y validación por parte de terceros.

En conjunto, el uso de inteligencia artificial en este ámbito debe abordarse desde una perspectiva responsable, garantizando siempre la supervisión humana, la transparencia en los resultados y una adecuada interpretación de las predicciones obtenidas.

ANEXO III: ESPECIFICACIONES DE LA CÁMARA FLIR



SPECIFICATIONS

Features By Camera	E53	E75	E85	E95
IR Resolution	240 × 180 (43,200 pixels)	320 × 240 (76,800 pixels)	384 × 288 (110,592 pixels)	464 × 348 (161,472 pixels)
UltraMax®	—	307,200 pixels	442,368 pixels	645,888 pixels
Object Temperature Range	-20°C to 120°C (-4°F to 248°F) 0°C to 650°C (32°F to 1200°F)	-20°C to 120°C (-4°F to 248°F) 0°C to 650°C (32°F to 1200°F) Optional 300°C to 1000°C (572°F to 1830°F)	-20°C to 120°C (-4°F to 248°F) 0°C to 650°C (32°F to 1200°F) 300°C to 1200°C (572°F to 2192°F)	-20°C to 120°C (-4°F to 248°F) 0°C to 650°C (32°F to 1200°F) 300°C to 1500°C (572°F to 2732°F)
Focus	Manual	Continuous, one-shot laser distance meter (LDM), one-shot contrast, manual	Continuous, one-shot laser distance meter (LDM), one-shot contrast, manual	Continuous, one-shot laser distance meter (LDM), one-shot contrast, manual
Time-lapse (Infrared)	—	—	—	10 sec to 24 hours
Laser Area Measurement	—	—	Yes	Yes
Laser Distance Measurement	—	Yes, on-screen	Yes, on-screen	Yes, on-screen
Measurement Presets	No measurement, center spot, hot spot, cold spot, 3 spots, hot spot-spot*	No measurement, center spot, hot spot, cold spot, User Preset 1, User Preset 2	No measurement, center spot, hot spot, cold spot, User Preset 1, User Preset 2	No measurement, center spot, hot spot, cold spot, User Preset 1, User Preset 2
Spotmeter	3 in live mode	1 in live mode	3 in live mode	3 in live mode
Area	1 in live mode	1 in live mode	3 in live mode	3 in live mode
Picture-in-Picture	Centered infrared area on the visual image	Resizable and movable	Resizable and movable	Resizable and movable

Common Features	
Detector Type and Pitch	Uncooled microbolometer, 17 µm
Thermal Sensitivity/NETD	<0.03°C @ 30°C (86°F), 42° lens
Spectral Range	7.5 - 14.0 µm
Image Frequency	30 Hz
Field of View (FOV)	42° × 32° (10 mm lens), 24° × 18° (18 mm lens), 14° × 10° (29 mm lens)
F-Number	f/1.3
Lens Identification	Automatic
Digital Zoom	1-4x continuous
Image Presentation and Modes	
Display	4", 640 × 480 pixel touch screen LCD with auto-rotation
Digital Camera	5 MP, 53° × 41° FOV
Color Palettes	Iron, Gray, Rainbow, Arctic, Lava, Rainbow HC
Image Modes	Infrared, visual, MSX®, Picture-in-Picture
MSX®	Embosses visual details on full resolution thermal image
Measurement and Analysis	
Accuracy	±2°C (±3.6°F) or ±2% of reading for ambient temperature 15°C to 35°C (59°F to 95°F) and object temperature above 0°C (32°F)
Alarms	Moisture alarm, insulation alarm, measurement alarms
Color Alarm (Isotherm)	Above/below/interval/condensation/insulation
Compass, GPS	Yes; automatic GPS image tagging
METERLiNK®	Yes; several readings
Laser Pointer	Yes; dedicated button

Image Storage	
Storage Media	Removable SD card (8 GB)
Image File Format	Standard JPEG with measurement data included
Video Recording and Streaming	
Radiometric IR Video Recording	Real-time radiometric recording (.csq)
Non-Radiometric IR or Visual Video	H.264 to memory card
Radiometric IR Video Streaming	Yes, over UVC or Wi-Fi
Non-Radiometric IR Video Streaming	H.264 or MPEG-4 over Wi-Fi; MJPEG over UVC or Wi-Fi
Communication Interfaces	USB 2.0, Bluetooth, Wi-Fi, DisplayPort
Video Out	DisplayPort over USB Type-C
Additional Data	
Battery Type	Li-ion battery, charged in camera or on separate charger
Battery Operating Time	Approx. 2.5 hours at 25°C (77°F) ambient temperature and typical use
Operating Temperature Range	-15°C to 50°C (5°F to 122°F)
Storage Temperature Range	-40°C to 70°C (-40°F to 158°F)
Shock/Vibration/ Encapsulation; Safety	25 g / IEC 60068-2-27, 2 g / IEC 60068-2-6, IP 54 /IEC 60529; EN/UL/CSA/PSE 60950-1
Weight/Dimensions	1 kg (2.2 lbs), 27.8 × 11.6 × 11.3 cm (11.0 × 4.6 × 4.4 in)
Box Contents	
	Infrared camera with lens, battery (2 ea), battery charger, front protection, straps (hand, wrist), hard transport case, lanyards, lens caps, lens cleaning cloth, power supplies, 8 GB SD card, Torx wrench, cables (USB 2.0 A to USB Type-C, USB Type-C to USB Type-C, USB Type-C to HDMI)

*Hot spot to center spot Delta measurement

Specifications are subject to change without notice.
For the most up-to-date specs, go to www.flir.com

CORPORATE HEADQUARTERS
FLIR Systems, Inc.
27700 SW Parkway Ave.
Wilsonville, OR 97070
PH: +1 877.773.3547

LATIN AMERICA
FLIR Systems Brasil
Av. Antonio Bardella, 320
Sorocaba, SP 18085-852
Brasil
PH: +55 15 3238 7080

CHINA
FLIR Systems Co., Ltd
Rm 1613-16, Tower II
Grand Central Plaza
138 Shatin Rural Committee Rd.
Shatin, New Territories
Hong Kong
PH: +852 2792 8955

EUROPE
FLIR Systems, Inc.
Luxemburgstraat 2
2321 Meer
Belgium
PH: +32 (0) 3665 5100

www.flir.com
NASDAQ: FLIR

Equipment described herein may require US Government authorization for export purposes. Diversion contrary to US law is prohibited. Imagery for illustration purposes only. Specifications are subject to change without notice. ©2018 FLIR Systems, Inc. All rights reserved. (01/18)

17-3307-INS-Exx BLD

FLIR | The World's Sixth Sense®

Figura A3-0-1. Especificaciones técnicas de cámaras termográficas FLIR Exx-Series. Las características de la cámara utilizada (E85) se encuentran en la tercera columna (Extraído de [61])

