ULPGC Universidad de Las Palmas de Gran Canaria | eite ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

ESCUELA DE INGENIERÍA DE TELECOMUNICACIÓN Y ELECTRÓNICA

TRABAJO FIN DE GRADO

CLASIFICACIÓN DE IMÁGENES RADIOGRÁFICAS UTILIZANDO INTELIGENCIA ARTIFICIAL

Titulación: Grado en Ingeniería en Tecnologías de la Telecomunicación

Mención: Sonido e Imagen

Autora: Paula Rodríguez Álvarez

Tutor: Juan Bautista Ruiz Alzola

Fecha: Julio 2024

Agradecimientos

En primer lugar, me gustaría agradecer a mi madre y a Santi por el esfuerzo que han hecho, aguantando los llanos y alegrías que me ha dado esta carrera, apoyándome siempre para que diera todo de mí. Repitiéndome cada vez que aprobaba alguna asignatura “Ay, tú no sabes la alegría que me acabas de dar, este es el mejor regalo”. Con la ayuda de los tuppers de mi madre siempre se estudiaba mejor.

A mi abuela que ha aguantado tenerme fuera siempre preguntándome “¿y cuándo vienes?” o “¿cómo se portan esos canariones?". Como dice ella, "este golpe ya está dado".

A todos mis familiares en general que siempre me han apoyado. No puedo olvidarme de los que no pudieron ver cómo se terminaba este camino, en especial de mi tío, que me decía siempre que iba a tener una sobrina ingeniera, pues ahora sí.

A mis amigos que han aguantado todos los veranos que llegara la última a Tenerife, porque tenía ocho mil asignaturas para extraordinaria, preguntándome siempre, “¿Pero tú cuando vuelves?”. Gracias por darme ánimos siempre para que cogiera fuerzas para estudiar y sacarlo todo.

A la universidad por darme la oportunidad de realizar un Erasmus, y vivir la mejor experiencia de mi vida, que me ayudó a dar un gran paso con las asignaturas de la carrera. Esta experiencia me permitió conocer a gente maravillosa con la que compartí momentos inolvidables.

A todos mis amigos de la carrera, con los que he compartido alegrías por aprobar y lágrimas por suspender. Hemos pasado muchos ratos en las peceras de Teleco, pero también muchos momentos fuera de ella, hablando hasta por los codos y haciendo planes en mi casa para que no todo fuera estudiar. En especial a los de Tenerife, que han sido como una familia para mí en Las Palmas, estando las 24 horas del día juntos, compartiendo todos los momentos de esta experiencia.

Por último, a mi tutor, Juan, por toda la ayuda durante este proyecto y por haberme descubierto un campo nuevo para mí que seguiré estudiando en una nueva etapa que me espera el próximo año.

Resumen

La clasificación precisa de imágenes radiográficas es fundamental en medicina, ya que estas imágenes proporcionan información para el diagnóstico y tratamiento de enfermedades y lesiones. La correcta interpretación de estas imágenes mejora la precisión diagnóstica y optimiza la planificación de tratamientos, lo cual es crucial en entornos clínicos donde el tiempo es esencial.

Este TFG (Trabajo Fin de Grado) se centra en el desarrollo de un sistema para la clasificación de imágenes radiográficas entre normales y anormales utilizando inteligencia artificial (IA), con el objetivo de mejorar la precisión y eficiencia en el diagnóstico médico en atención primaria, donde muchas veces no hay médicos especialistas para dar un diagnóstico preciso, conllevando a tardanzas debido a la necesidad de consultar con médicos especializados o incluso en errores diagnósticos.

Específicamente, se ha implementado un modelo de aprendizaje profundo para analizar y clasificar imágenes radiográficas, empleando técnicas avanzadas de redes neuronales convolucionales (CNN). En particular, se ha utilizado una arquitectura DenseNet, conocida por su capacidad para mejorar la propagación de información y gradientes a través de la red, lo que resulta en un modelo más eficiente y preciso.

Para el desarrollo y entrenamiento del modelo, se utilizó la base de datos MURA (Musculoskeletal Radiographs), una de las colecciones más grandes de imágenes radiográficas de extremidades disponibles públicamente.

Los resultados obtenidos durante el desarrollo y las pruebas demostraron que el sistema es capaz de clasificar imágenes radiográficas con alta precisión. Este TFG demuestra el potencial de la inteligencia artificial para revolucionar el campo de la radiología, ofreciendo una herramienta poderosa que puede integrarse en sistemas médicos existentes para mejorar la calidad y rapidez del diagnóstico.

Abstract

Accurate classification of radiographic images is fundamental in medicine, as these images provide information for the diagnosis and treatment of diseases and injuries. Correct interpretation of these images enhances diagnostic precision and optimizes treatment planning, which is crucial in clinical settings where time is essential.

This Final Degree Project (TFG) focuses on developing a system for classifying radiographic images into normal and abnormal using artificial intelligence (AI), with the aim of improving diagnostic precision and efficiency in primary care, where there are often no specialist doctors available to provide an accurate diagnosis, leading to delays due to the need to consult with specialists or even diagnostic errors.

Specifically, a deep learning model has been implemented to analyze and classify radiographic images, employing advanced techniques of convolutional neural networks (CNN). In particular, a DenseNet architecture was used, known for its ability to improve the propagation of information and gradients through the network, resulting in a more efficient and accurate model.

For the development and training of the model, the MURA (Musculoskeletal Radiographs) database was used, one of the largest publicly available collections of extremity radiographic images.

The results obtained during development and testing demonstrated that the system is capable of classifying radiographic images with high precision. This TFG demonstrates the potential of artificial intelligence to revolutionize the field of radiology, offering a powerful tool that can be integrated into existing medical systems to improve the quality and speed of diagnosis.

Índice

Índice de Figuras 11

Índice de Tablas 13

Índice de Ecuaciones 14

Lista de Acrónimos 15

1. Introducción 17

1.1. Antecedentes 17

1.2. Objetivos 19

1.3. Estructura de la memoria 19

2. Fundamentos teóricos 21

2.1. Radiografía musculoesquelética 21

2.1.1. ¿Qué es una radiografía? 21

2.1.2. ¿Cómo se genera una radiografía? 21

2.1.3. Proceso de formación de la imagen radiológica 22

2.1.4. Importancia de la radiografía en el diagnóstico médico 24

2.2. Sistema musculoesquelético 24

2.2.1. Estructura y función del sistema musculoesquelético 24

2.2.2. Enfermedades y lesiones comunes 26

2.2.3. Técnicas de diagnóstico por imagen 26

2.2.4. Radiografía musculoesquelética 29

2.2.5. Radiografía musculoesquelética en atención primaria 29

2.3. Inteligencia Artificial 30

2.3.1. Conceptos básicos de redes neuronales 30

2.3.2. Clasificación de imágenes con IA 32
3. Tecnologías utilizadas 35
3.1. Software utilizado 35
3.1.1. Pytorch 35
3.1.2. Jupyter Notebook 36
3.1.3. GitHub 37
3.2. Hardware utilizado 38
3.2.1. Ordenador Asus Zenbook UX425EA 38
3.2.2. Ordenador DELL Precision 5820 38
4. Base de datos MURA (Musculoskeletal Radiographs) 39
4.1. Descripción de la base de datos 39
4.2. Estructura de la base de datos 40
5. Metodología 42
5.1. Análisis exploratorio de datos 42
5.1.1. Distribución de las clases 42
5.1.2. Visualización de datos 44
5.1.3. Descripción estadística 47
5.2. Preparación de datos 49
5.2.1. Implementación de la clase MuraDataset 50
5.2.2. Preprocesamiento de imágenes 50
5.2.3. Creación de conjuntos de datos de entrenamiento y prueba 51
5.2.4. Inspección y visualización de imágenes preprocesadas 52
5.3. Selección del modelo 53
5.3.1. Arquitecturas de redes neuronales analizadas 53
5.3.2. Justificación de la selección del modelo 55
5.3.3. Configuración del modelo 56
5.4. Entrenamiento del modelo 57