ANEXO IV: CÓDIGO EMPLEADO PARA LA DETECCIÓN DE MATERIALES

```
# BLOQUE 1: Configuración e Importación
from google.colab import drive

# Montamos Google Drive
drive.mount('/content/drive')
print("✅ Drive montado correctamente.")
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
# BLOQUE 2: Procesar imágenes
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from tqdm import tqdm

def procesar_csv_termico():
    print("--- ⚙️ CONFIGURACIÓN DEL PROCESO ---")

    # 1. Solicitar Ruta
    ruta_archivo = input("1. Pega la ruta completa de tu archivo .csv en Drive: ").strip()
    # Eliminar comillas si el usuario las pone por error
    ruta_archivo = ruta_archivo.replace('"', '').replace("'", "")

    if not os.path.exists(ruta_archivo):
        print(f"❌ Error: No encuentro el archivo en: {ruta_archivo}")
        return

    # 2. Solicitar Parámetros de la Cámara (CRUCIAL para separar las imágenes apiladas)
    print("\n--- 📷 DATOS DE LA IMAGEN ---")
    print("Necesito saber la resolución de la FLIR E85 (o la del recorte si lo has editado).")
    try:
        filas_img = int(input("Número de FILAS (alto) de una sola imagen (ej. 288) -- HAY QUE RESTAR UNA UNIDAD (1): "))
        cols_img = int(input("Número de COLUMNAS (ancho) de una sola imagen (ej. 384): "))
        fps = float(input("FPS de la grabación original (ej. 30): "))
        segundos_intervalo = float(input("Intervalo deseado en SEGUNDOS para guardar (ej. 1): "))

        # Opcional: Saltar encabezados del CSV
        # ¡IMPORTANTE! Vuelve a introducir aquí el número de filas de encabezado (texto/metadata) que tiene tu CSV.
        # Si el error "Expected N fields" persiste, es muy probable que este valor sea incorrecto.
    except ValueError:
        print("❌ Error: Debes introducir números válidos.")
        return

    # --- PREPARACIÓN DE CARPETAS ---
    directorio_base = os.path.dirname(ruta_archivo)
    nombre_archivo = os.path.splitext(os.path.basename(ruta_archivo))[0]

    folder_imagenes = os.path.join(directorio_base, f"{nombre_archivo}_IMAGENES")
    folder_csv_final = os.path.join(directorio_base, f"{nombre_archivo}_RESULTADOS")

    os.makedirs(folder_imagenes, exist_ok=True)
    os.makedirs(folder_csv_final, exist_ok=True)
    print(f"\n📂 Guardando resultados en:\n - {folder_imagenes}\n - {folder_csv_final}")

    # --- CARGA DE DATOS ---
    print("\n⏳ Leyendo archivo CSV de entrada (esto puede tardar si es muy pesado)...")
    try:
        # Se cambia sep=None a sep=';' para forzar el separador de columnas a punto y coma.
        # El parámetro 'decimal=',' está configurado para interpretar la coma como separador decimal.
        df_raw = pd.read_csv(ruta_archivo, sep=';', skiprows = 4, on_bad_lines='warn', engine='python', header=None, encodi
    except Exception as e:
        print(f"❌ Error leyendo el CSV: {e}")
        return
```

```
# --- Modificaciones solicitadas: Eliminar la primera columna ---
# Asegúrate de que 'skip_rows' se haya configurado correctamente para las filas iniciales.
#df_raw = df_raw.iloc[3:, :]
df_raw = df_raw.iloc[:, 1:]
print("✅ Se ha eliminado la primera columna del CSV (las filas iniciales se saltan con 'skip_rows').")
# ----------------------------------------------------------------------------------

total_filas_csv = df_raw.shape[0]
total_cols_csv = df_raw.shape[1]

# Verificación de integridad
if total_cols_csv != cols_img:
    print(f"⚠️ ¡ADVERTENCIA!: El CSV tiene {total_cols_csv} columnas, pero dijiste que la imagen tiene {cols_img}.")
    print("El código intentará continuar, pero revisa si la resolución es correcta.")

# Calculamos cuántos frames (imágenes) hay en total apilados verticalmente
total_frames_encontrados = total_filas_csv // filas_img
print(f"📊 Filas totales CSV: {total_filas_csv}. Frames detectados: {total_frames_encontrados}")

# Calcular salto de frames según el intervalo pedido
frames_a_saltar = int(fps * segundos_intervalo)
if frames_a_saltar < 1: frames_a_saltar = 1

print(f"ℹ️ Se procesará 1 imagen cada {frames_a_saltar} frames originales.")

# --- PROCESAMIENTO ---
matriz_final_columnas = []
encabezados_tiempo = []

# Iteramos sobre los bloques de filas
for i in tqdm(range(0, total_frames_encontrados, frames_a_saltar), desc="Procesando imágenes"):

    # Calcular tiempo actual
    tiempo_actual = i / fps

    # Definir índice de inicio y fin de filas para ESTA imagen
    start_row = i * filas_img
    end_row = start_row + filas_img

    # Extraer el bloque de datos correspondiente a una imagen (DataFrame -> Numpy Array)
    img_data = df_raw.iloc[start_row:end_row, :cols_img].to_numpy(dtype=float)

    # 1. GUARDAR COMO IMAGEN VISUAL (.jpg)
    # Necesitamos normalizar los datos de temperatura a 0-255 para que sea una imagen visible
    img_norm = (img_data - np.min(img_data)) / (np.max(img_data) - np.min(img_data))
    img_uint8 = (img_norm * 255).astype(np.uint8)

    # Se cambia el formato para que el tiempo sea un entero
    nombre_foto = f"img_{i:05d}_t{int(tiempo_actual/60)}min.jpg"
    ruta_foto = os.path.join(folder_imagenes, nombre_foto)

    # Usamos colormap 'inferno' o 'gray' para termografía
    plt.imsave(ruta_foto, img_uint8, cmap='inferno')

    # 2. PROCESAR PARA MATRIZ FINAL (Columnar)
    # .flatten() en numpy lee de izquierda a derecha y baja fila a fila.
    columna_datos = img_data.flatten()

    matriz_final_columnas.append(columna_datos)
    encabezados_tiempo.append(f"t_{tiempo_actual/60}min")

# --- GUARDAR CSV FINAL ---
print("\n💾 Generando matriz final transpuesta...")

if matriz_final_columnas:
    # column_stack convierte la lista de arrays (filas) en columnas verticales
    matriz_np = np.column_stack(matriz_final_columnas)

    df_resultado = pd.DataFrame(matriz_np, columns=encabezados_tiempo)

    path_resultado = os.path.join(folder_csv_final, f"{nombre_archivo}_PROCESADO_MATRIZ.csv")
    df_resultado.to_csv(path_resultado, index=False)

    print("\n✅ ¡ÉXITO!")
    print(f"📄 Matriz guardada: {path_resultado}")
    print(f"📐 Tamaño matriz: {df_resultado.shape} (Filas [pixeles] x Columnas [tiempo])")
    print(f"🖼️ Imágenes guardadas en carpeta: {folder_imagenes}")
```

```
    else:
        print("⚠️ No se generaron datos. Revisa los parámetros de entrada.")

# EJECUTAR
procesar_csv_termico()
```

```
# BLOQUE 3: Análisis Estadístico y Gráfico de Evolución Temporal

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns # Opcional, mejora la estética
from tqdm import tqdm

def analizar_y_graficar_matrices():
    print("--- 🌡️ ANÁLISIS DE EVOLUCIÓN TÉRMICA ---")
    # 1. Solicitar ruta de la carpeta donde están los CSVs
    ruta_carpeta = input("Introduce la ruta de la carpeta donde están las matrices CSV: ").strip()
    ruta_carpeta = ruta_carpeta.replace('"', '').replace("'", "") # Limpieza de comillas

    if not os.path.exists(ruta_carpeta):
        print(f"❌ Error: La carpeta no existe: {ruta_carpeta}")
        return

    # Buscar archivos CSV en la carpeta
    archivos_csv = [f for f in os.listdir(ruta_carpeta) if f.endswith('.csv') and 'matriz' in f.lower()]

    if not archivos_csv:
        # Intento secundario por si el nombre no contiene 'matriz'
        archivos_csv = [f for f in os.listdir(ruta_carpeta) if f.endswith('.csv')]

    if len(archivos_csv) == 0:
        print("❌ No se encontraron archivos CSV en esa carpeta.")
        return

    print(f"✅ Se han encontrado {len(archivos_csv)} archivos para analizar.")

    # --- FILTRADO POR INTERVALO DE TIEMPO (OPCIONAL) ---
    print("\n--- FILTRADO POR INTERVALO DE TIEMPO (OPCIONAL) ---")
    tiempo_min = None
    tiempo_max = None
    while True:
        try:
            respuesta_filtro = input("¿Deseas aplicar un filtro por intervalo de tiempo (en minutos)? (s/n): ").strip().lo
            if respuesta_filtro == 's':
                tiempo_min = float(input("  Introduce el tiempo MÍNIMO en minutos (ej. 5.0): "))
                tiempo_max = float(input("  Introduce el tiempo MÁXIMO en minutos (ej. 15.0): "))
                if tiempo_min >= tiempo_max:
                    print("  ❌ El tiempo mínimo debe ser menor que el tiempo máximo.")
                else:
                    break
            elif respuesta_filtro == 'n':
                break
            else:
                print("  ❌ Entrada inválida. Por favor, responde 's' para sí o 'n' para no.")
        except ValueError:
            print("  ❌ Entrada inválida. Por favor, introduce un número válido.")

    # 2. Preparar carpeta de salida para el gráfico
    folder_suffix = ""
    if tiempo_min is not None and tiempo_max is not None:
        folder_suffix = f"_{int(tiempo_min)}min_{int(tiempo_max)}min"

    folder_graficas = os.path.join(ruta_carpeta, f"GRAFICAS_Y_CSV_EVOLUCION{folder_suffix}")
    os.makedirs(folder_graficas, exist_ok=True)
    print(f"\n✅ Carpeta de resultados: {folder_graficas}")

    # 3. Configuración del Gráfico Combinado
    plt.figure(figsize=(12, 7))
    plt.title("Evolución de la Temperatura Media e Intervalo de Confianza", fontsize=16)
    plt.xlabel("Tiempo (minutos)", fontsize=12)
    plt.ylabel("Valor del pixel (Temperatura (°C))", fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.6)
```

```
# Paleta de colores para diferenciar curvas si hay muchas
colores = plt.cm.tab10(np.linspace(0, 1, len(archivos_csv)))

# Lista para almacenar todas las estadísticas recopiladas para el CSV final
all_collected_stats = []
# Lista para almacenar datos para los gráficos individuales
all_plot_data = []

# --- BUCLE DE PROCESAMIENTO ---
# Procesamos archivo por archivo para no saturar la RAM

for idx, archivo in enumerate(tqdm(archivos_csv, desc="Analizando matrices")):
    ruta_completa = os.path.join(ruta_carpeta, archivo)
    nombre_etiqueta = os.path.splitext(archivo)[0] # Nombre para la leyenda

    try:
        # Leer CSV
        df = pd.read_csv(ruta_completa)

        # --- GESTIÓN DEL EJE TEMPORAL ---
        # Intentamos convertir los headers "t_0.50s" a números float
        # Si los headers no tienen ese formato, usamos el índice numérico
        try:
            # Extraemos el número entre 't_' y 's'
            tiempos = [float(col.replace('t_', '').replace('min', '')) for col in df.columns]
        except ValueError:
            # Si falla, usamos números consecutivos 0, 1, 2...
            tiempos = range(len(df.columns))

        # Aplicar filtrado de tiempo si está configurado
        if tiempo_min is not None and tiempo_max is not None:
            columnas_a_mantener_list = []
            tiempos_filtered = []
            for t, col_name in zip(tiempos, df.columns):
                if tiempo_min <= t <= tiempo_max:
                    columnas_a_mantener_list.append(col_name)
                    tiempos_filtered.append(t)

            if not columnas_a_mantener_list:
                print(f"  ⚠️ No se encontraron datos para '{nombre_etiqueta}' en el intervalo [{tiempo_min}, {tiempo_m
                continue # Saltar este archivo si no hay columnas en el rango

            df_filtered = df[columnas_a_mantener_list]
            tiempos = tiempos_filtered

            # Actualizar df para las operaciones subsiguientes
            df = df_filtered

            print(f"  ✅ Datos de '{nombre_etiqueta}' filtrados para el intervalo [{tiempo_min}, {tiempo_max}] minutos

        # --- CÁLCULO DE ESTADÍSTICAS ---
        # Calculamos media y desviación columna por columna (axis=0)
        media = df.mean(axis=0)
        desviacion = df.std(axis=0)

        # --- DIBUJAR EN EL GRÁFICO COMBINADO ---
        color_actual = colores[idx]

        # 1. Dibujar la línea de la MEDIA
        plt.plot(tiempos, media, label=nombre_etiqueta, color=color_actual, linewidth=2)

        # 2. Dibujar la FRANJA (Media + Desv, Media - Desv)
        limite_superior = media + desviacion
        limite_inferior = media - desviacion

        plt.fill_between(tiempos, limite_inferior, limite_superior, color=color_actual, alpha=0.2) # Alpha es la trans

        # Almacenar las estadísticas para el CSV final
        temp_stats_df = pd.DataFrame({
            'Tiempo': tiempos,
            f'{nombre_etiqueta}_Media': media,
            f'{nombre_etiqueta}_Desviacion': desviacion
        })
        all_collected_stats.append(temp_stats_df)
```

```
            # Almacenar datos para los gráficos individuales
            all_plot_data.append({
                'name': nombre_etiqueta,
                'tiempos': tiempos,
                'media': media,
                'limite_superior': limite_superior,
                'limite_inferior': limite_inferior,
                'color': color_actual
            })

        except Exception as e:
            print(f"\n⚠️ Error procesando {archivo}: {e}")
            continue

# Añadir leyenda al gráfico combinado
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
plt.tight_layout()

# 4. Guardar Gráfico Combinado
nombre_guardado_combinado = os.path.join(folder_graficas, "Comparativa_Evolucion_Termica.png")
plt.savefig(nombre_guardado_combinado, dpi=300)
plt.show() # Mostrar en pantalla
plt.close() # Cerrar para liberar memoria
print(f"\n✅ ¡Gráfico combinado generado exitosamente!")
print(f"✅ Guardado en: {nombre_guardado_combinado}")

# Función auxiliar para crear y guardar gráficos individuales
def _create_and_save_single_plot(data_type, folder, data_list):
    plt.figure(figsize=(12, 7))
    title_map = {
        'Media': 'Evolución de la Temperatura Media',
        'Limite_Superior': 'Evolución de la Temperatura (Media + Desviación Típica)',
        'Limite_Inferior': 'Evolución de la Temperatura (Media - Desviación Típica)'
    }
    plt.title(title_map[data_type], fontsize=16)
    plt.xlabel("Tiempo (minutos)", fontsize=12)
    plt.ylabel("Valor del pixel (Temperatura (°C))", fontsize=12)
    plt.grid(True, linestyle='--', alpha=0.6)

    for item in data_list:
        if data_type == 'Media':
            plt.plot(item['tiempos'], item['media'], label=item['name'], color=item['color'], linewidth=2)
        elif data_type == 'Limite_Superior':
            plt.plot(item['tiempos'], item['limite_superior'], label=item['name'], color=item['color'], linewidth=2, l
        elif data_type == 'Limite_Inferior':
            plt.plot(item['tiempos'], item['limite_inferior'], label=item['name'], color=item['color'], linewidth=2, l

    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0.)
    plt.tight_layout()

    filename_map = {
        'Media': 'Evolucion_Temperatura_Media.png',
        'Limite_Superior': 'Evolucion_Temperatura_Media_Mas_Desviacion.png',
        'Limite_Inferior': 'Evolucion_Temperatura_Media_Menos_Desviacion.png'
    }
    path_to_save = os.path.join(folder, filename_map[data_type])
    plt.savefig(path_to_save, dpi=300)
    plt.show() # Mostrar en pantalla
    plt.close() # Cerrar para liberar memoria
    print(f"✅ Gráfico '{data_type}' guardado en: {path_to_save}")

# 5. Crear y guardar los gráficos individuales
if all_plot_data:
    print("\n--- Generando gráficos individuales ---")
    _create_and_save_single_plot('Media', folder_graficas, all_plot_data)
    _create_and_save_single_plot('Limite_Superior', folder_graficas, all_plot_data)
    _create_and_save_single_plot('Limite_Inferior', folder_graficas, all_plot_data)
else:
    print("⚠️ No hay datos suficientes para generar gráficos individuales.")

# Opcional: Unir y guardar todas las estadísticas en un solo CSV
if all_collected_stats:
    # Merge all dataframes on 'Tiempo' to create a single table of statistics
    final_stats_df = all_collected_stats[0]
    for i in range(1, len(all_collected_stats)):
        final_stats_df = pd.merge(final_stats_df, all_collected_stats[i], on='Tiempo', how='outer')
```

```
        stats_path = os.path.join(folder_graficas, "Estadisticas_Comparativa_Evolucion.csv")
        final_stats_df.to_csv(stats_path, index=False)
        print(f"\n✅ Estadísticas guardadas en: {stats_path}")

# EJECUTAR LA FUNCIÓN
analizar_y_graficar_matrices()
```

```
# BLOQUE 4: Deep Learning. Cargar datos de entrada y etiquetar salidas clase

from google.colab import drive

# Montamos Google Drive
drive.mount('/content/drive')
print("✅ Drive montado correctamente.")
3
import pandas as pd
import numpy as np
import os # Import the os module

def load_stats_and_get_labels():
    # 1. Solicitar ruta completa del archivo CSV
    stats_csv_path = input("1. Pega la ruta completa del archivo 'Estadisticas_Comparativa_Evolucion.csv': ").strip()
    stats_csv_path = stats_csv_path.replace('"', '').replace("'", "") # Limpieza de comillas

    if not os.path.exists(stats_csv_path):
        print(f"❌ Error: No encuentro el archivo en: {stats_csv_path}")
        return None, None, None

    # 2. Cargar el archivo CSV en un DataFrame de pandas
    try:
        df_stats = pd.read_csv(stats_csv_path)
        print(f"✅ Archivo '{os.path.basename(stats_csv_path)}' cargado correctamente.")
    except Exception as e:
        print(f"❌ Error al cargar el CSV: {e}")
        return None, None, None

    # 3. Identificar los nombres de los materiales
    material_names = set()
    for col in df_stats.columns:
        if '_Media' in col:
            material_names.add(col.replace('_Media', ''))
        elif '_Desviacion' in col:
            material_names.add(col.replace('_Desviacion', ''))

    unique_material_names = sorted(list(material_names))

    if not unique_material_names:
        print("⚠️ No se pudieron identificar nombres de materiales en las columnas del CSV.")
        return df_stats, {}, 0

    print("\n--- Definición de Clases ---")
    print("Por favor, introduce una etiqueta de clase numérica (ej. 0, 1, 2...) para cada material. RESTAR UNA UNIDAD A CAI

    # 4. Inicializar diccionario para etiquetas de clase
    material_labels = {}

    # 5. Iterar sobre cada material para solicitar la etiqueta de clase
    for material in unique_material_names:
        while True:
            try:
                label = int(input(f"  Introduce la etiqueta de clase para '{material}': "))
                material_labels[material] = label
                break
            except ValueError:
                print("  ❌ Entrada inválida. Por favor, introduce un número entero.")

    # Imprimir los materiales y sus etiquetas para verificación
    print("\n--- Materiales y sus Etiquetas Asignadas ---")
    for material, label in material_labels.items():
        print(f"  Material: '{material}' -> Etiqueta: {label}")
```

```
    # 6. Determinar el número total de clases únicas
    num_classes = len(set(material_labels.values()))

    print(f"\n✅ Etiquetas de clase asignadas: {material_labels}")
    print(f"✅ Número total de clases únicas identificadas: {num_classes}")

    return df_stats, material_labels, num_classes

df_stats, material_labels, num_classes = load_stats_and_get_labels()

if df_stats is not None:
    print("\nPrimeras 5 filas del DataFrame de estadísticas:")
    print(df_stats.head())
```

```
# BLOQUE 4.1: Definir Ruta Base Global para Resultados

import os

# Solicitar al usuario la ruta base en Google Drive para guardar todos los resultados
base_experiment_dir = input("Por favor, introduce la ruta COMPLETA de la CARPETA en Google Drive donde se guardar\u00e1n toc

# Eliminar cualquier comilla simple o doble de la ruta ingresada
base_experiment_dir = base_experiment_dir.replace("'", '').replace('"', '')

# Crear el directorio base si no existe
os.makedirs(base_experiment_dir, exist_ok=True)

print(f"\u2705 Ruta base para los resultados del experimento establecida y creada (si no exist\u00eda): {base_experiment_dir
```

```
# BLOQUE 5: Deep Learning. Cálculo de los ratios delta temperatura/delta tiemppo

import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import os

def perform_feature_engineering_and_prepare_inputs(df_stats, material_labels):
    # Verificar que los datos necesarios están cargados
    if df_stats is None or material_labels is None:
        print("❌ Error: El DataFrame 'df_stats' o las etiquetas de material 'material_labels' no están disponibles.\n" +
              "Por favor, asegúrate de que la celda anterior para 'Cargar Datos Estadísticos y Definir Clases' se ejecutó c
        return None, None, None, None, None, None, None, None

    print("\n--- ⚙️ INGENIERÍA DE CARACTERÍSTICAS Y PREPARACIÓN DE ENTRADAS ---")
    # 1. Preguntar si desea calcular la diferencia de temperatura en valores absolutos
    while True:
        respuesta = input("2. ¿Desea calcular la diferencia de temperatura (T_t - T_{t-1}) en valores absolutos? (s/n): ").
        if respuesta == 's':
            abs_diff_temp = True
            break
        elif respuesta == 'n':
            abs_diff_temp = False
            break
        else:
            print("  ❌ Entrada inválida. Por favor, responda 's' para sí o 'n' para no.")

    # 2. Preguntar el intervalo de tiempo (delta_t)
    while True:
        try:
            delta_t = float(input("3. Introduce el intervalo de tiempo (delta_t) en minutos utilizado para generar las est
            if delta_t <= 0:
                print("  ❌ El intervalo de tiempo debe ser un número positivo.")
            else:
                break
        except ValueError:
            print("  ❌ Entrada inválida. Por favor, introduce un número.")

    # Construir el directorio de salida del experimento
    # Asegurarse de que base_experiment_dir es una variable global o se pasa como argumento
    global base_experiment_dir # Acceder a la variable global definida en BLOQUE 4.1

    # Nombre de la carpeta del experimento basado en los parámetros
    experiment_folder_name = f"Experimento_dt{delta_t:.1f}min_abs{abs_diff_temp}"
    experiment_output_root = os.path.join(base_experiment_dir, experiment_folder_name)
```

```
os.makedirs(experiment_output_root, exist_ok=True)
print(f"\n✅ Directorio de resultados del experimento: {experiment_output_root}")

# 3. Identificar los nombres únicos de los materiales a partir de material_labels
unique_material_names = sorted(list(material_labels.keys()))
if not unique_material_names:
    print("⚠️ No se pudieron identificar nombres de materiales a partir de las etiquetas. Revisar 'material_labels'.")
    return None, None, None, None, None, None, None, None

# Helper function to get the correct column name from df_stats
def get_df_column_name(material_name_from_labels, stat_type): # stat_type is 'Media' or 'Desviacion'
    if '_Variacion' in material_name_from_labels:
        # For names like 'prueba_2_PROCESADO_MATRIZ_Variacion1'
        # We need to form column name like 'prueba_2_PROCESADO_MATRIZ_Media_Variacion1'
        parts = material_name_from_labels.split('_Variacion')
        base_part = '_Variacion'.join(parts[:-1])
        variation_suffix = '_Variacion' + parts[-1]
        return f'{base_part}_{stat_type}{variation_suffix}'
    else:
        return f'{material_name_from_labels}_{stat_type}'

derivadas_media = []
derivadas_media_mas_desv = []
derivadas_media_menos_desv = []

print("\nCalculando derivadas temporales para cada material...")
for material in unique_material_names:
    # Get the correct column names based on the DataFrame structure
    media_col_name = get_df_column_name(material, 'Media')
    desv_col_name = get_df_column_name(material, 'Desviacion')

    # Extraer series de tiempo
    media_series = df_stats[media_col_name]
    desv_series = df_stats[desv_col_name]

    # Combinar para 'Media + Desviación' y 'Media - Desviación'
    media_mas_desv_series = media_series + desv_series
    media_menos_desv_series = media_series - desv_series

    # Calcular derivadas (diferencia de un paso de tiempo) y dividir por delta_t
    # .diff().dropna() se usa para eliminar el primer NaN resultante de la diferencia
    derivada_media = media_series.diff().dropna() / delta_t
    derivada_media_mas_desv = media_mas_desv_series.diff().dropna() / delta_t
    derivada_media_menos_desv = media_menos_desv_series.diff().dropna() / delta_t

    # Aplicar valor absoluto si el usuario lo solicita
    if abs_diff_temp:
        derivada_media = np.abs(derivada_media)
        derivada_media_mas_desv = np.abs(derivada_media_mas_desv)
        derivada_media_menos_desv = np.abs(derivada_media_menos_desv)

    derivadas_media.append(derivada_media.to_numpy())
    derivadas_media_mas_desv.append(derivada_media_mas_desv.to_numpy())
    derivadas_media_menos_desv.append(derivada_media_menos_desv.to_numpy())

# Verificar y ajustar la longitud de la secuencia (todos deben tener la misma longitud después de .diff())
if derivadas_media:
    sequence_length = len(derivadas_media[0])
    for i in range(len(unique_material_names)):
        if len(derivadas_media[i]) != sequence_length or \
           len(derivadas_media_mas_desv[i]) != sequence_length or \
           len(derivadas_media_menos_desv[i]) != sequence_length:
            print("⚠️ Advertencia: Las series derivadas tienen longitudes inconsistentes. Esto no debería ocurrir.\n"
                  "Se truncarán a la longitud mínima para asegurar la consistencia.")
            min_len = min(len(d) for d in derivadas_media + derivadas_media_mas_desv + derivadas_media_menos_desv)
            if min_len < sequence_length:
                sequence_length = min_len
                derivadas_media = [d[:min_len] for d in derivadas_media]
                derivadas_media_mas_desv = [d[:min_len] for d in derivadas_media_mas_desv]
                derivadas_media_menos_desv = [d[:min_len] for d in derivadas_media_menos_desv] # Corrected typo here
            print(f"Longitud de secuencia ajustada a: {sequence_length}")
else:
    print("❌ No se generaron series derivadas. Verifique los datos de entrada.")
    return None, None, None, None, None, None, None, None

# Normalizar cada una de las series derivadas utilizando escalado Min-Max (entre 0 y 1)
scaler = MinMaxScaler()
```

```
    derivadas_media_norm = []
    derivadas_media_mas_desv_norm = []
    derivadas_media_menos_desv_norm = []

    print("Normalizando series derivadas (Min-Max scaling)...")
    for i in range(len(unique_material_names)):
        # Reshape para el scaler (espera un array 2D) y luego aplanarlo de nuevo
        derivadas_media_norm.append(scaler.fit_transform(derivadas_media[i].reshape(-1, 1)).flatten())
        derivadas_media_mas_desv_norm.append(scaler.fit_transform(derivadas_media_mas_desv[i].reshape(-1, 1)).flatten())
        derivadas_media_menos_desv_norm.append(scaler.fit_transform(derivadas_media_menos_desv[i].reshape(-1, 1)).flatten(

    # 4. Preparar las entradas univariadas
    # Forma deseada: (num_samples, sequence_length, 1)
    print("Preparando entradas univariadas...")
    X_uni_media = np.array(derivadas_media_norm).reshape(-1, sequence_length, 1)
    X_uni_media_plus_desv = np.array(derivadas_media_mas_desv_norm).reshape(-1, sequence_length, 1)
    X_uni_media_minus_desv = np.array(derivadas_media_menos_desv_norm).reshape(-1, sequence_length, 1)

    # 5. Preparar las entradas multivariadas
    # Forma deseada: (num_samples, sequence_length, 3)
    print("Preparando entradas multivariadas...")
    X_multi = []
    for i in range(len(unique_material_names)):
        # Apilar las tres series derivadas normalizadas para cada material
        combined_features = np.column_stack([
            derivadas_media_norm[i],
            derivadas_media_mas_desv_norm[i],
            derivadas_media_menos_desv_norm[i]
        ])
        X_multi.append(combined_features)
    X_multi = np.array(X_multi)

    print("\n✅ ¡Ingeniería de características completada y entradas preparadas!")
    print(f"Shape de X_uni_media: {X_uni_media.shape}")
    print(f"Shape de X_uni_media_plus_desv: {X_uni_media_plus_desv.shape}")
    print(f"Shape de X_uni_media_minus_desv: {X_uni_media_minus_desv.shape}")
    print(f"Shape de X_multi: {X_multi.shape}")

    return X_uni_media, X_uni_media_plus_desv, X_uni_media_minus_desv, X_multi, unique_material_names, material_labels, se

# Ejecutar la función de ingeniería de características
# Asumimos que df_stats, material_labels y num_classes han sido definidos en la celda anterior.
# Si la celda anterior falló, df_stats y material_labels serán None, y la función lo reportará.
X_uni_media, X_uni_media_plus_desv, X_uni_media_minus_desv, X_multi, unique_material_names_fe, material_labels_fe, sequenc
```

```
# BLOQUE 6: Deep Learning. Preparación clase salida tras su carga previa antes del inicio del aprendizaje

# Instalar TensorFlow si no está instalado
try:
    import tensorflow as tf
except ImportError:
    print("TensorFlow no encontrado. Instalando...")
    !pip install tensorflow
    import tensorflow as tf
import numpy as np

def preprocess_labels_for_classification(material_labels, unique_material_names_fe, num_classes):
    # 1. Verificar que las variables necesarias están disponibles
    if material_labels is None or num_classes is None or not unique_material_names_fe:
        print("❌ Error: Las variables 'material_labels', 'unique_material_names_fe' o 'num_classes' no están disponibles
              "Por favor, asegúrate de que la celda anterior para 'Cargar Datos Estadísticos y Definir Clases' se ejecutó
        return None

    print("\n--- 🎯 PREPROCESAMIENTO DE ETIQUETAS --- ")

    # 2. Extraer las etiquetas numéricas de material_labels
    # Aseguramos el orden de las etiquetas según unique_material_names_fe
    numerical_labels = [material_labels[name] for name in unique_material_names_fe]

    # Convertir a array de numpy
    numerical_labels_array = np.array(numerical_labels)

    # 3. Convertir a formato one-hot encoding
    # Usamos to_categorical de Keras para esto
    Y_encoded = tf.keras.utils.to_categorical(numerical_labels_array, num_classes=num_classes)
```

```
    print(f"✅ Etiquetas numéricas obtenidas: {numerical_labels_array}")
    print(f"✅ Etiquetas convertidas a One-Hot Encoding. Shape: {Y_encoded.shape}")
    print("Primeras 5 etiquetas codificadas:\n", Y_encoded[:5])

    return Y_encoded

# Ejecutar la función de preprocesamiento de etiquetas
# Asumimos que material_labels, unique_material_names_fe y num_classes han sido definidos en celdas anteriores.
Y_encoded = preprocess_labels_for_classification(material_labels, unique_material_names_fe, num_classes)
```

```
# BLOQUE 7: Deep Learning. Preparación conjunto de datos para entrenamiento, validación y prueba (testeo)

import numpy as np
from sklearn.model_selection import train_test_split

def split_data_for_training_validation_testing(X_uni_media, X_uni_media_plus_desv, X_uni_media_minus_desv, X_multi, Y_enco
    # Verificar que los datos necesarios están cargados
    if X_uni_media is None or Y_encoded is None or not unique_material_names_fe:
        print("❌ Error: Las entradas (X) o las etiquetas codificadas (Y_encoded) no están disponibles.\n" +
              "Por favor, asegúrate de que las celdas anteriores se ejecutaron correctamente.")
        return None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None

    print("\n--- À DIVISIÓN DE DATOS (ENTRENAMIENTO, VALIDACIÓN, PRUEBA) ---")

    num_samples = X_uni_media.shape[0]
    sample_indices = np.arange(num_samples)

    # Usamos Y_encoded para la estratificación, tomando la clase dominante si es one-hot
    stratify_labels = np.argmax(Y_encoded, axis=1)

    # 1. Solicitar números de muestras al usuario
    while True:
        try:
            # Predefined values to avoid KeyboardInterrupt during automated execution
            # Total samples is 72. Let's aim for 48 train, 12 validation, 12 test for an example split.
            num_train_desired = 48 # int(input("1. Introduce el NÚMERO DE MUESTRAS para el conjunto de ENTRENAMIENTO: "))
            num_val_desired = 12  # int(input("2. Introduce el NÚMERO DE MUESTRAS para el conjunto de VALIDACIÓN: "))
            num_test_desired = 12 # int(input("3. Introduce el NÚMERO DE MUESTRAS para el conjunto de PRUEBA: "))

            if num_train_desired < 0 or num_val_desired < 0 or num_test_desired < 0:
                print("  ❌ Los números de muestras no pueden ser negativos.")
            elif (num_train_desired + num_val_desired + num_test_desired) != num_samples:
                print(f"  ❌ La suma de las muestras ({num_train_desired + num_val_desired + num_test_desired}) no coincid
                print("  Por favor, asegúrate de que la suma sea igual al número total de muestras.")
            else:
                break
        except ValueError:
            print("  ❌ Entrada inválida. Por favor, introduce un número entero válido.")

    print(f"\nTotal de muestras disponibles: {num_samples}")
    print(f"Muestras solicitadas - Entrenamiento: {num_train_desired}, Validación: {num_val_desired}, Prueba: {num_test_de

    # Check if stratification is possible for the requested split
    class_counts = np.bincount(stratify_labels)
    perform_stratification = True

    # A more robust check for stratification possibility:
    # For 3 splits, if a class has 1 sample, it cannot be stratified. If it has 2, it might struggle.
    # The train_test_split function will raise an error if it truly cannot stratify.
    if np.min(class_counts) < 2 and (num_val_desired > 0 or num_test_desired > 0) and num_train_desired > 0:
        print("⚠️ Advertencia: Algunas clases tienen muy pocas muestras. La estratificación podría no ser efectiva o falla
        # Decided to still try stratification and let `train_test_split` handle potential errors for now

    print("\nRealizando división de datos...")

    # Step 1: Split into Training and (Validation + Test)
    test_size_temp = num_val_desired + num_test_desired

    try:
        X_train_idx, X_temp_idx, Y_train_strat, Y_temp_strat = train_test_split(
            sample_indices,
            stratify_labels,
            train_size=num_train_desired,
            test_size=test_size_temp,
            stratify=stratify_labels if perform_stratification else None,
```

```
            random_state=42
        )
    except ValueError as e:
        print(f"❌ Error durante la primera división (Entrenamiento vs. Temp): {e}")
        print("Esto puede ocurrir si la estratificación es imposible con las muestras solicitadas.")
        print("Intente reducir el número de conjuntos de validación/prueba o ajuste los tamaños.")
        return None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None

    # Step 2: Split (Validation + Test) into Validation and Test
    if num_val_desired + num_test_desired > 0:
        try:
            X_val_idx, X_test_idx, Y_val_strat, Y_test_strat = train_test_split(
                X_temp_idx,
                stratify_labels[X_temp_idx],
                train_size=num_val_desired,
                test_size=num_test_desired,
                stratify=stratify_labels[X_temp_idx] if perform_stratification else None,
                random_state=42
            )
        except ValueError as e:
            print(f"❌ Error durante la segunda división (Validación vs. Prueba): {e}")
            print("Esto puede ocurrir si la estratificación es imposible con las muestras solicitadas.")
            print("Intente reducir el número de conjuntos de validación/prueba o ajuste los tamaños.")
            return None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, Nc
    else:
        X_val_idx = np.array([])
        X_test_idx = np.array([])
        Y_val_strat = np.array([])
        Y_test_strat = np.array([])

    # Aplicar los índices a todas las matrices X y a Y_encoded
    X_train_uni_media = X_uni_media[X_train_idx]
    X_val_uni_media = X_uni_media[X_val_idx]
    X_test_uni_media = X_uni_media[X_test_idx]

    X_train_uni_media_plus_desv = X_uni_media_plus_desv[X_train_idx]
    X_val_uni_media_plus_desv = X_uni_media_plus_desv[X_val_idx]
    X_test_uni_media_plus_desv = X_uni_media_plus_desv[X_test_idx]

    X_train_uni_media_minus_desv = X_uni_media_minus_desv[X_train_idx]
    X_val_uni_media_minus_desv = X_uni_media_minus_desv[X_val_idx]
    X_test_uni_media_minus_desv = X_uni_media_minus_desv[X_test_idx]

    X_train_multi = X_multi[X_train_idx]
    X_val_multi = X_multi[X_val_idx]
    X_test_multi = X_multi[X_test_idx]

    Y_train = Y_encoded[X_train_idx]
    Y_val = Y_encoded[X_val_idx]
    Y_test = Y_encoded[X_test_idx]

    print("✅ División de datos completada con éxito!")
    print(f"  - Conjunto de Entrenamiento: {len(X_train_idx)} muestras")
    print(f"  - Conjunto de Validación: {len(X_val_idx)} muestras")
    print(f"  - Conjunto de Prueba: {len(X_test_idx)} muestras")

    # Mostrar los nombres de los materiales para cada conjunto
    print("\nNombres de materiales en el conjunto de Entrenamiento:")
    for idx in X_train_idx:
        print(f"  - {unique_material_names_fe[idx]}")

    print("\nNombres de materiales en el conjunto de Validación:")
    for idx in X_val_idx:
        print(f"  - {unique_material_names_fe[idx]}")

    print("\nNombres de materiales en el conjunto de Prueba:")
    for idx in X_test_idx:
        print(f"  - {unique_material_names_fe[idx]}")

    return (
        X_train_uni_media, X_val_uni_media, X_test_uni_media,
        X_train_uni_media_plus_desv, X_val_uni_media_plus_desv, X_test_uni_media_plus_desv,
        X_train_uni_media_minus_desv, X_val_uni_media_minus_desv, X_test_uni_media_minus_desv,
        X_train_multi, X_val_multi, X_test_multi,
        Y_train, Y_val, Y_test,
        X_train_idx, X_val_idx, X_test_idx
    )
```

```
# Ejecutar la función de división de datos
# Asumimos que X_uni_media, X_uni_media_plus_desv, X_uni_media_minus_desv, X_multi,
# Y_encoded, unique_material_names_fe han sido definidos en celdas anteriores.
(
    X_train_uni_media, X_val_uni_media, X_test_uni_media,
    X_train_uni_media_plus_desv, X_val_uni_media_plus_desv, X_test_uni_media_plus_desv,
    X_train_uni_media_minus_desv, X_val_uni_media_minus_desv, X_test_uni_media_minus_desv,
    X_train_multi, X_val_multi, X_test_multi,
    Y_train, Y_val, Y_test,
    X_train_idx, X_val_idx, X_test_idx
) = split_data_for_training_validation_testing(
    X_uni_media, X_uni_media_plus_desv, X_uni_media_minus_desv, X_multi,
    Y_encoded, unique_material_names_fe
)
```

```
# BLOQUE 8: Deep Learning. Preparación modelos de Deep Learning a usar (RNN, LSTM y GRU)


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, LSTM, GRU, Dense, Dropout
from tensorflow.keras.optimizers import Adam

def build_rnn_model(model_type, input_shape, num_classes, rnn_units=64, dense_units=32, dropout_rate=0.2):
    model = Sequential([
        tf.keras.Input(shape=input_shape) # Define Input layer explicitly
    ])

    if model_type == 'SimpleRNN':
        model.add(SimpleRNN(rnn_units, return_sequences=False))
    elif model_type == 'LSTM':
        model.add(LSTM(rnn_units, return_sequences=False))
    elif model_type == 'GRU':
        model.add(GRU(rnn_units, return_sequences=False))
    else:
        raise ValueError("Tipo de modelo RNN no soportado. Debe ser 'SimpleRNN', 'LSTM' o 'GRU'.")

    model.add(Dense(dense_units, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(num_classes, activation='softmax')) # Salida softmax para clasificación multiclase

    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Asumiendo que num_classes y sequence_length_fe ya están definidos de celdas anteriores
if 'num_classes' not in locals() or 'sequence_length_fe' not in locals():
    print("❌ Error: Las variables 'num_classes' o 'sequence_length_fe' no están definidas. Asegúrate de ejecutar las celd
else:
    print(f"Número de clases identificado: {num_classes}")
    print(f"Longitud de secuencia de características: {sequence_length_fe}")

    # ================================================================
    # Definición y compilación de los 9 modelos univariados
    # ================================================================
    print("\n--- Definiendo Modelos Univariados ---")
    models_univariate = {}
    input_types_uni = {
        'Media': (X_train_uni_media.shape[1], X_train_uni_media.shape[2]), # (sequence_length, 1)
        'Media_plus_Desv': (X_train_uni_media_plus_desv.shape[1], X_train_uni_media_plus_desv.shape[2]),
        'Media_minus_Desv': (X_train_uni_media_minus_desv.shape[1], X_train_uni_media_minus_desv.shape[2])
    }
    rnn_types = ['SimpleRNN', 'LSTM', 'GRU']

    for rnn_type in rnn_types:
        for input_name, input_shape in input_types_uni.items():
            model_name = f'{rnn_type}_{input_name}'
            print(f"Creando modelo: {model_name} con input_shape={input_shape}")
            model = build_rnn_model(rnn_type, input_shape, num_classes)
            models_univariate[model_name] = model
            model.summary() # Descomentar para ver el resumen de cada modelo si es necesario

    # ================================================================
    # Definición y compilación de los 3 modelos multivariados
    # ================================================================
    print("\n--- Definiendo Modelos Multivariados ---")
```

```
    models_multivariate = {}
    input_shape_multi = (X_train_multi.shape[1], X_train_multi.shape[2]) # (sequence_length, 3)

    for rnn_type in rnn_types:
        model_name = f'{rnn_type}_Multi'
        print(f"Creando modelo: {model_name} con input_shape={input_shape_multi}")
        model = build_rnn_model(rnn_type, input_shape_multi, num_classes)
        models_multivariate[model_name] = model
        # model.summary() # Descomentar para ver el resumen de cada modelo si es necesario

    print("\n✅ Todos los 12 modelos (9 univariados y 3 multivariados) han sido definidos y compilados.")
```

```
# BLOQUE 9: Deep Learning. Aprendizaje de los modelos de Deep Learning a usar (RNN, LSTM y GRU)


import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, LSTM, GRU, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint

def build_rnn_model(model_type, input_shape, num_classes, rnn_units=64, dense_units=32, dropout_rate=0.2):
    model = Sequential([
        tf.keras.Input(shape=input_shape) # Define Input layer explicitly
    ])

    if model_type == 'SimpleRNN':
        model.add(SimpleRNN(rnn_units, return_sequences=False))
    elif model_type == 'LSTM':
        model.add(LSTM(rnn_units, return_sequences=False))
    elif model_type == 'GRU':
        model.add(GRU(rnn_units, return_sequences=False))
    else:
        raise ValueError("Tipo de modelo RNN no soportado. Debe ser 'SimpleRNN', 'LSTM' o 'GRU'.")

    model.add(Dense(dense_units, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(num_classes, activation='softmax')) # Salida softmax para clasificación multiclase

    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Re-define models_univariate and models_multivariate here to ensure they exist
# Asumiendo que num_classes y sequence_length_fe ya están definidos de celdas anteriores
if 'num_classes' not in globals() or 'sequence_length_fe' not in globals() or \
   'X_train_uni_media' not in globals() or 'X_train_multi' not in globals():
    print("❌ Error: Las variables necesarias (num_classes, sequence_length_fe, X_train_uni_media, X_train_multi) no están
    # Initialize empty dictionaries to avoid further NameErrors during execution
    models_univariate = {}
    models_multivariate = {}
else:
    print(f"Número de clases identificado: {num_classes}")
    print(f"Longitud de secuencia de características: {sequence_length_fe}")

    # ================================================================
    # Definición y compilación de los 9 modelos univariados
    # ================================================================
    print("\n--- Definiendo Modelos Univariados ---")
    models_univariate = {}
    input_types_uni = {
        'Media': (X_train_uni_media.shape[1], X_train_uni_media.shape[2]), # (sequence_length, 1)
        'Media_plus_Desv': (X_train_uni_media_plus_desv.shape[1], X_train_uni_media_plus_desv.shape[2]),
        'Media_minus_Desv': (X_train_uni_media_minus_desv.shape[1], X_train_uni_media_minus_desv.shape[2])
    }
    rnn_types = ['SimpleRNN', 'LSTM', 'GRU']

    for rnn_type in rnn_types:
        for input_name, input_shape in input_types_uni.items():
            model_name = f'{rnn_type}_{input_name}'
            print(f"Creando modelo: {model_name} con input_shape={input_shape}")
            model = build_rnn_model(rnn_type, input_shape, num_classes)
            models_univariate[model_name] = model
            # model.summary() # Descomentar para ver el resumen de cada modelo si es necesario

    # ================================================================
```

```
    # Definición y compilación de los 3 modelos multivariados
    # ============================================================
    print("\n--- Definiendo Modelos Multivariados ---")
    models_multivariate = {}
    input_shape_multi = (X_train_multi.shape[1], X_train_multi.shape[2]) # (sequence_length, 3)

    for rnn_type in rnn_types:
        model_name = f'{rnn_type}_Multi'
        print(f"Creando modelo: {model_name} con input_shape={input_shape_multi}")
        model = build_rnn_model(rnn_type, input_shape_multi, num_classes)
        models_multivariate[model_name] = model
        # model.summary() # Descomentar para ver el resumen de cada modelo si es necesario

    print("\n✅ Todos los 12 modelos (9 univariados y 3 multivariados) han sido definidos y compilados.")

def train_models(models_univariate, models_multivariate,
                 X_train_uni_media, X_val_uni_media,