5.4.1. Configuración de hiperparámetros 57
5.4.2. Configuración del entorno de ejecución 58
5.4.3. Configuración para el entrenamiento 59
5.4.4. Proceso de entrenamiento del modelo 60
5.4.5. Visualización de resultados del modelo 61
6. Resultados 63
6.1. Evaluación del modelo 63
6.1.1. Métricas de evaluación 63
6.2. Resultados obtenidos 64
6.3. Comparación con estudios previos 68
7. Conclusiones 70
8. Líneas futuras 72
9. Presupuesto 73
9.1. Trabajo tarifado por tiempo empleado 73
9.2. Amortización del inmovilizado material 74
9.2.1. Amortización del material hardware 75
9.2.2. Amortización del material software 75
9.3. Costes asociados a la redacción del documento 76
9.4. Derechos de visado del COITT 77
9.5. Gastos de tramitación y envío 77
9.6. Aplicación de impuestos 77
10. Bibliografía 79
11. Anexos 82
11.1. Análisis exploratorio de los datos 82
11.2. Estadística 92
11.3. Preparación de datos 94
11.4. Entrenamiento del modelo 98

11.5. Evaluación del modelo y obtención de resultados..... 104

Índice de Figuras

Figura 1. Esquema de un tubo de rayos X 22
Figura 2. Formación de imágenes 23
Figura 3. Las cinco densidades radiológicas 23
Figura 4. Sistema musculoesquelético 25
Figura 5. Sistemas de rayos X 27
Figura 6. Sistema de tomografía computarizada (TC) 27
Figura 7. Sistema de resonancia magnética (RM) 28
Figura 8. Sistema de ecografía 28
Figura 9. Ejemplo de una red neuronal totalmente conectada 30
Figura 10. Funciones de activación más utilizadas 32
Figura 11. Operación de convolución 33
Figura 12. Logo de Pytorch 36
Figura 13. Interfaz de Jupyter Notebook 37
Figura 14. Interfaz de GitHub 38
Figura 15. Cantidad de estudios normales y anormales 42
Figura 16. Cantidad de estudios normales y anormales por tipo de estudio (Train) 43
Figura 17. Cantidad de estudios normales y anormales por tipo de estudio (Valid) 44
Figura 18. Estudio anormal (izq.) y normal (dcha.) del codo 45
Figura 19. Estudio anormal (izq.) y normal (dcha.) del dedo 45
Figura 20. Estudio anormal (izq.) y normal (dcha.) del antebrazo 46
Figura 21. Estudio anormal (izq.) y normal (dcha.) de la mano 46
Figura 22. Estudio anormal (izq.) y normal (dcha.) del húmero 46
Figura 23. Estudio anormal (izq.) y normal (dcha.) del hombro 47
Figura 24. Estudio anormal (izq.) y normal (dcha.) de la muñeca 47
Figura 25. Visualización de imágenes preprocesadas 53

Figura 26. Ejemplo arquitectura DenseNet 54
Figura 27. Clasificador con mil características de salida 57
Figura 28. Clasificador con dos características de salida 57
Figura 29. Ejemplo de dos imágenes con su predicción de clasificación y correspondiente probabilidad. 62
Figura 30. Matriz de confusión del modelo de la muñeca 66
Figura 31. Curva ROC del modelo de la muñeca 67

Índice de Tablas

Tabla 1. Resumen de la distribución de estudios normales y anormales 40
Tabla 2. Resultados de las métricas de evaluación para el modelo de clasificación binaria de imágenes radiográficas de la muñeca. 66
Tabla 3. Extracción de la tabla de coeficientes de amortización lineal de la Agencia Tributaria (AEAT) 74
Tabla 4. Coste de amortización del hardware 75
Tabla 5. Coste de amortización del software 75
Tabla 6. Cálculo de P derivada de la ecuación de coste por redacción 76
Tabla 7. Coste total del proyecto con los impuestos correspondientes aplicados 78

Índice de Ecuaciones

Ecuación 1. Regla de propagación 31
Ecuación 2. Media muestral 48
Ecuación 3. Varianza muestral 49
Ecuación 4. Desviación típica 49
Ecuación 5. Precisión 63
Ecuación 6. Sensibilidad 64
Ecuación 7. Exactitud 64
Ecuación 8. Honorarios por tiempo empleado 74
Ecuación 9. Coste de amortización 74
Ecuación 10. Coste por redacción del proyecto 76
Ecuación 11. Honorarios por la redacción del proyecto 76
Ecuación 12. Resultado de los honorarios por la redacción del proyecto 76
Ecuación 13. Cálculo de los gastos de visado del COITT 77
Ecuación 14. Resultado del cálculo de los gastos de visado del COITT 77

Lista de Acrónimos

- **AEAT** Agencia Estatal de Administración Tributaria
- **AUC** Area Under the Curve (Área bajo la curva)
- **CNN** Convolutional neural network (Red neuronal convolucional)
- **COITT** Colegio Oficial de Ingenieros Técnicos de Telecomunicaciones
- **CPU** Central Processing Unit (Unidad Central de Procesamiento)
- **DICOM** Digital Imaging and Communication in Medicine
- **DXA** Densitometría Ósea
- **FN** False Negative (Falsos Negativo)
- **FP** False Positive (Falsos Positivo)
- **FPR** False Positive Rate (Tasa de Falsos Positivos)
- **GPU** Graphics Processing Unit (Unidad de Procesamiento de Gráficos)
- **IA** Inteligencia Artificial
- **IGIC** Impuesto General Indirecto Canario
- **MURA** Musculoskeletal Radiographs

- **PACS** Sistema de Comunicación y Archivo de Imágenes
- **ReLU** Rectified Linear Unit (Unidad Lineal Rectificada)
- **RM** Resonancia Magnética
- **ROC** Receiver Operating Characteristic (Característica Operativa del Receptor)
- **TC** Tomografía Computarizada
- **TN** True Negative (Verdadero Negativo)
- **TP** True Positive (Verdadero Positivo)
- **TPR** True Positive Rate (Tasa de Verdaderos Positivos)
- **ULPGC** Universidad de Las Palmas de Gran Canaria

1. Introducción

1.1. Antecedentes

Durante los últimos años hemos asistido a un notable crecimiento de tecnologías innovadoras dirigidas al ámbito médico. Estos avances facilitan una evaluación más precisa de los pacientes, buscando proporcionar diagnósticos con exactitud y, de ser posible, de manera inmediata. También facilitan una mejor diagnosis a la hora de evaluar a los pacientes, dotándola de precisión y, de ser posible, de mayor rapidez. Estas tecnologías pueden ser desarrollos nuevos en sus especialidades y mejoras en sistemas existentes. En particular, este proyecto se enfoca en optimizar y aprovechar un sistema ya conocido y ampliamente utilizado, la radiografía, con la ayuda de modernas técnicas de inteligencia artificial.

Una radiografía es una prueba rápida e indolora que genera imágenes de las estructuras internas del cuerpo, principalmente de los huesos. Para ello, los rayos X atraviesan el cuerpo y se absorben en distintas cantidades según la densidad del material a través del que pasan. Los materiales más densos, como los huesos y los metales, se visualizan en blanco en las radiografías. El aire en los pulmones se muestra en tono negro, mientras que la grasa y los músculos aparecen como sombras en tono gris [1].