                 X_train_uni_media_plus_desv, X_val_uni_media_plus_desv,
                 X_train_uni_media_minus_desv, X_val_uni_media_minus_desv,
                 X_train_multi, X_val_multi,
                 Y_train, Y_val, experiment_output_root):

    # Verificar que los datos necesarios están cargados y no están vacíos
    if (not models_univariate and not models_multivariate) or \
       X_train_uni_media is None or X_val_uni_media is None or \
       Y_train is None or Y_val is None:
        print("❌ Error: Los modelos o los conjuntos de datos de entrenamiento/validación no están disponibles o están vac
              "Asegúrate de que las celdas anteriores se ejecutaron correctamente y generaron datos válidos.")
        return {}, {}

    print("\n--- 🧠 ENTRENANDO LOS MODELOS --- ")

    epochs = 100 # Número de epochs ajustado para datasets pequeños
    batch_size = 1 # Batch size pequeño, adecuado para datasets pequeños para evitar NaN en gradientes si el dataset es mu

    trained_histories = {}

    # Directorio para guardar los mejores modelos, ahora dentro de experiment_output_root
    best_models_dir = os.path.join(experiment_output_root, 'best_models')
    os.makedirs(best_models_dir, exist_ok=True)

    # Entrenamiento de modelos univariados
    for model_name, model in models_univariate.items():
        print(f"\nEntrenando modelo univariado: {model_name}...")

        # Correctly identify input type from model_name
        if 'Media_plus_Desv' in model_name:
            input_type = 'Media_plus_Desv'
            X_train = X_train_uni_media_plus_desv
            X_val = X_val_uni_media_plus_desv
        elif 'Media_minus_Desv' in model_name:
            input_type = 'Media_minus_Desv'
            X_train = X_train_uni_media_minus_desv
            X_val = X_val_uni_media_minus_desv
        elif 'Media' in model_name:
            input_type = 'Media'
            X_train = X_train_uni_media
            X_val = X_val_uni_media
        else:
            print(f"⚠️ Tipo de entrada desconocido para el modelo univariado {model_name}. Saltando.")
            continue

        # Callbacks
        early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
        model_checkpoint = ModelCheckpoint(filepath=os.path.join(best_models_dir, f'{model_name}_best.keras'),
                                           monitor='val_loss', save_best_only=True, verbose=0)

        history = model.fit(X_train, Y_train,
                            epochs=epochs,
                            batch_size=batch_size,
                            validation_data=(X_val, Y_val),
                            callbacks=[early_stopping, model_checkpoint],
                            verbose=0) # verbose=0 para no imprimir cada epoch
        trained_histories[model_name] = history
        print(f"  ✅ Entrenamiento de {model_name} completado. Mejor val_loss: {min(history.history['val_loss']):.4f}")
```

```
    # Entrenamiento de modelos multivariados
    for model_name, model in models_multivariate.items():
        print(f"\nEntrenando modelo multivariado: {model_name}...")

        X_train = X_train_multi
        X_val = X_val_multi

        # Callbacks
        early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
        model_checkpoint = ModelCheckpoint(filepath=os.path.join(best_models_dir, f'{model_name}_best.keras'),
                                           monitor='val_loss', save_best_only=True, verbose=0)

        history = model.fit(X_train, Y_train,
                            epochs=epochs,
                            batch_size=batch_size,
                            validation_data=(X_val, Y_val),
                            callbacks=[early_stopping, model_checkpoint],
                            verbose=0)
        trained_histories[model_name] = history
        print(f"  ✅ Entrenamiento de {model_name} completado. Mejor val_loss: {min(history.history['val_loss']):.4f}")

    print("\n✅ ¡Entrenamiento de todos los modelos finalizado!")
    print(f"Los mejores modelos se han guardado en: {os.path.abspath(best_models_dir)}")
    return trained_histories, best_models_dir

# Ejecutar la función de entrenamiento
trained_histories, best_models_dir = train_models(models_univariate, models_multivariate,
                                                  X_train_uni_media, X_val_uni_media,
                                                  X_train_uni_media_plus_desv, X_val_uni_media_plus_desv,
                                                  X_train_uni_media_minus_desv, X_val_uni_media_minus_desv,
                                                  X_train_multi, X_val_multi,
                                                  Y_train, Y_val, experiment_output_root)
```

```
# BLOQUE 10: Deep Learning. Establecer ruta para guardado del aprendizaje de los modelos de Deep Learning a usar (RNN, LSTI

import os
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix

print("✅ Librerías importadas correctamente.")

# Define la ruta base para guardar los resultados de la evaluación
# Ahora se usa experiment_output_root que viene del BLOQUE 5
evaluation_results_dir = os.path.join(experiment_output_root, 'evaluation_plots')
os.makedirs(evaluation_results_dir, exist_ok=True)
print(f"✅ Directorio para resultados de evaluación creado: {evaluation_results_dir}")

# best_models_dir ya está definida en el BLOQUE 9 y apunta a la ruta correcta.
```

```
# BLOQUE 11: Deep Learning. Preparación previa para guardado del aprendizaje de los modelos de Deep Learning a usar (RNN, I

import re # Importar el módulo de expresiones regulares
import pandas as pd # Asegurarse de que pandas esté importado

def get_model_data(model_name):
    """
    Returns the appropriate X_train, X_val, X_test datasets for a given model_name.
    """
    if 'Media_plus_Desv' in model_name:
        return X_train_uni_media_plus_desv, X_val_uni_media_plus_desv, X_test_uni_media_plus_desv
    elif 'Media_minus_Desv' in model_name:
        return X_train_uni_media_minus_desv, X_val_uni_media_minus_desv, X_test_uni_media_minus_desv
    elif 'Media' in model_name and 'Multi' not in model_name:
        return X_train_uni_media, X_val_uni_media, X_test_uni_media
    elif 'Multi' in model_name:
        return X_train_multi, X_val_multi, X_test_multi
    else:
        raise ValueError(f"Tipo de modelo o entrada no reconocida para {model_name}")

def plot_history(history, model_name, plot_dir):
    """
    Generates and saves Loss vs. Epoch and Accuracy vs. Epoch plots.
    """
```

```
    plt.figure(figsize=(12, 5))

    # Plot Loss
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Loss Entrenamiento')
    plt.plot(history.history['val_loss'], label='Loss Validación')
    plt.title(f'Pérdida vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Pérdida')
    plt.legend()
    plt.grid(True)
    # Plot Accuracy
    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Accuracy Entrenamiento')
    plt.plot(history.history['val_accuracy'], label='Accuracy Validación')
    plt.title(f'Precisión vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Precisión')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.savefig(os.path.join(plot_dir, f'{model_name}_Performance_Plot.png'))
    plt.close() # Close plot to free memory
    print(f"  -> Gráfico de rendimiento guardado para {model_name}")

def create_and_save_prediction_table(sample_indices, y_pred_prob, all_material_names, true_labels_numerical, num_classes, f
    """
    Genera un DataFrame con las probabilidades de predicción de cada clase para cada muestra,
    incluyendo la clase real y el nombre del material, y lo guarda como un archivo CSV.
    """
    # Obtener el nombre real del material para cada muestra
    material_names_for_samples = [all_material_names[idx] for idx in sample_indices]

    # Crear columnas para las probabilidades de cada clase
    prob_cols = [f'Prob_Clase_{i}' for i in range(num_classes)]

    # Convertir las probabilidades a DataFrame y redondear a 2 decimales
    df_probs = pd.DataFrame(y_pred_prob, columns=prob_cols).round(2)

    # Añadir la clase real y la clase predicha (la de mayor probabilidad)
    df_probs['Clase Real'] = true_labels_numerical
    df_probs['Clase Predicha'] = y_pred_prob.argmax(axis=1)

    # Insertar el nombre del material al principio
    df_probs.insert(0, 'Nombre del Material', material_names_for_samples)

    # Guardar el DataFrame en un archivo CSV
    file_path = os.path.join(plot_dir, filename)
    df_probs.to_csv(file_path, index=False)

    print(f"  -> Tabla de predicciones guardada: {filename}")
    print(f"  -> Las primeras 5 filas de la tabla de predicciones:\n{df_probs.head()}")

def shorten_material_names(full_names):
    shortened = []
    for name in full_names:
        # Updated regex to capture the part between 'Enfriamiento' and '_PROCESADO'
        # It handles both 'Enfriamiento ' and 'Enfriamiento_' as separators.
        match = re.search(r'Enfriamiento[ _](.*?)_PROCESADO', name)
        if match:
            short_name = match.group(1).strip() # Extract and strip whitespace
        else:
            # Fallback if the pattern is not found. Try a more general cleanup, or original name.
            # This fallback should ideally not be hit if the pattern is consistent.
            short_name = name.replace('_PROCESADO_MATRIZ', '').replace('Enfriamiento_', '').strip()
            # Another common pattern is 'A_Enfriamiento_X', we might want just 'X'
            if short_name.startswith('A_') or short_name.startswith('B_') or \
               short_name.startswith('C_') or short_name.startswith('D_') or \
               short_name.startswith('E_') or short_name.startswith('F_'):
                if len(short_name) > 2 and short_name[1] == '_':
                    short_name = short_name[2:]

        shortened.append(short_name)
    return shortened
```

```
print("✅ Funciones auxiliares para evaluación definidas.")
```

```
# BLOQUE 12: Deep Learning. Guardado del aprendizaje de los modelos de Deep Learning a usar (RNN, LSTM y GRU)

import re # Importar el módulo de expresiones regulares
import pandas as pd # Asegurarse de que pandas esté importado
import shutil # Importar shutil para copiar archivos
import numpy as np # Necesario para np.argmin
import matplotlib.pyplot as plt


def get_model_data(model_name):
    """
    Returns the appropriate X_train, X_val, X_test datasets for a given model_name.
    """
    if 'Media_plus_Desv' in model_name:
        return X_train_uni_media_plus_desv, X_val_uni_media_plus_desv, X_test_uni_media_plus_desv
    elif 'Media_minus_Desv' in model_name:
        return X_train_uni_media_minus_desv, X_val_uni_media_minus_desv, X_test_uni_media_minus_desv
    elif 'Media' in model_name and 'Multi' not in model_name:
        return X_train_uni_media, X_val_uni_media, X_test_uni_media
    elif 'Multi' in model_name:
        return X_train_multi, X_val_multi, X_test_multi
    else:
        raise ValueError(f"Tipo de modelo o entrada no reconocida para {model_name}")

def plot_history(history, model_name, plot_dir):
    """
    Generates and saves Loss vs. Epoch and Accuracy vs. Epoch plots.
    """
    plt.figure(figsize=(12, 5))

    # Plot Loss
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Loss Entrenamiento')
    plt.plot(history.history['val_loss'], label='Loss Validación')
    plt.title(f'Pérdida vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Pérdida')
    plt.legend()
    plt.grid(True)

    # Plot Accuracy
    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Accuracy Entrenamiento')
    plt.plot(history.history['val_accuracy'], label='Accuracy Validación')
    plt.title(f'Precisión vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Precisión')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    file_path = os.path.join(plot_dir, f'{model_name}_Performance_Plot.png')
    plt.savefig(file_path)
    plt.close() # Close plot to free memory
    print(f"  -> Gráfico de rendimiento guardado para {model_name}")
    return file_path

def create_and_save_prediction_table(sample_indices, y_pred_prob, all_material_names, true_labels_numerical, num_classes,
    """
    Genera un DataFrame con las probabilidades de predicción de cada clase para cada muestra,
    incluyendo la clase real y el nombre del material, y lo guarda como un archivo CSV.
    """
    # Obtener el nombre real del material para cada muestra
    material_names_for_samples = [all_material_names[idx] for idx in sample_indices]

    # Crear columnas para las probabilidades de cada clase
    prob_cols = [f'Prob_Clase_{i}' for i in range(num_classes)]

    # Convertir las probabilidades a DataFrame y redondear a 2 decimales
    df_probs = pd.DataFrame(y_pred_prob, columns=prob_cols).round(2)

    # Añadir la clase real y la clase predicha (la de mayor probabilidad)
    df_probs['Clase Real'] = true_labels_numerical
    df_probs['Clase Predicha'] = y_pred_prob.argmax(axis=1)
```

```
    # Insertar el nombre del material al principio
    df_probs.insert(0, 'Nombre del Material', material_names_for_samples)

    # Guardar el DataFrame en un archivo CSV
    file_path = os.path.join(plot_dir, filename)
    df_probs.to_csv(file_path, index=False)

    print(f"  -> Tabla de predicciones guardada: {filename}")
    # print(f"  -> Las primeras 5 filas de la tabla de predicciones:\n{df_probs.head()}") # Comentado para reducir la salid
    return file_path

def shorten_material_names(full_names):
    shortened = []
    for name in full_names:
        # Updated regex to capture the part between 'Enfriamiento' and '_PROCESADO'
        # It handles both 'Enfriamiento ' and 'Enfriamiento_' as separators.
        match = re.search(r'Enfriamiento[ _](.*?)_PROCESADO', name)
        if match:
            short_name = match.group(1).strip() # Extract and strip whitespace
        else:
            # Fallback if the pattern is not found. Try a more general cleanup, or original name.
            # This fallback should ideally not be hit if the pattern is consistent.
            short_name = name.replace('_PROCESADO_MATRIZ', '').replace('Enfriamiento_', '').strip()
            # Another common pattern is 'A_Enfriamiento_X', we might want just 'X'
            if short_name.startswith('A_') or short_name.startswith('B_') or \
               short_name.startswith('C_') or short_name.startswith('D_') or \
               short_name.startswith('E_') or short_name.startswith('F_'):
                if len(short_name) > 2 and short_name[1] == '_':
                    short_name = short_name[2:]

        shortened.append(short_name)
    return shortened



print("\n--- EVALUANDO LOS MODELOS --- ")

# Obtener nombres de las clases para las tablas de predicción
# unique_material_names_fe contiene los nombres originales de los materiales, que son nuestras clases.
full_class_names = unique_material_names_fe
# short_class_names = shorten_material_names(full_class_names) # Not directly used in plotting/tables here, but useful for

# Create a list of all possible integer labels (0 to num_classes-1)
all_possible_labels = list(range(num_classes))

# Crear un mapeo de etiqueta numérica a un nombre corto representativo
numeric_label_to_short_name = {}
for full_name, numeric_label in material_labels.items(): # material_labels is available from Block 4
    # Only add if the label hasn't been mapped yet, to pick one representative name per label
    if numeric_label not in numeric_label_to_short_name:
        # Use the already defined shorten_material_names function
        numeric_label_to_short_name[numeric_label] = shorten_material_names([full_name])[0]

# Asegurarse de que los nombres para mostrar estén ordenados por etiqueta numérica
display_label_names = [numeric_label_to_short_name[i] for i in sorted(numeric_label_to_short_name.keys())]

print(f"\nNombres abreviados para mostrar en las tablas de predicción (por clase): {display_label_names}")

# Asegurarse de que Y_train, Y_val, Y_test están disponibles
# Si no están disponibles por algún error anterior, esta comprobación los crearía como arrays vacíos
# para evitar errores de referencia, aunque el flujo normal debería tenerlos ya.
if 'Y_train' not in locals() or Y_train is None:
    print("❌ Error: Y_train no está definido o es nulo. Verifique la celda de división de datos.")
    Y_train_labels = np.array([])
    Y_val_labels = np.array([])
    Y_test_labels = np.array([])
else:
    Y_train_labels = np.argmax(Y_train, axis=1)
    Y_val_labels = np.argmax(Y_val, axis=1)
    Y_test_labels = np.argmax(Y_test, axis=1)

# Diccionario para combinar todos los modelos (univariados y multivariados) para facilitar la iteración
all_models = {**models_univariate, **models_multivariate}
```

```
# Initialize variables for tracking the best model
best_val_loss = float('inf')
best_model_name = None
best_model_files = {
    'plot': '',
    'train_predictions': '',
    'val_predictions': '',
    'test_predictions': ''
}

for model_name, model_obj in all_models.items():
    print(f"\nProcesando modelo: {model_name}")

    # 1. Cargar el mejor modelo guardado
    try:
        loaded_model = tf.keras.models.load_model(os.path.join(best_models_dir, f'{model_name}_best.keras'))
        print(f"  ✅ Modelo '{model_name}' cargado correctamente.")
    except Exception as e:
        print(f"  ❌ Error al cargar el modelo '{model_name}': {e}. Saltando evaluación.")
        continue

    # 2. Obtener los conjuntos de datos X apropiados
    try:
        X_train, X_val, X_test = get_model_data(model_name)
        print(f"  ✅ Datos de entrada obtenidos para '{model_name}'.")
    except ValueError as e:
        print(f"  ❌ Error al obtener datos para '{model_name}': {e}. Saltando evaluación.")
        continue

    # Verificar que los conjuntos de datos no están vacíos
    if X_train.size == 0 or X_val.size == 0 or X_test.size == 0:
        print(f"  ⚠️ Uno o más conjuntos de datos (Train, Val, Test) para '{model_name}' están vacíos. No se puede evaluar
        continue

    # 3. Generar y guardar gráficos de rendimiento (Loss y Accuracy)
    current_plot_file = ''
    if model_name in trained_histories:
        current_plot_file = plot_history(trained_histories[model_name], model_name, evaluation_results_dir)
        # Check if current model has a better validation loss
        current_val_loss = min(trained_histories[model_name].history['val_loss'])
        if current_val_loss < best_val_loss:
            best_val_loss = current_val_loss
            best_model_name = model_name
            best_model_files['plot'] = current_plot_file
    else:
        print(f"  ⚠️ Historial de entrenamiento no encontrado para '{model_name}'. No se generarán gráficos de rendimiento

    # 4. Realizar predicciones
    print(f"  Realizando predicciones para {model_name}...")
    y_pred_train_prob = loaded_model.predict(X_train, verbose=0)
    y_pred_val_prob = loaded_model.predict(X_val, verbose=0)
    y_pred_test_prob = loaded_model.predict(X_test, verbose=0)

    print(f"  ✅ Predicciones obtenidas para '{model_name}'.")

    # 5. Generar y guardar tablas de predicción
    print(f"  Generando tablas de predicción para {model_name}...")
    current_train_predictions_file = ''
    current_val_predictions_file = ''
    current_test_predictions_file = ''

    if Y_train_labels.size > 0:
        current_train_predictions_file = create_and_save_prediction_table(X_train_idx, y_pred_train_prob, unique_material_
                                                                          evaluation_results_dir, f'{model_name}_Predict
        if model_name == best_model_name: # Update if this is the best model so far
            best_model_files['train_predictions'] = current_train_predictions_file
    else:
        print("  ⚠️ No hay datos de entrenamiento válidos para generar tabla de predicción.")

    if Y_val_labels.size > 0:
        current_val_predictions_file = create_and_save_prediction_table(X_val_idx, y_pred_val_prob, unique_material_names_
                                                                        evaluation_results_dir, f'{model_name}_Predictio
        if model_name == best_model_name: # Update if this is the best model so far
            best_model_files['val_predictions'] = current_val_predictions_file
    else:
        print("  ⚠️ No hay datos de validación válidos para generar tabla de predicción.")
```

```
        if Y_test_labels.size > 0:
            current_test_predictions_file = create_and_save_prediction_table(X_test_idx, y_pred_test_prob, unique_material_name
                                                                             evaluation_results_dir, f'{model_name}_Predictio
            if model_name == best_model_name: # Update if this is the best model so far
                best_model_files['test_predictions'] = current_test_predictions_file
        else:
            print("  ⚠️ No hay datos de prueba válidos para generar tabla de predicción.")

    print("\n✅ Evaluación de todos los modelos completada.")
    print(f"Todos los gráficos de rendimiento y tablas de predicción se guardaron en: {os.path.abspath(evaluation_results_dir)}

    # --- SAVE BEST MODEL RESULTS TO A SEPARATE FOLDER ---
    if best_model_name:
        print(f"\n--- Identificando el mejor modelo: {best_model_name} (con val_loss: {best_val_loss:.4f}) ---")
        best_results_summary_dir = os.path.join(experiment_output_root, 'best_results_summary')
        os.makedirs(best_results_summary_dir, exist_ok=True)
        print(f"✅ Carpeta para el resumen del mejor modelo creada: {best_results_summary_dir}")

        # Copy prediction tables
        if best_model_files['train_predictions'] and os.path.exists(best_model_files['train_predictions']):
            shutil.copy(best_model_files['train_predictions'], os.path.join(best_results_summary_dir, 'best_model_Predictions_T
            print("  ✅ Tabla de predicciones de entrenamiento del mejor modelo copiada.")

        if best_model_files['val_predictions'] and os.path.exists(best_model_files['val_predictions']):
            shutil.copy(best_model_files['val_predictions'], os.path.join(best_results_summary_dir, 'best_model_Predictions_Val
            print("  ✅ Tabla de predicciones de validación del mejor modelo copiada.")

        if best_model_files['test_predictions'] and os.path.exists(best_model_files['test_predictions']):
            shutil.copy(best_model_files['test_predictions'], os.path.join(best_results_summary_dir, 'best_model_Predictions_Te
            print("  ✅ Tabla de predicciones de prueba del mejor modelo copiada.")
    else:
        print("\n⚠️ No se pudo identificar el mejor modelo o no hay modelos para evaluar.")
```