El principal problema que se pretende mejorar es la atención médica primaria que se recibe a la hora de descartar o valorar una patología a través de una radiografía. Para esto, el médico de atención primaria puede solicitar una radiografía en el centro de salud que tendrá que valorar con objeto de descartar o diagnosticar una patología o, en su caso, referir el caso a un servicio especializado, por lo general en un hospital. Es esencial que el profesional de atención primaria, que no es un radiólogo especialista, pueda concluir con seguridad un diagnóstico por sí mismo o, cuando ello no sea posible, derivar el caso al servicio especializado. Un exceso de derivaciones a servicios especializados de casos que pueden

tratarse adecuadamente en atención primaria supone una sobrecarga de los servicios especializados y mayor tiempo de espera para casos que realmente los necesitan. Pero no derivar a los servicios especializados casos que lo requieren suponen un riesgo serio para la salud del paciente. Es necesario, por tanto, un equilibrio en el que se resuelvan en atención primaria los casos que no requieran atención especializada, pero sin dejar de derivar los casos que si lo necesitan.

Incluso cuando se consulta a un especialista médico, la interpretación de radiografías es un procedimiento que demanda tiempo y exige una significativa dedicación y atención por parte de los radiólogos. La introducción de la inteligencia artificial en este campo se ha revelado como un elemento de interés para mejorar la eficacia del diagnóstico. Los sistemas de inteligencia artificial pueden realizar análisis automáticos y rápidos de las imágenes radiográficas, resaltando áreas de interés y asistiendo a los radiólogos identificando casos más urgentes o complejos. Esto no solo agiliza el proceso de diagnóstico, sino que también aligera la carga laboral de los profesionales de la salud [2].

Otros de los problemas encontrados son los errores diagnósticos. Estos, son más frecuentes de lo que se desea, entre el 2% al 30% de los informes radiológicos pueden tener errores, y los errores diagnósticos serían los responsables del 45% de los eventos adversos en radiología [3].

La inteligencia artificial puede ser una herramienta de ayuda para disminuir errores y elevar la precisión en el proceso de diagnóstico. Los algoritmos de aprendizaje automático, entrenados con conjuntos de datos, pueden identificar patrones sutiles y cotejarlos con casos previos, evitando interpretaciones erróneas y facilitando una toma de decisiones más precisa. Esta capacidad brinda una segunda opinión confiable y complementa el juicio clínico de los profesionales médicos [2].

Debe entenderse que la inteligencia artificial no sustituye al profesional sanitario, sino que es una nueva herramienta tecnológica que le puede ayudar a hacer su trabajo. Estamos viviendo una revolución en la inteligencia artificial, particularmente por las modernas redes neuronales y el aprendizaje profundo, y es pronto para saber el espacio que ocupará en los próximos años en la atención médica. El éxito de los modernos sistemas de visión artificial apunta a que será posible obtener automáticamente hallazgos radiológicos por medios automáticos. Pero con frecuencia, el estudio de imagen no es suficiente para hacer un

diagnóstico, debiéndose valorar otras cuestiones de la historia clínica del paciente. Además, a diferencia de lo que sucede en otras áreas de la visión artificial, no abundan las bases de datos de imágenes radiológicas (ni de otro tipo de datos médicos) para el entrenamiento y evaluación, entre otros motivos por la necesaria protección de los datos personales de los pacientes. Además, el marco regulatorio para los productos médicos es complejo pues debe asegurar la seguridad del paciente, de modo que las autorizaciones para lo comercialización de nuevos sistemas es costos y larga. Todo ello hace que sea necesario prestar atención desde el ámbito universitario a la aplicación de la inteligencia artificial en medicina en diferentes aplicaciones, a lo que contribuye el presente trabajo de fin de grado.

1.2. Objetivos

El objetivo principal del proyecto es el entrenamiento y evaluación de redes neuronales utilizando una base de datos para la clasificación de imágenes radiográficas:

O1. Identificar bases de datos públicas de imágenes radiográficas para entrenamiento con IA.

O2. Entrenar redes neuronales profundas para clasificación de imágenes.

O3. Evaluar los resultados obtenidos y valorar la viabilidad de los sistemas automáticos de ayuda radiográfica basados en IA para la atención primaria.

1.3. Estructura de la memoria

La memoria del proyecto se estructura en nueve capítulos que se describen brevemente en los siguientes puntos:

- **Capítulo 1:** En este capítulo se revisan los antecedentes que justifican la realización de este proyecto, los objetivos de este y se describe la estructura de la memoria.
- **Capítulo 2:** En este capítulo se introducen los fundamentos teóricos que sustentan el proyecto: la radiografía musculoesquelética, el sistema musculoesquelético y la inteligencia artificial.
- **Capítulo 3:** En este capítulo se detallan las tecnologías tanto de software como de hardware que se utilizaron.
- **Capítulo 4:** En este capítulo se examina la base de datos MURA (Musculoskeletal Radiographs) focalizándose en su estructura y características fundamentales.

- **Capítulo 5:** En este capítulo se expone detalladamente la metodología utilizada para llevar a cabo el proyecto, abarcando desde el análisis exploratorio de datos hasta el entrenamiento.
- **Capítulo 6:** En este capítulo se presentan los resultados obtenidos del estudio, incluyendo la evaluación del modelo desarrollado y su comparación con estudios previos.
- **Capítulo 7:** En este capítulo se presentan las conclusiones derivadas del estudio realizado.
- **Capítulo 8:** En este capítulo se presentan las posibles líneas futuras.
- **Capítulo 9:** En este capítulo se presenta el presupuesto haciendo un análisis del coste económico necesario para abordar el proyecto.

2. Fundamentos teóricos

2.1. Radiografía musculoesquelética

2.1.1. ¿Qué es una radiografía?

Una radiografía es una prueba rápida e indolora que genera imágenes de las estructuras internas del cuerpo, principalmente de los huesos. Para ello, los rayos X atraviesan el cuerpo y se absorben en distintas cantidades según la densidad del material a través del que pasan. Los materiales más densos, como los huesos y los metales, se visualizan en blanco en las radiografías. El aire en los pulmones se representa en tono negro, mientras que la grasa y los músculos aparecen como sombras en tono gris [1].

2.1.2. ¿Cómo se genera una radiografía?

La generación de una radiografía se realiza en un sistema basado en la emisión de rayos X dentro de una ampolla al vacío. Este sistema consta de un cátodo (con un filamento incandescente que actúa como fuente de electrones) y un ánodo (que genera la radiación cuando los electrones impactan en él). La energía necesaria proviene de una fuente de alto voltaje, y el conjunto está protegido por una estructura metálica aislante (generalmente plomo) que incluye un colimador. El colimador controla la anchura del haz de rayos X, permitiendo que la mayor cantidad de radiación sea ortogonal al objeto a radiografiar, minimizando la radiación dispersa.

El haz de rayos X que sale del colimador se propaga en línea recta, comportándose de la siguiente manera:

- Dispersión: Parte de la radiación se dispersa en el entorno, dependiendo del grado de colimación.
- Radiación Directa: Atraviesa el objeto de estudio (por ejemplo, una parte del cuerpo del paciente).
    - Absorción: Una parte es absorbida por el objeto, afectada por los parámetros físicos aplicados (amperaje y voltaje).
    - Reflexión (Efecto Compton): Otra parte es reflejada fuera del objeto. Para minimizar este efecto, se utiliza una rejilla antidifusora entre el cuerpo y la placa radiográfica.
    - Atenuación y Transmisión: La radiación que atraviesa el objeto con la debida atenuación es la que impresiona la placa radiográfica, creando la imagen [4].

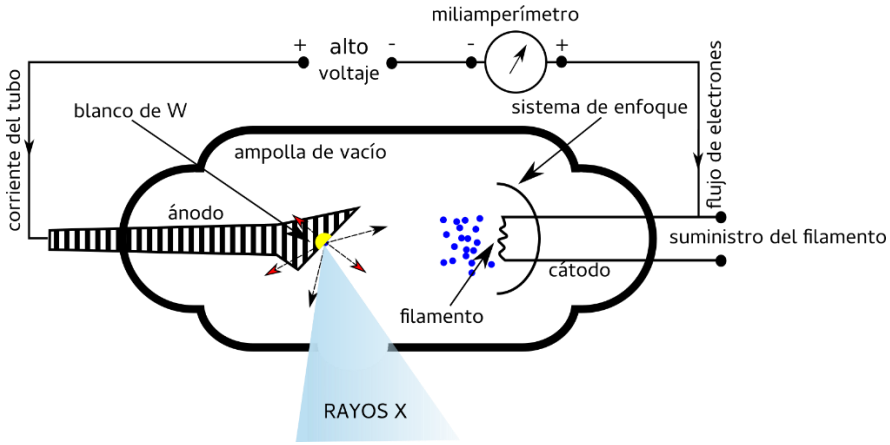


Figura 1. Esquema de un tubo de rayos X

Página web: https://www.famaf.unc.edu.ar/~pperez1/manuales/cdr/tubos-de-rayos-x.html

2.1.3. Proceso de formación de la imagen radiológica

1. Emisión: El tubo de rayos X emite un haz de radiación.
2. Atraviesa el Cuerpo: La radiación atraviesa el cuerpo del paciente, siendo absorbida en distintos grados por los diferentes tejidos.
3. Impresión: La radiación que atraviesa los tejidos impresiona la placa radiográfica.
4. Formación de la Imagen Latente: Los rayos X son absorbidos por los cristales de plata presentes en la emulsión de la película, almacenando energía y formando una imagen latente.
5. Revelado y Fijación: Para obtener la imagen definitiva, la placa se somete a procesos químicos de revelado y fijación. La radiografía final muestra una imagen en escala de

grises, donde las áreas negras indican radiotransparencia y las áreas blancas indican radiodensidad.

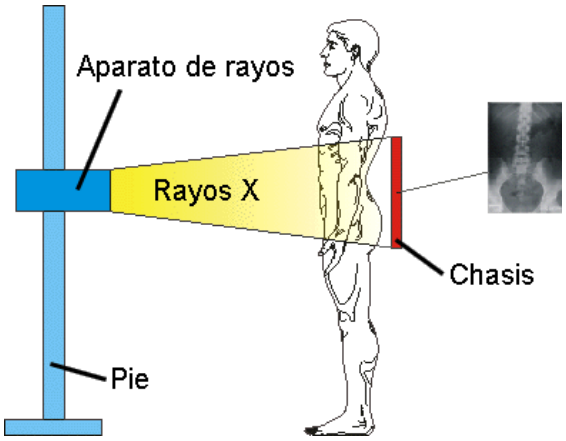


Figura 2. Formación de imágenes

Página web: https://rinconeducativo.org/contenidoextra/rayos_x/radiografas.html

En una radiografía, se identifican cinco densidades radiológicas básicas que se representan en distintos tonos de gris:

- Aire: Negro.
- Grasa: Gris oscuro.
- Agua/Partes Blandas: Gris claro.
- Calcio/Hueso: Blanco.
- Metal: Blanco opaco (este no se encuentra naturalmente en el organismo).

Figura 3. Las cinco densidades radiológicas

Página web: https://radiologiaclub.com/2021/02/01/animales-vistos-por-imagenes-radiologicas/

Actualmente, la radiología digital ha reemplazado la analógica, permitiendo obtener imágenes directamente en formato digital. Estas imágenes pueden almacenarse y compartirse a través de sistemas PACS (Sistema de Comunicación y Archivo de Imágenes).

Además, los hospitales utilizan tecnologías avanzadas como tomografía computarizada (TC), resonancia magnética (RM) y ecografía para obtener imágenes diagnósticas digitales [4].

2.1.4. Importancia de la radiografía en el diagnóstico médico

La radiografía es una herramienta crucial en el diagnóstico médico moderno. Permite a los profesionales de la salud visualizar el interior del cuerpo para detectar una variedad de afecciones, desde fracturas óseas hasta enfermedades pulmonares. Además, guía procedimientos terapéuticos como la radioterapia contra el cáncer. Aunque ofrece beneficios significativos, como diagnósticos más rápidos y precisos, y procedimientos menos invasivos, es esencial tener en cuenta los riesgos asociados, como la exposición a la radiación y posibles reacciones adversas a los agentes de contraste. Por lo tanto, su uso debe ser evaluado cuidadosamente por los profesionales de la salud, equilibrando los beneficios con los riesgos para cada paciente individualmente. En resumen, la radiografía sigue siendo una herramienta indispensable en el ámbito médico, con un papel vital en el diagnóstico y tratamiento de diversas condiciones. Su continua evolución tecnológica promete un futuro aún más prometedor en la práctica médica [5].

2.2. Sistema musculoesquelético

2.2.1. Estructura y función del sistema musculoesquelético

El sistema musculoesquelético es un componente esencial del cuerpo humano, compuesto por huesos, músculos, cartílagos, tendones, ligamentos, articulaciones y otros tejidos conectivos. Este sistema es responsable de dar forma, soporte y estabilidad al cuerpo, facilitando el movimiento y protegiendo los órganos internos vitales.

Los huesos, que forman el esqueleto humano, son estructuras rígidas compuestas por varios tipos de tejidos, como el tejido óseo, cartílago, tejido conectivo denso, vasos sanguíneos y nervios. Estos órganos complejos desempeñan funciones cruciales, incluyendo el soporte del cuerpo, la facilitación del movimiento, la protección de los órganos internos, la producción de células sanguíneas y el almacenamiento de minerales.

Los músculos, unidos a los huesos mediante tendones, son tejidos blandos capaces de contraerse y generar movimiento. Además de contribuir a la postura y estabilidad del cuerpo, los músculos generan calor. Existen tres tipos principales de tejido muscular: músculo

esquelético, músculo cardíaco y músculo liso. El músculo esquelético es el tipo de tejido muscular que forma la mayor parte del sistema musculoesquelético y su control es voluntario.

Las articulaciones son estructuras donde se encuentran dos o más huesos, permitiendo movimiento y flexibilidad, y ofreciendo una conexión mecánica entre los huesos. Según su estructura y función, las articulaciones pueden ser fibrosas, cartilaginosas o sinoviales.

Los ligamentos y tendones son tejidos conectivos esenciales en el sistema musculoesquelético. Los ligamentos unen huesos con huesos, estabilizando las articulaciones, mientras que los tendones conectan músculos con huesos, transmitiendo la fuerza muscular para producir movimiento.

El cartílago es un tejido conectivo flexible que recubre y protege las superficies articulares, permitiendo un movimiento suave entre los huesos. Además, el cartílago da forma y soporte a otras estructuras del cuerpo, como las orejas y la nariz.

El sistema musculoesquelético es fundamental para la movilidad y la actividad física, posibilitando desde movimientos simples, como levantar un objeto, hasta acciones más complejas, como correr o bailar [6].

Figura 4. Sistema musculoesquelético

Página web: https://radiologiaclub.com/2021/02/01/animales-vistos-por-imagenes-radiologicas/

2.2.2. Enfermedades y lesiones comunes

Los trastornos musculoesqueléticos abarcan más de 150 condiciones que afectan al sistema locomotor. Estos trastornos incluyen desde problemas repentinos y de corta duración como fracturas, esguinces y distensiones, hasta enfermedades crónicas que pueden llevar a limitaciones de la capacidades funcionales e incapacidad permanente.

Estos trastornos suelen manifestarse con dolor persistente y limitaciones en la movilidad, destreza y funcionamiento general, lo cual reduce la capacidad laboral de las personas. Pueden afectar a las articulaciones, como en casos de artrosis, artritis reumatoide, artritis psoriásica, gota y espondilitis anquilosante; a los huesos, como en osteoporosis, osteopenia, fracturas por fragilidad ósea y fracturas traumáticas; a los músculos, como en la sarcopenia; y a la columna vertebral, causando dolor de espalda y cuello. Además, pueden implicar varios sistemas o regiones del cuerpo, con manifestaciones como el dolor regional o generalizado y enfermedades inflamatorias, incluyendo trastornos del tejido conectivo o vasculitis, que tienen manifestaciones musculoesqueléticas, como el lupus eritematoso sistémico [7].