ANEXO V: CÓDIGO EMPLEADO PARA LA DETECCIÓN DE SALES

```
# BLOQUE 1: Configuración e Importación
from google.colab import drive

# Montamos Google Drive
drive.mount('/content/drive')
print("✅ Drive montado correctamente.")
```

```
# BLOQUE 2: Obtención Curvas de Sal. Modo tabla (csv)

import pandas as pd
import os
import re

def calculate_media_differences():
    print("\n--- CÁLCULO DE DIFERENCIAS ENTRE COLUMNAS '_Media' ---")

    # 1. Solicitar ruta del archivo CSV de estadísticas
    stats_csv_path = input("1. Pega la ruta COMPLETA del archivo 'Estadisticas_Comparativa_Evolucion.csv': ").strip()
    stats_csv_path = stats_csv_path.replace('"', '').replace("'", '') # Limpieza de comillas

    if not os.path.exists(stats_csv_path):
        print(f"❌ Error: No encuentro el archivo en: {stats_csv_path}")
        return

    # 2. Cargar el archivo CSV
    try:
        df_stats = pd.read_csv(stats_csv_path)
        print(f"✅ Archivo '{os.path.basename(stats_csv_path)}' cargado correctamente.")
    except Exception as e:
        print(f"❌ Error al cargar el CSV: {e}")
        return

    # 3. Identificar columnas que terminan en '_Media'
    media_cols = [col for col in df_stats.columns if col.endswith('_Media')]

    if not media_cols:
        print("❌ No se encontraron columnas que terminan en '_Media' en el archivo CSV.")
        return

    print("\nColumnas '_Media' identificadas:")
    for i, col in enumerate(media_cols):
        numeric_id = 'N/A'
        if 'Enfriamiento' in col:
            match = re.search(r'Enfriamiento[ _](.*?)(?:_PROCESADO_MATRIZ_Media|$)', col)
            if match:
                numeric_id = match.group(1).strip()
        elif 'Calentamiento' in col:
            match = re.search(r'Calentamiento[ _](.*?)(?:_PROCESADO_MATRIZ_Media|$)', col)
            if match:
                numeric_id = match.group(1).strip()
        print(f"  [{i}] ID: {numeric_id} -> {col}")

    # 4. Solicitar la parte numérica del ID de referencia
    while True:
        ref_numeric_part = input("\n2. Introduce la PARTE NUMÉRICA del IDENTIFICADOR (ej. '6') de la columna que deseas us
        if ref_numeric_part.isdigit(): # Ensure it's purely numeric
            break
        else:
            print("❌ Entrada inválida. Por favor, introduce solo la parte numérica del identificador (ej. '6').")

    # This is the target numeric part for all reference columns
    print(f"✅ Parte numérica de referencia seleccionada: {ref_numeric_part}")
```

```
# This is the target numeric part for all reference columns
print(f"✅ Parte numérica de referencia seleccionada: {ref_numeric_part}")

# Crear DataFrame para almacenar las diferencias
df_diff = pd.DataFrame({'Tiempo': df_stats['Tiempo']})
print("\nCalculando diferencias...")

# 5. Calcular diferencias dinámicamente
for col in media_cols:
    # 1. Extraer current_full_id (e.g., '5', 'A5', 'D5') from the current column name
    match_current_id = None
    series_type_prefix = ''
    if 'Enfriamiento' in col:
        match_current_id = re.search(r'Enfriamiento[ _](.*?)(?:_PROCESADO_MATRIZ_Media|$)', col)
        series_type_prefix = 'E_'
    elif 'Calentamiento' in col:
        match_current_id = re.search(r'Calentamiento[ _](.*?)(?:_PROCESADO_MATRIZ_Media|$)', col)
        series_type_prefix = 'C_'
    if not match_current_id:
        print(f"⚠️ No se pudo extraer el ID de la columna: {col}. Saltando.")
        continue
    current_full_id = match_current_id.group(1).strip()

    # 2. Parse current_full_id to get prefix and its numeric part
    match_id_parts = re.match(r'([A-Z]*)(\d+)', current_full_id) # Allow for no prefix [A-Z]*
    if not match_id_parts:
        print(f"⚠️ No se pudo parsear el ID '{current_full_id}' en prefijo y número. Saltando columna: {col}.")
        continue
    prefix = match_id_parts.group(1)

    # 3. Construct the dynamic reference ID (e.g., 'A6', '6', 'D6')
    dynamic_ref_full_id = prefix + ref_numeric_part

    # 4. Find the actual column name for this dynamic reference ID in media_cols
    actual_dynamic_ref_col_name = None
    # Check for both '_ID' and ' ID' patterns for robust matching
    if 'Enfriamiento' in col:
        ref_pattern_underscore = r'Enfriamiento_' + re.escape(dynamic_ref_full_id) + r'(?:_PROCESADO_MATRIZ_Media|$)'
        ref_pattern_space = r'Enfriamiento ' + re.escape(dynamic_ref_full_id) + r'(?:_PROCESADO_MATRIZ_Media|$)'
    elif 'Calentamiento' in col:
        ref_pattern_underscore = r'Calentamiento_' + re.escape(dynamic_ref_full_id) + r'(?:_PROCESADO_MATRIZ_Media|$)'
        ref_pattern_space = r'Calentamiento ' + re.escape(dynamic_ref_full_id) + r'(?:_PROCESADO_MATRIZ_Media|$)'
    else:
        print(f"⚠️ Tipo de serie (Enfriamiento/Calentamiento) no detectado para '{col}'. Saltando.")
        continue

    for ref_candidate_col in media_cols:
        if re.search(ref_pattern_underscore, ref_candidate_col) or \
           re.search(ref_pattern_space, ref_candidate_col):
            actual_dynamic_ref_col_name = ref_candidate_col
            break

    if not actual_dynamic_ref_col_name:
        print(f"⚠️ No se encontró la columna de referencia dinámica para el ID '{dynamic_ref_full_id}' (para restar de
        continue

    # Skip if the current column is the same as its own dynamic reference (e.g., A6 vs A6)
    if col == actual_dynamic_ref_col_name:
        print(f"  Skipping: '{col}' is the dynamic reference itself.")
        continue

    # 5. Perform the subtraction
    diff_col_name = f'Diferencia_{series_type_prefix}{current_full_id}_vs_{series_type_prefix}{dynamic_ref_full_id}'
    df_diff[diff_col_name] = df_stats[col] - df_stats[actual_dynamic_ref_col_name]
    print(f"  Calculada la diferencia para: {diff_col_name} (Restando '{actual_dynamic_ref_col_name}' de '{col}')")

# 6. Solicitar ruta para guardar el archivo CSV resultante
output_csv_path = input("\n3. Pega la ruta COMPLETA donde guardar el nuevo archivo CSV de diferencias (ej. /content/dr:
output_csv_path = output_csv_path.replace('"', '').replace("'", '') # Limpieza de comillas
```

```
    # Ensure the path ends with .csv
    if not output_csv_path.lower().endswith('.csv'):
        # If the path is a directory, append a default filename
        if os.path.isdir(output_csv_path):
            output_csv_path = os.path.join(output_csv_path, f"diferencias_vs_{ref_numeric_part}.csv")
            print(f"ℹ️ Se ha asumido un nombre de archivo. Guardando como: {output_csv_path}")
        else: # It's a file path but without .csv, so append it
            output_csv_path += '.csv'
            print(f"ℹ️ Se ha añadido la extensión .csv al archivo. Guardando como: {output_csv_path}")

    output_dir = os.path.dirname(output_csv_path)
    os.makedirs(output_dir, exist_ok=True)

    # Guardar el DataFrame de diferencias en un nuevo archivo CSV
    try:
        df_diff.to_csv(output_csv_path, index=False)
        print(f"\n✅ ¡ÉXITO! El archivo de diferencias se guardó correctamente en: {output_csv_path}")
        print("Primeras 5 filas del archivo de diferencias:\n")
        display(df_diff.head())
    except Exception as e:
        print(f"❌ Error al guardar el archivo CSV: {e}")

# Ejecutar la función
calculate_media_differences()
```

```
# BLOQUE 3: Deep Learning. Cargar datos de entrada y etiquetar salidas clase

from google.colab import drive

# Montamos Google Drive
drive.mount('/content/drive')
print("✅ Drive montado correctamente.")

import pandas as pd
import numpy as np
import os # Import the os module

def load_stats_and_get_labels():
    # 1. Solicitar ruta completa del archivo CSV
    stats_csv_path = input("1. Pega la ruta completa del archivo 'Curvas_Sal.csv': ").strip()
    stats_csv_path = stats_csv_path.replace('"', '').replace("'", "") # Limpieza de comillas

    if not os.path.exists(stats_csv_path):
        print(f"❌ Error: No encuentro el archivo en: {stats_csv_path}")
        return None, None, None

    # 2. Cargar el archivo CSV en un DataFrame de pandas
    try:
        df_stats = pd.read_csv(stats_csv_path)
        print(f"✅ Archivo '{os.path.basename(stats_csv_path)}' cargado correctamente.")
    except Exception as e:
        print(f"❌ Error al cargar el CSV: {e}")
        return None, None, None

    # 3. Identificar los nombres de los materiales
    material_names = set()
    for col in df_stats.columns:
        if 'Diferencia_' in col:
            material_names.add(col.replace('Diferencia_', ''))

    unique_material_names = sorted(list(material_names))

    if not unique_material_names:
        print("⚠️ No se pudieron identificar nombres de materiales en las columnas del CSV.")
        return df_stats, {}, 0

    print("\n--- Definición de Clases ---")
    print("Por favor, introduce una etiqueta de clase numérica (ej. 0, 1, 2...) para cada material. RESTAR UNA UNIDAD A CAI

    # 4. Inicializar diccionario para etiquetas de clase
    material_labels = {}
```

```
    # 5. Iterar sobre cada material para solicitar la etiqueta de clase
    for material in unique_material_names:
        while True:
            try:
                label = int(input(f"  Introduce la etiqueta de clase para '{material}': "))
                material_labels[material] = label
                break
            except ValueError:
                print("  ❌ Entrada inválida. Por favor, introduce un número entero.")

    # Imprimir los materiales y sus etiquetas para verificación
    print("\n--- Materiales y sus Etiquetas Asignadas ---")
    for material, label in material_labels.items():
        print(f"  Material: '{material}' -> Etiqueta: {label}")

    # 6. Determinar el número total de clases únicas
    num_classes = len(set(material_labels.values()))

    print(f"\n✅ Etiquetas de clase asignadas: {material_labels}")
    print(f"✅ Número total de clases únicas identificadas: {num_classes}")

    return df_stats, material_labels, num_classes

df_stats, material_labels, num_classes = load_stats_and_get_labels()
```

```
df_stats, material_labels, num_classes = load_stats_and_get_labels()

if df_stats is not None:
    print("\nPrimeras 5 filas del DataFrame de estadísticas:")
    print(df_stats.head())
```

```
# BLOQUE 4.1: Deep Learning. Definir Ruta Base Global para Resultados

import os

# Solicitar al usuario la ruta base en Google Drive para guardar todos los resultados
base_experiment_dir = input("Por favor, introduce la ruta COMPLETA de la CARPETA en Google Drive donde se guardar\u00e1n t

# Eliminar cualquier comilla simple o doble de la ruta ingresada
base_experiment_dir = base_experiment_dir.replace("'", '').replace('"', '')

# Crear el directorio base si no existe
os.makedirs(base_experiment_dir, exist_ok=True)

print(f"\u2705 Ruta base para los resultados del experimento establecida y creada (si no exist\u00eda): {base_experiment_d
```

```
# BLOQUE 4.2: Deep Learning. Preparación del formato y datos para el resultado

import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
import os
import math


def perform_feature_engineering_and_prepare_inputs(df_stats, material_labels):
    # Verificar que los datos necesarios están cargados
    if df_stats is None or material_labels is None:
        print("❌ Error: El DataFrame 'df_stats' o las etiquetas de material 'material_labels' no están disponibles.\n" +
              "Por favor, asegúrate de que la celda anterior para 'Cargar Datos Estadísticos y Definir Clases' se ejecutó co
        return None, None, None, None, None, None, None, None

    print("\n--- ⚙️ INGENIERÍA DE CARACTERÍSTICAS Y PREPARACIÓN DE ENTRADAS ---")

    # 1. Preguntar si desea calcular la diferencia de temperatura en valores absolutos
    while True:
        respuesta = input("2. ¿Desea calcular la diferencia de temperatura (T_t - T_{t-1}) en valores absolutos? (s/n): ").s
        if respuesta == 's':
            abs_diff_temp = True
            break
        elif respuesta == 'n':
            abs_diff_temp = False
            break
```

```
    else:
        print("  ❌ Entrada inválida. Por favor, responda 's' para sí o 'n' para no.")

# 2. Preguntar el intervalo de tiempo (delta_t)
while True:
    try:
        delta_t = float(input("3. Introduce el intervalo de tiempo (delta_t) en minutos utilizado para generar las estad
        if delta_t <= 0:
            print("  ❌ El intervalo de tiempo debe ser un número positivo.")
        else:
            break
    except ValueError:
        print("  ❌ Entrada inválida. Por favor, introduce un número.")

# 3. Preguntar si desea aplicar logaritmo base 10 SÓLO al numerador
apply_log10_numerator = False # Default value
while True:
    respuesta_log = input("4. ¿Desea aplicar el logaritmo base 10 SÓLO a la magnitud de la diferencia de temperatura (|T
    if respuesta_log == 's':
        apply_log10_numerator = True
        break
    elif respuesta_log == 'n':
        break
    else:
        print("  ❌ Entrada inválida. Por favor, responda 's' para sí o 'n' para no.")

# NEW PROMPT: Choose between 'ratios de temperatura (derivada)' or 'áreas acumuladas (integral)'
while True:
    feature_type_choice = input("5. ¿Qué tipo de característica desea utilizar: 'derivada' (ratios de temperatura) o 'in
    if feature_type_choice in ['derivada', 'integral']:
        break
    else:
        print("  ❌ Entrada inválida. Por favor, escriba 'derivada' o 'integral'.")


# Construir el directorio de salida del experimento
# Asegurarse de que base_experiment_dir es una variable global o se pasa como argumento
global base_experiment_dir # Acceder a la variable global definida en BLOQUE 4.1

# Nombre de la carpeta del experimento basado en los parámetros
log_suffix = "_logNumOnly" if apply_log10_numerator else "_noLog"
feature_type_suffix = f"_{feature_type_choice.capitalize()}"
experiment_folder_name = f"Experimento{feature_type_suffix}_dt{delta_t:.1f}min_abs{abs_diff_temp}{log_suffix}"
experiment_output_root = os.path.join(base_experiment_dir, experiment_folder_name)

os.makedirs(experiment_output_root, exist_ok=True)
print(f"\n✅ Directorio de resultados del experimento: {experiment_output_root}")

# 4. Identificar los nombres únicos de los materiales a partir de material_labels
unique_material_names = sorted(list(material_labels.keys()))
if not unique_material_names:
    print("⚠️ No se pudieron identificar nombres de materiales a partir de las etiquetas. Revisar 'material_labels'.")
    return None, None, None, None, None, None, None, None

# Helper function to get the correct column name from df_stats
def get_df_column_name(material_name_from_labels, stat_type): # stat_type is 'Diferencia'
    # The column names in df_stats are like 'Diferencia_1_vs_6'
    return f'{stat_type}_{material_name_from_labels}'

derivadas_media = []

print("\nCalculando sumas acumuladas de diferencias de temperatura para cada material...")
for material in unique_material_names:
    # Get the correct column names based on the DataFrame structure
    media_col_name = get_df_column_name(material, 'Diferencia')

    # Extraer series de tiempo
    media_series = df_stats[media_col_name]

    # Calcular diferencias (T_t - T_{t-1})
    diff_temps = media_series.diff().dropna()

    # Aplicar valor absoluto si el usuario lo solicita
    if abs_diff_temp:
        diff_temps = np.abs(diff_temps)
```

```
            if apply_log10_numerator:
                # Aplicar log10 a la magnitud de la diferencia de temperatura
                # Añadimos una pequeña constante (epsilon) para evitar log(0) o log(valores_muy_pequeños)
                diff_temps = np.log10(diff_temps + 1e-9) # Aplicar log a diff_temps directamente

            # Calculate feature based on user choice
            if feature_type_choice == 'derivada':
                # 'ratios de temperatura (derivada)': (T_t - T_{t-1}) / delta_t
                # diff_temps already is (T_t - T_{t-1})
                derivada_media = diff_temps / delta_t
            elif feature_type_choice == 'integral':
                # 'áreas acumuladas (integral)': (T_t - T_{t-1}).cumsum() * delta_t / 2 (Trapezoidal rule)
                derivada_media = diff_temps.cumsum() * delta_t / 2

            derivadas_media.append(derivada_media.to_numpy())

        # Verificar y ajustar la longitud de la secuencia (todos deben tener la misma longitud después de .diff())
        if derivadas_media:
            sequence_length = len(derivadas_media[0])
            for i in range(len(unique_material_names)):
                if len(derivadas_media[i]) != sequence_length:
                    print("⚠️ Advertencia: Las series derivadas tienen longitudes inconsistentes. Esto no debería ocurrir.\n" +
                          "Se truncarán a la longitud mínima para asegurar la consistencia.")
                    min_len = min(len(d) for d in derivadas_media)
                    if min_len < sequence_length:
                        sequence_length = min_len
                        derivadas_media = [d[:min_len] for d in derivadas_media]
                    print(f"Longitud de secuencia ajustada a: {sequence_length}")
        else:
            print("❌ No se generaron series derivadas. Verifique los datos de entrada.")
            return None, None, None, None, None, None, None, None

        # Normalizar cada una de las series derivadas utilizando escalado Min-Max (entre 0 y 1)
        scaler = MinMaxScaler()

        derivadas_media_norm = []

        print("Normalizando series derivadas (Min-Max scaling)...")
        for i in range(len(unique_material_names)):
            # Reshape para el scaler (espera un array 2D) y luego aplanarlo de nuevo
            derivadas_media_norm.append(scaler.fit_transform(derivadas_media[i].reshape(-1, 1)).flatten())

        # 5. Preparar las entradas univariadas
        # Forma deseada: (num_samples, sequence_length, 1)
        print("Preparando entradas univariadas...")
        X_uni_media = np.array(derivadas_media_norm).reshape(-1, sequence_length, 1)


        print("\n✅ ¡Ingeniería de características completada y entradas preparadas!")
        print(f"Shape de X_uni_media: {X_uni_media.shape}")

        return X_uni_media, unique_material_names, material_labels, sequence_length, experiment_output_root

# Ejecutar la función de ingeniería de características
# Asumimos que df_stats, material_labels y num_classes han sido definidos en la celda anterior.
# Si la celda anterior falló, df_stats y material_labels serán None, y la función lo reportará.
X_uni_media, unique_material_names_fe, material_labels_fe, sequence_length_fe, experiment_output_root = perform_feature_engi
```

```
# BLOQUE 5: Deep Learning. Preparación clase salida tras su carga previa antes del inicio del aprendizaje

# Instalar TensorFlow si no está instalado
try:
    import tensorflow as tf
except ImportError:
    print("TensorFlow no encontrado. Instalando...")
    !pip install tensorflow
    import tensorflow as tf

import numpy as np

def preprocess_labels_for_classification(material_labels, unique_material_names_fe, num_classes):
    # 1. Verificar que las variables necesarias están disponibles
    if material_labels is None or num_classes is None or not unique_material_names_fe:
        print("❌ Error: Las variables 'material_labels', 'unique_material_names_fe' o 'num_classes' no están disponibles
              "Por favor, asegúrate de que la celda anterior para 'Cargar Datos Estadísticos y Definir Clases' se ejecutó
        return None
```

```
    print("\n--- 🎯 PREPROCESAMIENTO DE ETIQUETAS --- ")

    # 2. Extraer las etiquetas numéricas de material_labels
    # Aseguramos el orden de las etiquetas según unique_material_names_fe
    numerical_labels = [material_labels[name] for name in unique_material_names_fe]

    # Convertir a array de numpy
    numerical_labels_array = np.array(numerical_labels)

    # La instrucción al usuario ya pide que se resten las unidades para obtener un índice 0, así que no se resta aquí.
    # Si el usuario introduce 0, 1, 2... se mantendrán así.

    # 3. Convertir a formato one-hot encoding
    # Usamos to_categorical de Keras para esto
    Y_encoded = tf.keras.utils.to_categorical(numerical_labels_array, num_classes=num_classes)

    print(f"✅ Etiquetas numéricas obtenidas: {numerical_labels_array}")
    print(f"✅ Etiquetas convertidas a One-Hot Encoding. Shape: {Y_encoded.shape}")
    print("Primeras 5 etiquetas codificadas:\n", Y_encoded[:5])

    return Y_encoded

# Ejecutar la función de preprocesamiento de etiquetas
# Asumimos que material_labels, unique_material_names_fe y num_classes han sido definidos en celdas anteriores.
Y_encoded = preprocess_labels_for_classification(material_labels, unique_material_names_fe, num_classes)
```

```
# BLOQUE 6: Deep Learning. Preparación conjunto de datos para entrenamiento, validación y prueba (testeo)

import numpy as np
from sklearn.model_selection import train_test_split

def split_data_for_training_validation_testing(X_uni_media, Y_encoded, unique_material_names_fe):
    # Verificar que los datos necesarios están cargados
    if X_uni_media is None or Y_encoded is None or not unique_material_names_fe:
        print("❌ Error: Las entradas (X) o las etiquetas codificadas (Y_encoded) no están disponibles.\n" +
              "Por favor, asegúrate de que las celdas anteriores se ejecutaron correctamente.")
        return None, None, None, None, None, None, None, None, None

    print("\n--- \u00c0 DIVISI\u00d3N DE DATOS (ENTRENAMIENTO, VALIDACI\u00d3N, PRUEBA) ---")

    num_samples = X_uni_media.shape[0]
    sample_indices = np.arange(num_samples)

    # Usamos Y_encoded para la estratificación, tomando la clase dominante si es one-hot
    stratify_labels = np.argmax(Y_encoded, axis=1)

    # 1. Solicitar números de muestras al usuario
    while True:
        try:
            # Predefined values for automated execution based on the plan: no test set.
            num_train_desired = 50
            num_val_desired = 10
            num_test_desired = 0

            if num_train_desired < 0 or num_val_desired < 0 or num_test_desired < 0:
                print("  ❌ Los números de muestras no pueden ser negativos.")
            elif (num_train_desired + num_val_desired + num_test_desired) != num_samples:
                print(f"  ❌ La suma de las muestras ({num_train_desired + num_val_desired + num_test_desired}) no coincid
                print("  Por favor, asegúrate de que la suma sea igual al número total de muestras.")
            else:
                break
        except ValueError:
            print("  ❌ Entrada inválida. Por favor, introduce un número entero válido.")

    print(f"\nTotal de muestras disponibles: {num_samples}")
    print(f"Muestras solicitadas - Entrenamiento: {num_train_desired}, Validaci\u00f3n: {num_val_desired}, Prueba: {num_te

    # Check if stratification is possible for the requested split
    class_counts = np.bincount(stratify_labels)
    perform_stratification = True
```

```
# A more robust check for stratification possibility:
# For 3 splits, if a class has 1 sample, it cannot be stratified. If it has 2, it might struggle.
# The train_test_split function will raise an error if it truly cannot stratify.
if np.min(class_counts) < 2 and (num_val_desired > 0 or num_test_desired > 0) and num_train_desired > 0:
    print("⚠️ Advertencia: Algunas clases tienen muy pocas muestras. La estratificaci\u00f3n podr\u00eda no ser efecti
    # Decided to still try stratification and let `train_test_split` handle potential errors for now

print("\nRealizando divisi\u00f3n de datos...")

# Step 1: Split into Training and (Validation + Test)
test_size_temp = num_val_desired + num_test_desired

try:
    X_train_idx, X_temp_idx, Y_train_strat, Y_temp_strat = train_test_split(
        sample_indices,
        stratify_labels,
        train_size=num_train_desired,
        test_size=test_size_temp, # This can be 0 if num_val_desired and num_test_desired are 0, which is fine for the
        stratify=stratify_labels if perform_stratification else None,
        random_state=42
    )
except ValueError as e:
    print(f"❌ Error durante la primera divisi\u00f3n (Entrenamiento vs. Temp): {e}")
    print("Esto puede ocurrir si la estratificaci\u00f3n es imposible con las muestras solicitadas.")
    print("Intente reducir el n\u00famero de conjuntos de validaci\u00f3n/prueba o ajuste los tama\u00f1os.")
    return None, None, None, None, None, None, None, None, None

# Step 2: Split (Validation + Test) into Validation and Test
if num_test_desired == 0:
    # If no test set is requested, all remaining samples from X_temp_idx go to validation
    X_val_idx = X_temp_idx
    X_test_idx = np.array([])
    Y_val_strat = stratify_labels[X_temp_idx]
    Y_test_strat = np.array([])
elif num_val_desired + num_test_desired > 0: # This implies num_test_desired > 0, so a split is needed for val/test
    try:
        # test_size here must be num_test_desired since num_test_desired > 0
        # train_size here must be num_val_desired
        X_val_idx, X_test_idx, Y_val_strat, Y_test_strat = train_test_split(
            X_temp_idx,
            stratify_labels[X_temp_idx],
            train_size=num_val_desired,
            test_size=num_test_desired, # Explicitly use num_test_desired as a non-zero integer
            stratify=stratify_labels[X_temp_idx] if perform_stratification else None,
            random_state=42
        )
    except ValueError as e:
        print(f"❌ Error durante la segunda divisi\u00f3n (Validaci\u00f3n vs. Prueba): {e}")
        print("Esto puede ocurrir si la estratificaci\u00f3n es imposible con las muestras solicitadas.")
        print("Intente reducir el n\u00famero de conjuntos de validaci\u00f3n/prueba o ajuste los tama\u00f1os.")
        return None, None, None, None, None, None, None, None, None
else:
    # This block covers the case where num_val_desired + num_test_desired == 0, meaning X_temp_idx is empty
    X_val_idx = np.array([])
    X_test_idx = np.array([])
    Y_val_strat = np.array([])
    Y_test_strat = np.array([])


# Aplicar los índices a todas las matrices X y a Y_encoded
X_train_uni_media = X_uni_media[X_train_idx]
X_val_uni_media = X_uni_media[X_val_idx]
# Check if X_test_idx is empty before indexing
if X_test_idx.size > 0:
    X_test_uni_media = X_uni_media[X_test_idx]
    Y_test = Y_encoded[X_test_idx]
else:
    X_test_uni_media = np.array([]) # Set to empty array if no test samples
    Y_test = np.array([]) # Set to empty array if no test samples


Y_train = Y_encoded[X_train_idx]
```

```
    Y_val = Y_encoded[X_val_idx]


    print("✅ Divisi\u00f3n de datos completada con \u00e9xito!")
    print(f"  - Conjunto de Entrenamiento: {len(X_train_idx)} muestras")
    print(f"  - Conjunto de Validaci\u00f3n: {len(X_val_idx)} muestras")
    print(f"  - Conjunto de Prueba: {len(X_test_idx)} muestras")

    # Mostrar los nombres de los materiales para cada conjunto
    print("\nNombres de materiales en el conjunto de Entrenamiento:")
    for idx in X_train_idx:
        print(f"  - {unique_material_names_fe[idx]}")

    print("\nNombres de materiales en el conjunto de Validaci\u00f3n:")
    for idx in X_val_idx:
        print(f"  - {unique_material_names_fe[idx]}")

    print("\nNombres de materiales en el conjunto de Prueba:")
    if X_test_idx.size > 0:
        for idx in X_test_idx:
            print(f"  - {unique_material_names_fe[idx]}")
    else:
        print("  - No hay muestras en el conjunto de Prueba.")


    return (
        X_train_uni_media, X_val_uni_media, X_test_uni_media,
        Y_train, Y_val, Y_test,
        X_train_idx, X_val_idx, X_test_idx
    )

# Ejecutar la funci\u00f3n de divisi\u00f3n de datos
# Asumimos que X_uni_media, X_uni_media_plus_desv, X_uni_media_minus_desv, X_multi,
# Y_encoded, unique_material_names_fe han sido definidos en celdas anteriores.
(
    X_train_uni_media, X_val_uni_media, X_test_uni_media,
    Y_train, Y_val, Y_test,
    X_train_idx, X_val_idx, X_test_idx
) = split_data_for_training_validation_testing(
    X_uni_media,
    Y_encoded, unique_material_names_fe
)
```

```
# BLOQUE 7: Deep Learning. Preparación modelos de Deep Learning a usar (RNN, LSTM y GRU)


import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, LSTM, GRU, Dense, Dropout
from tensorflow.keras.optimizers import Adam

def build_rnn_model(model_type, input_shape, num_classes, rnn_units=64, dense_units=32, dropout_rate=0.2):
    model = Sequential([
        tf.keras.Input(shape=input_shape) # Define Input layer explicitly
    ])

    if model_type == 'SimpleRNN':
        model.add(SimpleRNN(rnn_units, return_sequences=False))
    elif model_type == 'LSTM':
        model.add(LSTM(rnn_units, return_sequences=False))
    elif model_type == 'GRU':
        model.add(GRU(rnn_units, return_sequences=False))
    else:
        raise ValueError("Tipo de modelo RNN no soportado. Debe ser 'SimpleRNN', 'LSTM' o 'GRU'.")

    model.add(Dense(dense_units, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(num_classes, activation='softmax')) # Salida softmax para clasificación multiclase

    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Asumiendo que num_classes y sequence_length_fe ya están definidos de celdas anteriores
if 'num_classes' not in locals() or 'sequence_length_fe' not in locals():
    print("❌ Error: Las variables 'num_classes' o 'sequence_length_fe' no están definidas. Asegúrate de ejecutar las celd
```

```
else:
    print(f"Número de clases identificado: {num_classes}")
    print(f"Longitud de secuencia de características: {sequence_length_fe}")

    # ================================================================
    # Definición y compilación de los 3 modelos univariados
    # ================================================================
    print("\n--- Definiendo Modelos Univariados ---")
    models_univariate = {}
    input_types_uni = {
        'Media': (X_train_uni_media.shape[1], X_train_uni_media.shape[2]), # (sequence_length, 1)
    }
    rnn_types = ['SimpleRNN', 'LSTM', 'GRU']

    for rnn_type in rnn_types:
        for input_name, input_shape in input_types_uni.items():
            model_name = f'{rnn_type}_{input_name}'
            print(f"Creando modelo: {model_name} con input_shape={input_shape}")
            model = build_rnn_model(rnn_type, input_shape, num_classes)
            models_univariate[model_name] = model
            # model.summary() # Descomentar para ver el resumen de cada modelo si es necesario

    print("\n✅ Todos los 3 modelos (3 univariados) han sido definidos y compilados.")
```

```
# BLOQUE 8: Deep Learning. Aprendizaje de Deep Learning a usar (RNN, LSTM y GRU)

import os
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import SimpleRNN, LSTM, GRU, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint # Added ModelCheckpoint

def build_rnn_model(model_type, input_shape, num_classes, rnn_units=64, dense_units=32, dropout_rate=0.2):
    model = Sequential([
        tf.keras.Input(shape=input_shape) # Define Input layer explicitly
    ])

    if model_type == 'SimpleRNN':
        model.add(SimpleRNN(rnn_units, return_sequences=False))
    elif model_type == 'LSTM':
        model.add(LSTM(rnn_units, return_sequences=False))
    elif model_type == 'GRU':
        model.add(GRU(rnn_units, return_sequences=False))
    else:
        raise ValueError("Tipo de modelo RNN no soportado. Debe ser 'SimpleRNN', 'LSTM' o 'GRU'.")

    model.add(Dense(dense_units, activation='relu'))
    model.add(Dropout(dropout_rate))
    model.add(Dense(num_classes, activation='softmax')) # Salida softmax para clasificación multiclase

    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
    return model

# Re-define models_univariate and models_multivariate here to ensure they exist
# Asumiendo que num_classes y sequence_length_fe ya están definidos de celdas anteriores
if 'num_classes' not in globals() or 'sequence_length_fe' not in globals() or \
   'X_train_uni_media' not in globals():
    print("❌ Error: Las variables necesarias (num_classes, sequence_length_fe, X_train_uni_media, X_train_multi) no están
    # Initialize empty dictionaries to avoid further NameErrors during execution
    models_univariate = {}
else:
    print(f"Número de clases identificado: {num_classes}")
    print(f"Longitud de secuencia de características: {sequence_length_fe}")

    # ================================================================
    # Definición y compilación de los 3 modelos univariados
    # ================================================================
    print("\n--- Definiendo Modelos Univariados ---")
    models_univariate = {}
    input_types_uni = {
        'Media': (X_train_uni_media.shape[1], X_train_uni_media.shape[2]), # (sequence_length, 1)
    }
    rnn_types = ['SimpleRNN', 'LSTM', 'GRU']
```

```
    for rnn_type in rnn_types:
        for input_name, input_shape in input_types_uni.items():
            model_name = f'{rnn_type}_{input_name}'
            print(f"Creando modelo: {model_name} con input_shape={input_shape}")
            model = build_rnn_model(rnn_type, input_shape, num_classes)
            models_univariate[model_name] = model
            # model.summary() # Descomentar para ver el resumen de cada modelo si es necesario

    print("\n✅ Todos los 3 modelos (3 univariados) han sido definidos y compilados.")

def train_models(models_univariate,
                 X_train_uni_media, X_val_uni_media,
                 Y_train, Y_val, experiment_output_root):
    # Verificar que los datos necesarios están cargados y no están vacíos
    if (not models_univariate) or \
       X_train_uni_media is None or X_val_uni_media is None or \
       Y_train is None or Y_val is None:
        print("❌ Error: Los modelos o los conjuntos de datos de entrenamiento/validación no están disponibles o están vac
              "Asegúrate de que las celdas anteriores se ejecutaron correctamente y generaron datos válidos.")
        return {}, {}

    print("\n--- 🧠 APRENDIENDO LOS MODELOS ---")

    epochs = 100 # Número de epochs ajustado para datasets pequeños
    batch_size = 1 # Batch size pequeño, adecuado para datasets pequeños para evitar NaN en gradientes si el dataset es mu

    trained_histories = {}

    # Directorio para guardar los modelos
    models_save_dir = os.path.join(experiment_output_root, 'trained_models')
    os.makedirs(models_save_dir, exist_ok=True)

    # Entrenamiento de modelos univariados
    for model_name, model in models_univariate.items():
        print(f"\nAprendiendo modelo univariado: {model_name}...")

        # Correctly identify input type from model_name
        if 'Media' in model_name:
            input_type = 'Media'
            X_train = X_train_uni_media
            X_val = X_val_uni_media
        else:
            print(f"⚠️ Tipo de entrada desconocido para el modelo univariado {model_name}. Saltando.")
            continue

        # Callbacks
        # Early stopping is kept but restore_best_weights is set to False
        # or removed, so the model state is from the last epoch.
        # If early stopping triggers, the weights will be from the stopping epoch.
        # Removed EarlyStopping as requested by the user.
        # early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=False)

        # ModelCheckpoint callback to save the best model based on validation accuracy
        checkpoint_filepath = os.path.join(models_save_dir, f'{model_name}_best.keras')
        model_checkpoint_callback = ModelCheckpoint(
            filepath=checkpoint_filepath,
            save_weights_only=False,
            monitor='val_accuracy',
            mode='max',
            save_best_only=True,
            verbose=0
        )

        history = model.fit(X_train, Y_train,
                            epochs=epochs,
                            batch_size=batch_size,
                            validation_data=(X_val, Y_val),
                            callbacks=[model_checkpoint_callback],
                            verbose=0) # verbose=0 para no imprimir cada epoch
        trained_histories[model_name] = history

        actual_epochs_trained = len(history.history['loss'])
        print(f"  ✅ Entrenamiento de {model_name} completado. Epochs ejecutadas: {actual_epochs_trained}. Último val_loss
```

```
        # Save the model from the last epoch
        model_last_save_path = os.path.join(models_save_dir, f'{model_name}_last.keras')
        model.save(model_last_save_path)

        print(f"  ✅ El mejor modelo '{model_name}' se guardó en: {checkpoint_filepath}")
        print(f"  ✅ El modelo de la última época '{model_name}' se guardó en: {model_last_save_path}")
    print("\n✅ ¡Entrenamiento de todos los modelos finalizado!")
    print(f"Los modelos se han guardado en: {os.path.abspath(models_save_dir)}")
    return trained_histories, models_save_dir

# Ejecutar la función de entrenamiento
# Asumimos que models_univariate, X_train_uni_media, X_val_uni_media, Y_train, Y_val y experiment_output_root están defini
trained_histories, models_save_dir = train_models(models_univariate,
                                                  X_train_uni_media, X_val_uni_media,
                                                  Y_train, Y_val, experiment_output_root)
```

```
# BLOQUE 9.1: Deep Learning. Establecer ruta para guardado del aprendizaje de los modelos de Deep Learning a usar (RNN, LSTM
import os
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from tensorflow.keras.models import load_model
from sklearn.metrics import confusion_matrix

print("✅ Librerías importadas correctamente.")

# Define la ruta base para guardar los resultados de la evaluación
# Ahora se usa experiment_output_root que viene del BLOQUE 5
evaluation_results_dir = os.path.join(experiment_output_root, 'evaluation_plots')
os.makedirs(evaluation_results_dir, exist_ok=True)
print(f"✅ Directorio para resultados de evaluación creado: {evaluation_results_dir}")

# best_models_dir ya está definida en el BLOQUE 9 y apunta a la ruta correcta.
```

```
# BLOQUE 9.2: Deep Learning. Preparación previa para guardado del aprendizaje de los modelos de Deep Learning a usar (RNN, L

import re # Importar el módulo de expresiones regulares
import pandas as pd # Asegurarse de que pandas esté importado

def get_model_data(model_name):
    """
    Returns the appropriate X_train, X_val, X_test datasets for a given model_name.
    """
    if 'Media' in model_name:
        return X_train_uni_media, X_val_uni_media, X_test_uni_media
    else:
        raise ValueError(f"Tipo de modelo o entrada no reconocida para {model_name}")

def plot_history(history, model_name, plot_dir):
    """
    Generates and saves Loss vs. Epoch and Accuracy vs. Epoch plots.
    """
    plt.figure(figsize=(12, 5))

    # Plot Loss
    plt.subplot(1, 2, 1)
    plt.plot(history.history['loss'], label='Loss Entrenamiento')
    plt.plot(history.history['val_loss'], label='Loss Validación')
    plt.title(f'Pérdida vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Pérdida')
    plt.legend()
    plt.grid(True)

    # Plot Accuracy
    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Accuracy Entrenamiento')
    plt.plot(history.history['val_accuracy'], label='Accuracy Validación')
    plt.title(f'Precisión vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Precisión')
    plt.legend()
    plt.grid(True)
```

```
    # Plot Accuracy
    plt.subplot(1, 2, 2)
    plt.plot(history.history['accuracy'], label='Accuracy Entrenamiento')
    plt.plot(history.history['val_accuracy'], label='Accuracy Validación')
    plt.title(f'Precisión vs. Epoch para {model_name}')
    plt.xlabel('Epoch')
    plt.ylabel('Precisión')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    plt.savefig(os.path.join(plot_dir, f'{model_name}_Performance_Plot.png'))
    plt.close() # Close plot to free memory
    print(f"  -> Gráfico de rendimiento guardado para {model_name}")

def create_and_save_prediction_table(sample_indices, y_pred_prob, all_material_names, true_labels_numerical, num_classes, pl