2.2.3. Técnicas de diagnóstico por imagen

Se emplean diversas pruebas de diagnóstico por imagen para detectar y evaluar trastornos musculoesqueléticos.

Radiografías: Son generalmente las primeras en realizarse. Las radiografías son útiles para detectar anomalías óseas y evaluar áreas con dolor, deformidades o posibles anomalías. Ayudan a diagnosticar fracturas, tumores, traumatismos, infecciones y deformidades como la displasia del desarrollo de la cadera. También pueden indicar ciertas artropatías, como la artritis reumatoide o la artrosis. Sin embargo, las radiografías simples no muestran los tejidos blandos. La artrografía, que utiliza una sustancia radiopaca, es otra técnica radiológica usada para visualizar estructuras articulares internas, pero la resonancia magnética (RM) la ha sustituido en muchos casos.

Figura 5. Sistemas de rayos X
Página web: ttps://www.elhospital.com/es/noticias/sistemas-de-rayos-x-analogicos-configurables-q-rad

Gammagrafía ósea: Este método usa una sustancia radiactiva para detectar fracturas no visibles en radiografías, infecciones óseas o tumores. Aunque identifica problemas óseos, no diferencia entre fracturas, tumores o infecciones.

Tomografía computarizada (TC) y resonancia magnética (RM): Ambas ofrecen más detalle que las radiografías simples. La RM es útil para diagnosticar problemas en músculos, ligamentos y tendones, y se prefiere para detectar lesiones graves de tejidos blandos. La TC es una alternativa cuando la RM no está disponible o no es recomendable, proporcionando imágenes detalladas de huesos y detectando fracturas no evidentes en radiografías.

Figura 6. Sistema de tomografía computarizada (TC)
Página web: https://medsystems.es/sistema-de-tomografa-computarizada-toshiba-aquilion-64-p-328.html

Figura 7. Sistema de resonancia magnética (RM)
Página web: https://www.philips.com.pe/healthcare/solutions/magnetic-resonance

Densitometría ósea (DXA): Es el método más preciso para evaluar la densidad ósea, utilizado para diagnosticar osteopenia y osteoporosis, predecir el riesgo de fracturas y monitorizar tratamientos. La prueba es rápida, indolora y conlleva poca radiación.

Ecografía: Cada vez más utilizada para identificar anomalías e inflamaciones en articulaciones y tendones, así como para guiar inyecciones o extracciones de líquido articular. Es una alternativa menos costosa y sin radiación en comparación con la TC y la RM.

Figura 8. Sistema de ecografía
Página web: https://www.medimaging.es/ultrasonido/articles/294754422/disenan-sistema-para-ecografia-con-funciones-para-mama-y-tiroides.html

Estas técnicas permiten diagnosticar y evaluar con precisión diversas condiciones musculoesqueléticas, facilitando un tratamiento adecuado [8].

2.2.4. Radiografía musculoesquelética

La radiografía musculoesquelética es una técnica diagnóstica utilizada para obtener imágenes de los huesos, articulaciones y músculos o tejidos blandos. Esta técnica puede aplicarse a extremidades completas, como brazos o piernas, o a partes específicas como el pie, tobillo, pierna, mano, muñeca, antebrazo y brazo. También se utiliza para visualizar los huesos y articulaciones de la columna vertebral, la pelvis, los hombros y el cráneo. Su objetivo principal es identificar alteraciones en los huesos, fracturas o tumores, e inflamaciones en las articulaciones [9].

2.2.5. Radiografía musculoesquelética en atención primaria

La radiografía musculoesquelética juega un papel crucial en la atención primaria, siendo una herramienta diagnóstica esencial para los médicos. Se utiliza para el diagnóstico y manejo de diversas afecciones y patologías que afectan al sistema musculoesquelético, permitiendo una evaluación inicial rápida y eficaz de las lesiones óseas, articulares y de partes blandas.

Estas pruebas de imagen suelen ser el primer paso en el diagnóstico de afecciones como fracturas, luxaciones, artritis y otras patologías óseas y articulares. Son relativamente accesibles y rápidas de realizar, lo que permite una evaluación inmediata del estado del paciente. En comparación con otras técnicas de imagen más avanzadas como la resonancia magnética o la tomografía computarizada, las radiografías son más económicas y ofrecen una excelente herramienta diagnóstica inicial.

Los resultados de las radiografías pueden ayudar a los médicos de atención primaria a determinar el mejor tratamiento, ya sea un tratamiento conservador, la derivación a especialistas o una intervención quirúrgica. Es crucial que los médicos de atención primaria utilicen las radiografías musculoesqueléticas de forma apropiada, siguiendo las directrices clínicas basadas en la evidencia para evitar pruebas innecesarias que no alterarán el tratamiento del paciente.

La interpretación de radiografías requiere conocimientos y experiencia. La formación continuada y la posibilidad de consultar con radiólogos pueden mejorar la precisión diagnóstica en la atención primaria. Sin embargo, con la introducción de una herramienta de inteligencia artificial, como la que se desarrolla en este proyecto, que clasifica imágenes radiográficas indicando si son normales o anormales, el diagnóstico sería más fiable y rápido.

Esta innovación reduce la necesidad de consultar con un radiólogo, permitiendo a los médicos de atención primaria tomar decisiones informadas de manera más eficiente.

En conclusión, la radiografía musculoesquelética es una valiosa herramienta diagnóstica en atención primaria que, utilizada adecuadamente, puede mejorar significativamente el manejo de los pacientes con patologías musculoesqueléticas. La incorporación de herramientas de inteligencia artificial en este proceso no solo optimiza la precisión diagnóstica, sino que también acelera el tiempo de respuesta, mejorando la calidad del cuidado médico [10].

2.3. Inteligencia Artificial

2.3.1. Conceptos básicos de redes neuronales

Las redes neuronales son una herramienta muy eficaz en el campo de la inteligencia artificial y el aprendizaje automático. Basadas en el funcionamiento del cerebro humano, estas estructuras computacionales se utilizan para abordar una amplia variedad de problemas complejos [11].

Una red neuronal es un modelo de aprendizaje automático que toma decisiones de manera similar al cerebro humano, empleando procesos que imitan la forma en que las neuronas biológicas colaboran para identificar patrones, evaluar opciones y alcanzar conclusiones. Esta está compuesta por capas de nodos o neuronas artificiales: una capa de entrada, una o más capas ocultas y una capa de salida.

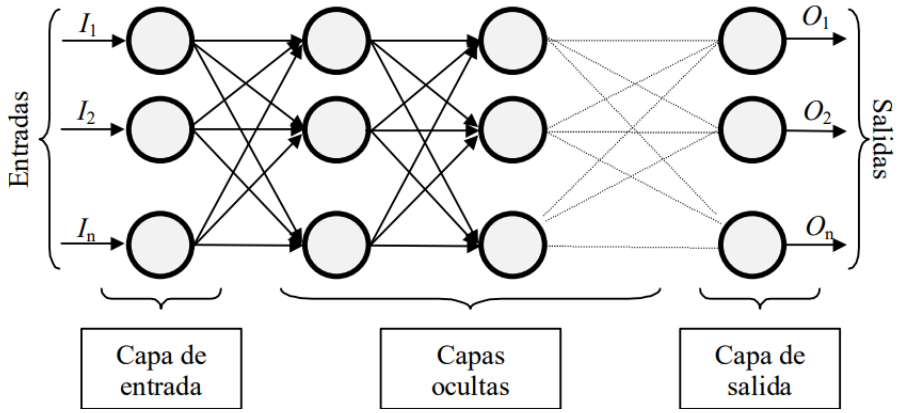


Figura 9. Ejemplo de una red neuronal totalmente conectada

Fuente: C. Alberto Ruiz Marta Susana Basualdo Autor and D. Jorge Matich, "Cátedra: Informática Aplicada a la Ingeniería de Procesos-Orientación I Redes Neuronales: Conceptos Básicos y Aplicaciones," 2001.

- Capa de entrada: Recibe información del exterior.
- Capas ocultas: Realizan el procesamiento principal de la red.
- Capa de salida: Entrega el resultado del procesamiento al exterior y transmite información a otras neuronas.

A una neurona artificial se le asigna un peso sináptico w_{ij}, con $j = 1, \dots, n$ a las entradas x_j provenientes de otras neuronas. Este peso es un valor numérico que puede variar durante el entrenamiento. Estos pesos son fundamentales para la utilidad de la red neuronal, ya que en ellos se almacena la información.

En un modelo neuronal, es necesario tener una regla de propagación para combinar las salidas de cada neurona con los pesos establecidos por el patrón de conexión. Esto determina la valoración de las entradas que recibe cada neurona. Generalmente, se realiza una suma de las entradas, considerando el peso sináptico asociado a cada una, aunque también se pueden utilizar otras operaciones.

$$h_i(x_1, \dots, x_n, w_{i1}, \dots, w_{in} = \sum_{i=1}^{n} w_{ij} x_j$$

Ecuación 1. Regla de propagación

El valor obtenido mediante la regla de propagación se filtra a través de una función conocida como función de activación, que determina la salida de la neurona. La elección de la función de activación depende del objetivo de entrenamiento de la red neuronal. En la Figura 10 se presentan las funciones de activación más comunes [12].

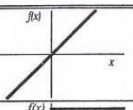
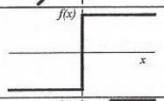
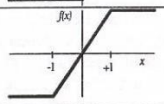
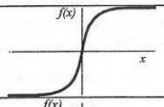
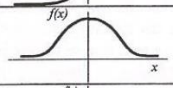
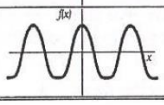
	Función	Rango	Gráfica
Identidad	$y = x$	$[-\infty, +\infty]$	
Escalón	$y = sign(x)$ $y = H(x)$	$\{-1, +1\}$ $\{0, +1\}$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	$[-1, +1]$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = tgh(x)$	$[0, +1]$ $[-1, +1]$	
Gaussiana	$y = Ae^{-Bx^2}$	$[0,+1]$	
Sinusoidal	$y = A\,\mathrm{sen}(\omega x + \varphi)$	$[-1,+1]$	

Figura 10. Funciones de activación más utilizadas

Fuente: Edgar Serna M., *DESARROLLO E INNOVACIÓN EN INGENIERÍA*. Medellín, Antioquia: Editorial IAI, 2017

2.3.2. Clasificación de imágenes con IA

Para la realización de este proyecto se ha utilizado una red neuronal convolucional (CNN), también conocida como ConvNet. Este algoritmo especializado de aprendizaje profundo está diseñado para tareas que requieren el reconocimiento de objetos, como detección, segmentación o clasificación de imágenes, objetivo principal de este trabajo.

Las redes neuronales convolucionales se componen de tres tipos principales de capas:

- Capa convolucional

  La capa convolucional es el componente principal de una CNN (Red Neuronal Convolucional) y donde se realizan la mayoría de los cálculos. Para funcionar, requiere datos de entrada, un filtro y un mapa de características. La entrada suele ser una imagen en color, representada como una matriz de píxeles 3D (altura, anchura y profundidad, que corresponden a los canales RGB). Un filtro, también llamado kernel, es una matriz 2D de pesos que se desplaza por la imagen en un proceso llamado convolución, detectando características específicas.

  El filtro, generalmente de tamaño 3x3, se aplica a secciones de la imagen y calcula productos escalares entre los píxeles y los pesos del filtro, generando un mapa de características o mapa de activación. Este filtro se mueve por la imagen según un parámetro llamado stride (el número de píxeles que avanza en cada paso). Los pesos

del filtro se ajustan durante el entrenamiento mediante retropropagación y descenso del gradiente, pero antes del entrenamiento se deben configurar tres hiperparámetros clave:

- Número de filtros: Afecta a la profundidad de la salida, generando múltiples mapas de características.
- Stride: La distancia que el filtro se mueve sobre la imagen; un stride mayor reduce el tamaño de la salida.
- Zero-padding: Añade ceros alrededor de la imagen para ajustar mejor los filtros. Hay tres tipos:
  - Valid padding: Sin padding, se descartan convoluciones incompletas.
  - Same padding: Mantiene el tamaño de la salida igual al de la entrada.
  - Full padding: Aumenta el tamaño de la salida añadiendo ceros en los bordes.

Tras cada operación de convolución, la CNN aplica una transformación de unidad lineal rectificada (ReLU) al mapa de características, añadiendo no linealidad al modelo.

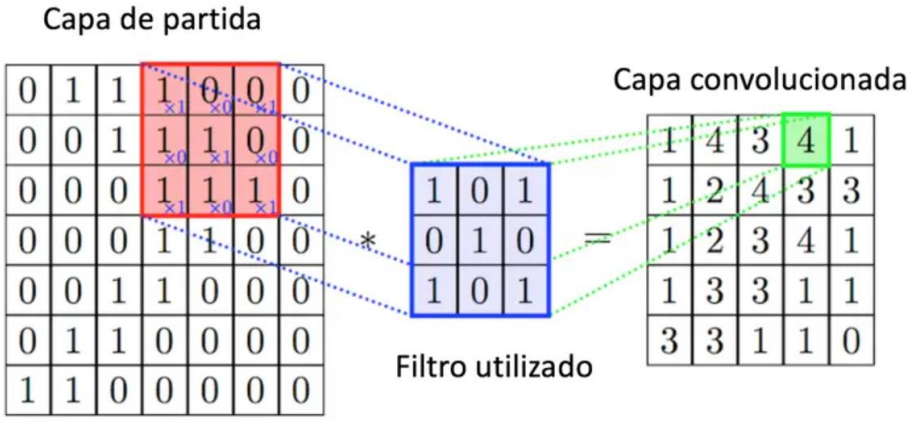


Figura 11. Operación de convolución

Página web: https://www.diegocalvo.es/red-neuronal-convolucional/

- Capa de agrupación

  La capa de agrupación, o submuestreo, reduce la dimensión mediante la reducción del número de parámetros de entrada. Al igual que la capa convolucional, utiliza un filtro que barre toda la entrada, pero sin pesos. En lugar de eso, el filtro aplica una

función de agregación sobre los valores del campo receptivo, generando la matriz de salida. Hay dos tipos principales de agrupación:

  - Agrupación máxima: El filtro selecciona el valor máximo del campo receptivo y lo envía a la matriz de salida. Este método es más común que la agrupación media.
  - Agrupación media: El filtro calcula el valor promedio dentro del campo receptivo y lo envía a la matriz de salida.

Aunque se pierde información durante la agrupación, esta capa tiene varios beneficios, como la reducción de la complejidad del modelo, la mejora de la eficiencia y la disminución del riesgo de sobreajuste.

- Capa totalmente conectada

  La capa totalmente conectada, conecta directamente cada nodo de la capa anterior con cada nodo de la capa de salida, a diferencia de las capas parcialmente conectadas. Esta capa se encarga de la clasificación, utilizando las características extraídas por las capas anteriores y sus diferentes filtros.

3. Tecnologías utilizadas

En esta sección, se describen las principales herramientas y plataformas de software empleadas durante el desarrollo de este proyecto. Estas herramientas fueron esenciales para la implementación, gestión y evaluación de este modelo de inteligencia artificial para la clasificación de imágenes radiográficas.

3.1. Software utilizado

3.1.1. Pytorch

PyTorch es un marco de deep learning de código abierto diseñado para crear redes neuronales, combinando la biblioteca de machine learning de Torch con una API de alto nivel en Python. Destacado por su flexibilidad y facilidad de uso, es ampliamente adoptado en las comunidades académicas y de investigación.