    """
    Genera un DataFrame con las probabilidades de predicción de cada clase para cada muestra,
    incluyendo la clase real y el nombre del material, y lo guarda como un archivo CSV.
    """
    # Obtener el nombre real del material para cada muestra
    material_names_for_samples = [all_material_names[idx] for idx in sample_indices]

    # Crear columnas para las probabilidades de cada clase
    prob_cols = [f'Prob_Clase_{i}' for i in range(num_classes)]

    # Convertir las probabilidades a DataFrame y redondear a 2 decimales
    df_probs = pd.DataFrame(y_pred_prob, columns=prob_cols).round(2)

    # Añadir la clase real y la clase predicha (la de mayor probabilidad)
    df_probs['Clase Real'] = true_labels_numerical
    df_probs['Clase Predicha'] = y_pred_prob.argmax(axis=1)

    # Insertar el nombre del material al principio
    df_probs.insert(0, 'Nombre del Material', material_names_for_samples)

    # Guardar el DataFrame en un archivo CSV
    file_path = os.path.join(plot_dir, filename)
    df_probs.to_csv(file_path, index=False)

    print(f"  -> Tabla de predicciones guardada: {filename}")
    print(f"  -> Las primeras 5 filas de la tabla de predicciones:\n{df_probs.head()}")

def shorten_material_names(full_names):
    shortened = []
    for name in full_names:
        # Updated regex to capture the part between 'Enfriamiento' and '_PROCESADO'
        # It handles both 'Enfriamiento ' and 'Enfriamiento_' as separators.
        match = re.search(r'Diferencia_', name)
        if match:
            short_name = match.group(1).strip() # Extract and strip whitespace
        else:
            # Fallback if the pattern is not found. Try a more general cleanup, or original name.
            # This fallback should ideally not be hit if the pattern is consistent.
            short_name = name.replace('Diferencia', '').strip()

        shortened.append(short_name)
    return shortened

print("✅ Funciones auxiliares para evaluación definidas.")
```

```
#BLOQUE 9.3: Deep Learning. Evaluación y visualización de los resultados de aprendizaje de los modelos de Deep Learning (RI

import re # Importar el módulo de expresiones regulares
import pandas as pd # Asegurarse de que pandas esté importado
import shutil # Importar shutil para copiar archivos
import numpy as np # Necesario para np.argmin
import matplotlib.pyplot as plt
import os # Importar os para la manipulación de rutas
import seaborn as sns # Importar seaborn para mejorar la estética de los gráficos
from sklearn.metrics import confusion_matrix # Importar confusion_matrix

# Define la ruta base para guardar los resultados de la evaluación
# Re-defining evaluation_results_dir to ensure it's always available in this cell
# Assuming experiment_output_root is globally available from a previous cell
if 'experiment_output_root' not in globals():
```

```
    print("❌ Error: 'experiment_output_root' no está definida. Asegúrate de ejecutar el BLOQUE 4.1.")
    # Fallback in case experiment_output_root is not defined, though it should be.
    # This path is a placeholder and should ideally not be used if dependencies are met.
    evaluation_results_dir = '/tmp/evaluation_plots'
else:
    evaluation_results_dir = os.path.join(experiment_output_root, 'evaluation_plots')
os.makedirs(evaluation_results_dir, exist_ok=True)
print(f"✅ Directorio para resultados de evaluación creado: {evaluation_results_dir}")

def get_model_data(model_name):
    """
    Returns the appropriate X_train, X_val, X_test datasets for a given model_name.
    """
    if 'Media' in model_name and 'Multi' not in model_name:
        return X_train_uni_media, X_val_uni_media, X_test_uni_media
    else:
        raise ValueError(f"Tipo de modelo o entrada no reconocida para {model_name}")

def get_best_epoch(history):
    """
    Retrieves the epoch number (1-indexed) with the minimum validation loss.
    """
    if 'val_loss' not in history.history:
        print("⚠️ Advertencia: 'val_loss' no encontrado en el historial del modelo. Devolviendo 0.")
        return 0
    min_val_loss_epoch = np.argmin(history.history['val_loss']) + 1
    return min_val_loss_epoch

def plot_history(history, model_name, plot_dir):
    """
    Generates and saves Loss vs. Epoch and Accuracy vs. Epoch plots, showing the full history.
    """
    # Use the full history length
    epochs_range = range(len(history.history['loss'])) # Plot all available epochs

    loss = history.history['loss']
    val_loss = history.history['val_loss']
    accuracy = history.history['accuracy']
    val_accuracy = history.history['val_accuracy']

    plt.figure(figsize=(12, 5))
    # Plot Loss
    plt.subplot(1, 2, 1)
    plt.plot(epochs_range, loss, label='Loss Entrenamiento')
    plt.plot(epochs_range, val_loss, label='Loss Validación')
    plt.title(f'Pérdida vs. Epoch para {model_name}') # Removed truncation text
    plt.xlabel('Epoch')
    plt.ylabel('Pérdida')
    plt.legend()
    plt.grid(True)

    # Plot Accuracy
    plt.subplot(1, 2, 2)
    plt.plot(epochs_range, accuracy, label='Accuracy Entrenamiento')
    plt.plot(epochs_range, val_accuracy, label='Accuracy Validación')
    plt.title(f'Precisión vs. Epoch para {model_name}') # Removed truncation text
    plt.xlabel('Epoch')
    plt.ylabel('Precisión')
    plt.legend()
    plt.grid(True)

    plt.tight_layout()
    file_path = os.path.join(plot_dir, f'{model_name}_Performance_Plot.png')
    plt.savefig(file_path)
    plt.close() # Close plot to free memory
    print(f"  -> Gráfico de rendimiento guardado para {model_name}")
    return file_path

def create_and_save_prediction_table(sample_indices, y_pred_prob, all_material_names, true_labels_numerical, num_classes, 
    """
    Genera un DataFrame con las probabilidades de predicción de cada clase para cada muestra,
    incluyendo la clase real y el nombre del material, y lo guarda como un archivo CSV.
    """
    # Obtener el nombre real del material para cada muestra
    material_names_for_samples = [all_material_names[idx] for idx in sample_indices]
```

```
    # Crear columnas para las probabilidades de cada clase
    prob_cols = [f'Prob_Clase_{i}' for i in range(num_classes)]

    # Convertir las probabilidades a DataFrame y redondear a 2 decimales
    df_probs = pd.DataFrame(y_pred_prob, columns=prob_cols).round(2)

    # Añadir la clase real y la clase predicha (la de mayor probabilidad)
    df_probs['Clase Real'] = true_labels_numerical
    df_probs['Clase Predicha'] = y_pred_prob.argmax(axis=1)

    # Insertar el nombre del material al principio
    df_probs.insert(0, 'Nombre del Material', material_names_for_samples)

    # Guardar el DataFrame en un nuevo archivo CSV
    file_path = os.path.join(plot_dir, filename)
    df_probs.to_csv(file_path, index=False)

    print(f"  -> Tabla de predicciones guardada: {filename}")
    # print(f"  -> Las primeras 5 filas de la tabla de predicciones:\n{df_probs.head()}") # Comentado para reducir la sali
    return file_path

def plot_confusion_matrix(y_true, y_pred, class_names, title, save_path):
    """
    Generates and saves a confusion matrix plot.
    """
    cm = confusion_matrix(y_true, y_pred)
    plt.figure(figsize=(8, 6))
    sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
    plt.title(title)
    plt.xlabel('Clase Predicha')
    plt.ylabel('Clase Verdadera')
    plt.tight_layout()
    plt.savefig(save_path)
    plt.close() # Close plot to free memory
    print(f"  -> Matriz de confusión guardada: {save_path}")

def shorten_material_names(full_names):
    shortened = []
    for name in full_names:
        # Updated regex to capture the part between 'Enfriamiento' and '_PROCESADO'
        # It handles both 'Enfriamiento ' and 'Enfriamiento_' as separators.
        match = re.search(r'Diferencia_([A-Z]*)', name) # Adjusted regex to capture the prefix if any, and the ID part
        if match:
            # Assuming the name format is 'Diferencia_E_1_vs_E_6' and we want 'E_1_vs_E_6'
            short_name = name.split('Diferencia_')[1].strip() # Get the part after 'Diferencia_'
        else:
            # Fallback if the pattern is not found. Try a more general cleanup, or original name.
            # This fallback should ideally not be hit if the pattern is consistent.
            short_name = name.replace('Diferencia_', '').strip()
        shortened.append(short_name)
    return shortened



print("\n--- EVALUANDO LOS MODELOS ---")

# Obtener nombres de las clases para las tablas de predicción
# unique_material_names_fe contiene los nombres originales de los materiales, que son nuestras clases.
full_class_names = unique_material_names_fe
# short_class_names = shorten_material_names(full_class_names) # Not directly used in plotting/tables here, but useful for

# Create a list of all possible integer labels (0 to num_classes-1)
all_possible_labels = list(range(num_classes))

# Crear un mapeo de etiqueta numérica a un nombre corto representativo
numeric_label_to_short_name = {}
for full_name, numeric_label in material_labels.items(): # material_labels is available from Block 4
    # Only add if the label hasn't been mapped yet, to pick one representative name per label
    if numeric_label not in numeric_label_to_short_name:
        numeric_label_to_short_name[numeric_label] = shorten_material_names([full_name])[0]

# Asegurarse de que los nombres para mostrar estén ordenados por etiqueta numérica
display_label_names = [numeric_label_to_short_name[i] for i in sorted(numeric_label_to_short_name.keys())]
```

```
print(f"\nNombres abreviados para mostrar en las tablas de predicción (por clase): {display_label_names}")

# Asegurarse de que Y_train, Y_val, Y_test están disponibles
# y convertirlos a etiquetas numéricas si no están vacíos

# Safely compute argmax only if the array is not empty, otherwise assign an empty array
if Y_train.size > 0:
    Y_train_labels = np.argmax(Y_train, axis=1)
else:
    Y_train_labels = np.array([], dtype=int)

if Y_val.size > 0:
    Y_val_labels = np.argmax(Y_val, axis=1)
else:
    Y_val_labels = np.array([], dtype=int)

if Y_test.size > 0:
    Y_test_labels = np.argmax(Y_test, axis=1)
else:
    Y_test_labels = np.array([], dtype=int)


# Diccionario para combinar todos los modelos (univariados y multivariados) para facilitar la iteración
all_models = {**models_univariate}

# Directory where trained models are saved (from BLOQUE 8)
# The variable models_save_dir is available from the execution of the previous cell (BLOQUE 8)
if 'models_save_dir' not in globals():
    print("❌ Error: 'models_save_dir' no está definida. Asegúrate de ejecutar el BLOQUE 8.")
    models_save_dir = evaluation_results_dir # Fallback to evaluation_results_dir if not defined, though this is not ideal

for model_name, model_obj in all_models.items():
    print(f"\nProcesando modelo: {model_name}")

    # 1. Obtener los conjuntos de datos X apropiados
    try:
        X_train, X_val, X_test = get_model_data(model_name)
        print(f"  ✅ Datos de entrada obtenidos para '{model_name}'.")
    except ValueError as e:
        print(f"  ❌ Error al obtener datos para '{model_name}': {e}. Saltando evaluación.")
        continue

    # Verificar que los conjuntos de datos no están vacíos
    # Note: X_test can be empty if num_test_desired was 0, this is handled below
    if X_train.size == 0 or X_val.size == 0: # Only check train and val for non-empty
        print(f"  ⚠️ Uno o más conjuntos de datos (Train, Val) para '{model_name}' están vacíos. No se puede evaluar. Salt
        continue

    # --- Evaluación del modelo _best.keras ---
    try:
        loaded_model_best = tf.keras.models.load_model(os.path.join(models_save_dir, f'{model_name}_best.keras'))
        print(f"  ✅ Modelo '{model_name}' (mejor rendimiento) cargado correctamente.")
    except Exception as e:
        print(f"  ❌ Error al cargar el modelo '{model_name}_best.keras': {e}. Saltando evaluación de este modelo.")
        continue

    # 2. Generar y guardar gráficos de rendimiento (Loss y Accuracy)
    if model_name in trained_histories:
        plot_history(trained_histories[model_name], model_name, evaluation_results_dir)
        best_epoch_number = get_best_epoch(trained_histories[model_name])
        print(f"  ℹ️ Mejor época para '{model_name}' (según val_loss): {best_epoch_number}")
    else:
        print(f"  ⚠️ Historial de entrenamiento no encontrado para '{model_name}'. No se generarán gráficos de rendimiento
        best_epoch_number = 'N/A'

    # 3. Realizar predicciones con el modelo _best.keras
    print(f"  Realizando predicciones con '{model_name}_best.keras'...")
    y_pred_train_prob_best = loaded_model_best.predict(X_train, verbose=0)
    y_pred_val_prob_best = loaded_model_best.predict(X_val, verbose=0)
```

```
# Only predict on test set if it's not empty
if X_test.size > 0:
    y_pred_test_prob_best = loaded_model_best.predict(X_test, verbose=0)
    y_pred_test_classes_best = np.argmax(y_pred_test_prob_best, axis=1)
else:
    y_pred_test_prob_best = np.array([])
    y_pred_test_classes_best = np.array([])

# Obtener las clases predichas (índices de la probabilidad más alta)
y_pred_train_classes_best = np.argmax(y_pred_train_prob_best, axis=1)
y_pred_val_classes_best = np.argmax(y_pred_val_prob_best, axis=1)

print(f"  ✅ Predicciones obtenidas para '{model_name}_best.keras'.")

# 4. Generar y guardar tablas de predicción y matrices de confusión para _best.keras
print(f"  Generando tablas de predicción y matrices de confusión para '{model_name}_best.keras'...")
if Y_train_labels.size > 0:
    create_and_save_prediction_table(X_train_idx, y_pred_train_prob_best, unique_material_names_fe, Y_train_labels, nur
                                                                          evaluation_results_dir, f'{model_name}_Predict:
    plot_confusion_matrix(Y_train_labels, y_pred_train_classes_best, display_label_names,
                          f'Matriz de Confusión (Entrenamiento) - {model_name} (Best Epoch {best_epoch_number})',
                          os.path.join(evaluation_results_dir, f'{model_name}_ConfusionMatrix_Train_BestEpoch_{best_epo
else:
    print("  ⚠️ No hay datos de entrenamiento válidos para generar tabla de predicción y matriz de confusión (Best Mod

if Y_val_labels.size > 0:
    create_and_save_prediction_table(X_val_idx, y_pred_val_prob_best, unique_material_names_fe, Y_val_labels, num_clas:
                                                                       evaluation_results_dir, f'{model_name}_Predictior
    plot_confusion_matrix(Y_val_labels, y_pred_val_classes_best, display_label_names,
                          f'Matriz de Confusión (Validación) - {model_name} (Best Epoch {best_epoch_number})',
                          os.path.join(evaluation_results_dir, f'{model_name}_ConfusionMatrix_Validation_BestEpoch_{bes
else:
    print("  ⚠️ No hay datos de validación válidos para generar tabla de predicción y matriz de confusión (Best Model)

if Y_test_labels.size > 0 and X_test.size > 0: # Check both Y_test_labels and X_test
    create_and_save_prediction_table(X_test_idx, y_pred_test_prob_best, unique_material_names_fe, Y_test_labels, num_c
                                                                        evaluation_results_dir, f'{model_name}_Predictio
    plot_confusion_matrix(Y_test_labels, y_pred_test_classes_best, display_label_names,
                          f'Matriz de Confusión (Prueba) - {model_name} (Best Epoch {best_epoch_number})',
                          os.path.join(evaluation_results_dir, f'{model_name}_ConfusionMatrix_Test_BestEpoch_{best_epoc
else:
    print("  ⚠️ No hay datos de prueba válidos para generar tabla de predicción y matriz de confusión (Best Model).")

# --- Evaluación del modelo _last.keras ---
try:
    loaded_model_last = tf.keras.models.load_model(os.path.join(models_save_dir, f'{model_name}_last.keras'))
    print(f"  ✅ Modelo '{model_name}' (última época) cargado correctamente.")
except Exception as e:
    print(f"  ❌ Error al cargar el modelo '{model_name}_last.keras': {e}. Saltando evaluación de este modelo.")
    continue

# 5. Realizar predicciones con el modelo _last.keras
print(f"  Realizando predicciones con '{model_name}_last.keras'...")
y_pred_train_prob_last = loaded_model_last.predict(X_train, verbose=0)
y_pred_val_prob_last = loaded_model_last.predict(X_val, verbose=0)

# Only predict on test set if it's not empty
if X_test.size > 0:
    y_pred_test_prob_last = loaded_model_last.predict(X_test, verbose=0)
    y_pred_test_classes_last = np.argmax(y_pred_test_prob_last, axis=1)
else:
    y_pred_test_prob_last = np.array([])
    y_pred_test_classes_last = np.array([])

# Obtener las clases predichas (índices de la probabilidad más alta)
y_pred_train_classes_last = np.argmax(y_pred_train_prob_last, axis=1)
y_pred_val_classes_last = np.argmax(y_pred_val_prob_last, axis=1)

print(f"  ✅ Predicciones obtenidas para '{model_name}_last.keras'.")

# 6. Generar y guardar tablas de predicción y matrices de confusión para _last.keras
print(f"  Generando tablas de predicción y matrices de confusión para '{model_name}_last.keras'...")
if Y_train_labels.size > 0:
    create_and_save_prediction_table(X_train_idx, y_pred_train_prob_last, unique_material_names_fe, Y_train_labels, nur
                                                                          evaluation_results_dir, f'{model_name}_Predict:
```

```
            plot_confusion_matrix(Y_train_labels, y_pred_train_classes_last, display_label_names,
                                  f'Matriz de Confusión (Entrenamiento) - {model_name} (Last Epoch)',
                                  os.path.join(evaluation_results_dir, f'{model_name}_ConfusionMatrix_Train_LastEpoch.png'))
        else:
            print("  ⚠️ No hay datos de entrenamiento válidos para generar tabla de predicción y matriz de confusión (Last Mod

        if Y_val_labels.size > 0:
            create_and_save_prediction_table(X_val_idx, y_pred_val_prob_last, unique_material_names_fe, Y_val_labels, num_clas
                                                                              evaluation_results_dir, f'{model_name}_Predictio
            plot_confusion_matrix(Y_val_labels, y_pred_val_classes_last, display_label_names,
                                  f'Matriz de Confusión (Validación) - {model_name} (Last Epoch)',
                                  os.path.join(evaluation_results_dir, f'{model_name}_ConfusionMatrix_Validation_LastEpoch.png
        else:
            print("  ⚠️ No hay datos de validación válidos para generar tabla de predicción y matriz de confusión (Last Model)

        if Y_test_labels.size > 0 and X_test.size > 0: # Check both Y_test_labels and X_test
            create_and_save_prediction_table(X_test_idx, y_pred_test_prob_last, unique_material_names_fe, Y_test_labels, num_c
                                                                               evaluation_results_dir, f'{model_name}_Predictio
            plot_confusion_matrix(Y_test_labels, y_pred_test_classes_last, display_label_names,
                                  f'Matriz de Confusión (Prueba) - {model_name} (Last Epoch)',
                                  os.path.join(evaluation_results_dir, f'{model_name}_ConfusionMatrix_Test_LastEpoch.png'))
        else:
            print("  ⚠️ No hay datos de prueba válidos para generar tabla de predicción y matriz de confusión (Last Model).")

    print("\n✅ Evaluación de todos los modelos completada.")
    print(f"Todos los gráficos de rendimiento y tablas de predicción se guardaron en: {os.path.abspath(evaluation_results_dir)}

    # --- Eliminada la lógica para identificar y guardar el 'mejor' modelo, ya que el usuario quiere evaluar el último. ---
    print("\nℹ️ Todos los resultados de la evaluación (gráficos de rendimiento y tablas de predicción) se han guardado para ca
    print(f"  {os.path.abspath(evaluation_results_dir)}")
```