PyTorch soporta diversas arquitecturas de redes neuronales, desde regresión lineal simple hasta modelos complejos como redes convolucionales y transformadores generativos, aplicados en visión por computador y procesamiento del lenguaje natural. Su integración con Python y la disponibilidad de bibliotecas de modelos preconfigurados, incluyendo modelos preentrenados, permiten a los científicos de datos desarrollar y ejecutar redes de deep learning de manera eficiente, reduciendo el tiempo dedicado a codificación y estructura matemática. Una característica clave de PyTorch es la capacidad de ejecutar y probar partes del código en tiempo real, lo que agiliza la creación de prototipos y la depuración de grandes modelos de deep learning.

La estructura matemática y de programación de PyTorch facilita y acelera los flujos de trabajo de aprendizaje automático, sin limitar la complejidad o el rendimiento de las redes neuronales profundas [13].

En este proyecto, se utiliza PyTorch para desarrollar el clasificador de imágenes radiográficas, diferenciando entre normales y anormales. La capacidad de PyTorch para ejecutar y probar código en tiempo real es útil para iterar rápidamente y depurar el modelo, mejorando tanto su precisión como su rendimiento. Además, las bibliotecas preconfiguradas y modelos preentrenados disponibles en PyTorch permiten acelerar significativamente el proceso de desarrollo y experimentación.

Figura 12. Logo de Pytorch

3.1.2. Jupyter Notebook

Jupyter Notebook es una herramienta de código abierto ampliamente utilizada para desarrollar y compartir código en lenguajes como Python, R y Julia. Permite integrar texto, código, gráficos y otros elementos multimedia en un documento interactivo, siendo ideal para la investigación reproducible.

Con Jupyter Notebook, se puede escribir y ejecutar código en celdas individuales, facilitando la exploración y el análisis de datos de manera interactiva. Además, permite agregar texto explicativo, ecuaciones matemáticas, imágenes y visualizaciones para crear documentos coherentes y fáciles de entender.

Jupyter Notebook organiza los documentos en celdas de diferentes tipos: código, texto, Markdown, ecuaciones, etc. Al ejecutar una celda de código, el resultado se muestra directamente debajo, proporcionando una interacción rápida y dinámica con el código y los datos, lo cual es especialmente beneficioso en la ciencia de datos y el análisis exploratorio.

La herramienta ofrece una variedad de funciones, como la importación de bibliotecas de diferentes lenguajes, la visualización de datos en gráficos interactivos y la exportación de notebooks a distintos formatos. Además, Jupyter Notebook facilita la colaboración en tiempo real a través de la integración con plataformas git como GitHub, GitLab y Bitbucket, lo cual es esencial para muchos proyectos [14].

En este proyecto, se utiliza Jupyter Notebook para desarrollar y probar todo el código del clasificador de imágenes radiográficas. La capacidad de ejecutar y probar código en celdas individuales permite avanzar de manera incremental, verificando que cada parte del proyecto funcionara correctamente antes de pasar a la siguiente.

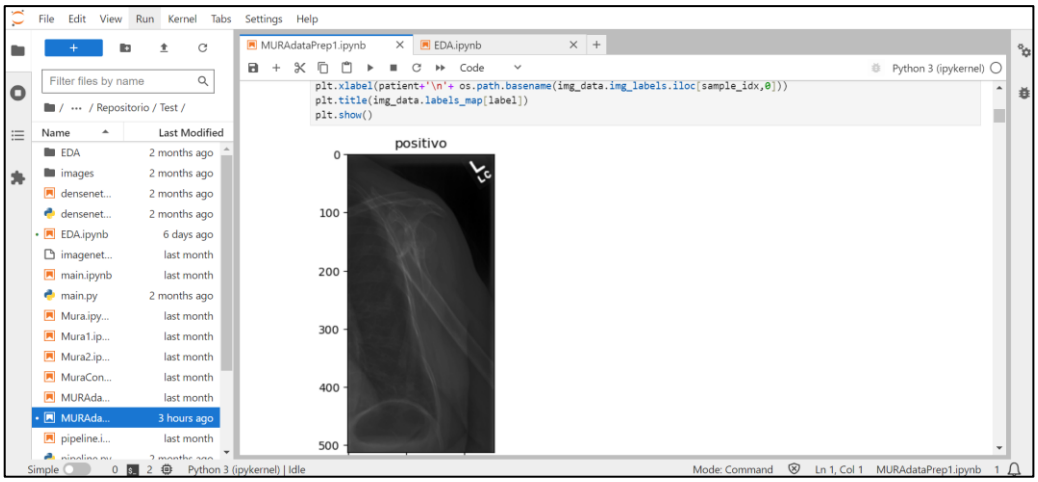


Figura 13. Interfaz de Jupyter Notebook

3.1.3. GitHub

GitHub es una plataforma de desarrollo colaborativo en la nube que utiliza el sistema de control de versiones Git para alojar proyectos. Permite a los desarrolladores almacenar y gestionar su código mientras mantiene un registro detallado de los cambios realizados. Habitualmente, el código es de acceso público, lo que facilita la colaboración en proyectos compartidos y el seguimiento de su evolución. Además, GitHub actúa como una red social, conectando a desarrolladores con usuarios. Los usuarios pueden descargar programas o aplicaciones y contribuir a su desarrollo mediante sugerencias de mejoras y participación en discusiones en foros temáticos [15].

En este proyecto, se utiliza GitHub para mantener un registro continuo del trabajo a medida que se avanza. Se sube regularmente el código a un repositorio en GitHub para asegurar que todos los cambios estén guardados de forma segura y accesible desde cualquier dispositivo.

Esto no solo permite tener una copia de seguridad del proyecto en todo momento, sino que también facilita la supervisión y revisión por parte del tutor.

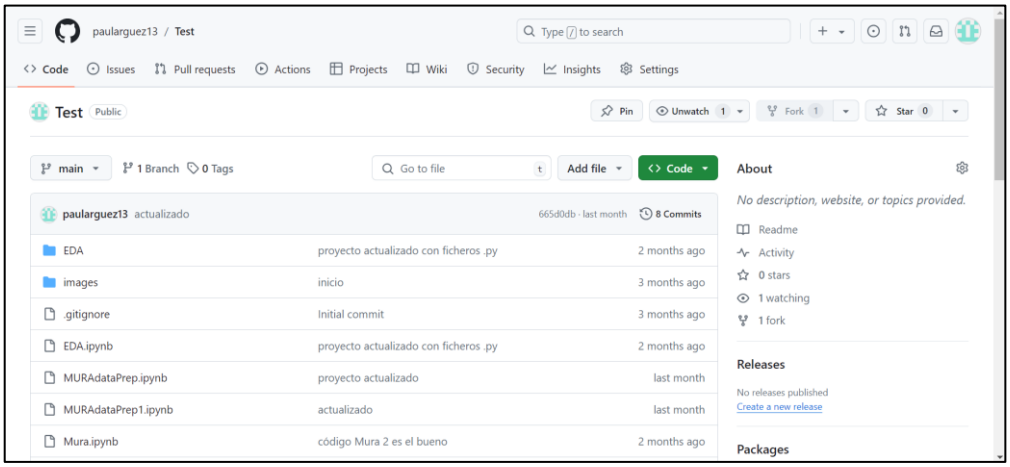


Figura 14. Interfaz de GitHub

3.2. Hardware utilizado

3.2.1. Ordenador Asus Zenbook UX425EA

Las características técnicas más destacadas del ordenador son:

- Procesador: 11th Gen Intel(R) Core (TM) i7-1165G7 @ 2,80GHz 2,80 GHz
- RAM: 16,0 GB
- Sistema operativo: Windows 11 de 64 bits, procesador basado en x64

3.2.2. Ordenador DELL Precision 5820

Las características técnicas más destacadas del ordenador son:

- Procesador: Intel Xeon(R) W-2245 CPU @3,90GHz x 16
- RAM: 128 GB
- Sistema operativo: Ubuntu 20.04.6 LTS de 64 bits

4. Base de datos MURA (Musculoskeletal Radiographs)

Para la realización de este proyecto, es necesario identificar una base de datos pública de imágenes radiográficas adecuada para entrenar la red neuronal. La disponibilidad de datos de alta calidad y en cantidad suficiente es crucial para el desarrollo y validación de modelos de inteligencia artificial (IA) efectivos. Durante una extensa búsqueda y evaluación de diversas bases de datos públicas, se consideraron varios factores clave, como la extensión, diversidad, accesibilidad y relevancia clínica de las bases de datos disponibles.