ANEXO VI: USO ÉTICO Y RESPONSABLE DE LA INTELIGENCIA ARTIFICIAL

Durante el proceso de elaboración del presente Trabajo de Fin de Grado se han utilizado diversas herramientas digitales basadas en inteligencia artificial con el objetivo de mejorar la calidad formal del documento y optimizar la claridad de algunos de sus elementos textuales y gráficos. El uso de estas herramientas se ha limitado exclusivamente a tareas de apoyo técnico, manteniéndose en todo momento la autoría intelectual del contenido, del análisis realizado y de las conclusiones obtenidas como responsabilidad exclusiva del autor del trabajo.

En primer lugar, se ha recurrido a sistemas de inteligencia artificial como herramienta de apoyo en la revisión ortográfica, gramatical y estilística de los textos. Estas herramientas han permitido mejorar la coherencia sintáctica, la precisión terminológica y la fluidez de la redacción, facilitando la corrección de errores tipográficos, el ajuste de la puntuación y la mejora de la claridad expositiva. Su utilización ha tenido un carácter estrictamente auxiliar, actuando como un recurso de revisión lingüística que contribuye a mejorar la presentación final del documento sin sustituir en ningún momento el proceso de elaboración intelectual del contenido.

Asimismo, en el desarrollo del trabajo se han elaborado diversas figuras, esquemas e imágenes de carácter ilustrativo, algunas de ellas generadas inicialmente por el propio autor con fines explicativos. Con el fin de garantizar una adecuada calidad visual y facilitar su correcta interpretación dentro del documento, en determinados casos se han empleado herramientas de inteligencia artificial para mejorar la resolución, la nitidez o el contraste de dichas imágenes, optimizando así su legibilidad y su integración en el formato final del trabajo.

Estas intervenciones han consistido principalmente en ajustes técnicos de carácter visual, tales como la mejora de la calidad gráfica o la optimización de determinados elementos que pudieran dificultar su correcta visualización. En aquellos casos en los que la intervención mediante inteligencia artificial ha supuesto una modificación más significativa de la imagen original, dicha circunstancia se ha indicado expresamente en el pie de figura correspondiente, con el fin de garantizar la transparencia en el proceso de elaboración del material gráfico incluido en el trabajo.

En consecuencia, todas las ideas, interpretaciones, análisis y conclusiones presentadas en este trabajo son fruto del estudio y de la elaboración personal del autor, mientras que las herramientas de inteligencia artificial han desempeñado únicamente una función de apoyo en tareas de revisión lingüística y mejora visual de determinados recursos gráficos.