Por todas estas razones, se ha elegido la base de datos MURA (Musculoskeletal Radiographs). MURA es una de las mayores bases de datos públicas de radiografías musculoesqueléticas, abarcando imágenes de múltiples partes del cuerpo. Su fácil accesibilidad y excelente documentación facilitan la comprensión de la estructura y el contenido de los datos. Las imágenes de MURA son de gran importancia clínica, ya que representan una variedad de situaciones comunes en la radiología musculoesquelética. Esta característica asegura que los modelos entrenados con esta base de datos sean altamente aplicables en entornos clínicos reales, mejorando significativamente la capacidad para clasificar imágenes como normales o anormales.

4.1. Descripción de la base de datos

La base de datos MURA es un extenso conjunto de datos de radiografías musculoesqueléticas, compuesto por 14.656 estudios de la extremidad superior pertenecientes a 11.967 pacientes y compilada por la Universidad de Stanford [16]. Cada estudio incluye una o más imágenes etiquetadas manualmente por radiólogos como normales o anormales, acumulando un total de 40.005 imágenes. Estas imágenes corresponden a una de las siete regiones de las extremidades superiores: hombro, húmero, codo, antebrazo,

muñeca, mano y dedo. El conjunto de datos se divide en 8.941 estudios radiográficos musculoesqueléticos normales y 5.715 anormales. La Tabla 1 resume la distribución de estudios normales y anormales. Dado su tamaño, MURA es uno de los conjuntos de datos públicos de imágenes radiográficas más grandes disponibles.

Study	Train		Validation		Total
	Normal	Abnormal	Normal	Abnormal	
Elbow	1094	660	92	66	1912
Finger	1280	655	92	83	2110
Hand	1497	521	101	66	2185
Humerus	321	271	68	67	727
Forearm	590	287	69	64	1010
Shoulder	1364	1457	99	95	3015
Wrist	2134	1326	140	97	3697
Total No. of Studies	8280	5177	661	538	14656

Tabla 1. Resumen de la distribución de estudios normales y anormales

Las imágenes incluidas en MURA están anonimizadas y cumplen con la normativa HIPAA, siendo extraídas del sistema PACS (Sistema de Comunicación y Archivo de Imágenes) del Hospital de Stanford. Cada estudio fue etiquetado como normal o anormal por radiólogos certificados del Hospital de Stanford en el momento de la interpretación radiográfica clínica, realizada en un entorno de radiología diagnóstica entre los años 2001 y 2012. Este proceso de etiquetado se llevó a cabo durante la interpretación de imágenes DICOM en pantallas PACS de calidad médica, con una resolución mínima de 3 megapíxeles, luminancia máxima de 400 cd/m^2 y mínima de 1 cd/m^2, un tamaño de píxel de 0,2 y una resolución nativa de 1500x2000 píxeles. Cabe destacar que las imágenes clínicas varían en resolución y relación de aspecto [17].

4.2. Estructura de la base de datos

La base de datos se divide en dos conjuntos: entrenamiento (train) y validación (valid). Cada conjunto se divide a su vez en 7 subconjuntos, correspondientes a las diferentes regiones de las extremidades superiores: hombro (XR_SHOULDER), húmero (XR_HUMERUS), codo (XR_ELBOW), antebrazo (XR_FOREARM), muñeca (XR_WRIST), mano (XR_HAND) y dedo (XR_FINGER). En cada carpeta de extremidad hay muchas carpetas correspondientes a los pacientes, y dentro de cada carpeta de cada uno está la del estudio con las imágenes. Además, la base de datos incluye los archivos “train_image_paths” y “valid_image_paths”,

que contienen las rutas de las imágenes radiográficas, y los archivos “train_labeled_studies” y “valid_labeled_studies”, que incluyen la ruta de cada estudio junto con su etiqueta correspondiente, indicando si es positivo (anormal) o negativo (normal).

El conjunto de entrenamiento cuenta con:

- Pacientes: 11.184
- Estudios: 13.457
- Imágenes: 36.808

El conjunto de validación incluye:

- Pacientes: 783
- Estudios: 1.199
- Imágenes: 3.197

Es importante destacar que no hay solapamiento de pacientes entre los conjuntos de entrenamiento y validación. Esto asegura que los modelos desarrollados y evaluados con estos datos puedan generalizar mejor a nuevos casos clínicos, al no estar entrenados y validados con datos provenientes de los mismos pacientes.

5. Metodología

5.1. Análisis exploratorio de datos

En este apartado se presenta un análisis exploratorio de la base de datos MURA, para comprender mejor las características y tendencias de los datos recopilados. El código correspondiente a este apartado se encuentra en el Anexo 11.1.

5.1.1. Distribución de las clases

La base de datos MURA contiene 8.941 estudios etiquetados como normales y 5.715 anormales. Es importante destacar que esta base de datos se organiza en dos carpetas distintas: "train" y "valid". En la carpeta "train" se encuentran 8.280 estudios normales (61.53%) y 5.177 estudios anormales (38.47%), mientras que en la carpeta "valid" hay 661 estudios normales (55.13%) y 538 estudios anormales (44.87%). Estos números se pueden observar detalladamente en la Figura 15.

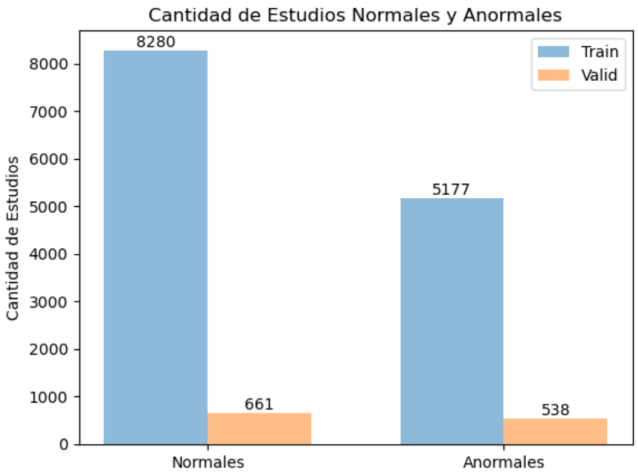


Figura 15. Cantidad de estudios normales y anormales

La base de datos clasifica los estudios radiológicos en siete regiones distintas de las extremidades superiores: hombro, húmero, codo, antebrazo, muñeca, mano y dedo. A

continuación, se presenta un desglose detallado de la cantidad de estudios normales y anormales en cada una de estas regiones en conjunto de datos de entrenamiento (véase Figura 16).

- Hombro: 1364 (48,35%) estudios normales, 1457 (51,65%) estudios anormales.
- Húmero: 321(54,22%) estudios normales, 271(45,78%) estudios anormales.
- Codo: 1094 (62,37%) estudios normales, 660 (37,63%) estudios anormales.
- Antebrazo: 590 (67,27%) estudios normales, 287 (32,73%) estudios anormales.
- Muñeca: 2134 (61,68%) estudios normales, 1326 (38,32%) estudios anormales.
- Mano: 1497 (74,18%) estudios normales, 521(25,82%) estudios anormales.
- Dedo: 1280 (66,15%) estudios normales, 655 (33,85%) estudios anormales.

En el conjunto de entrenamiento se puede observar que el hombro es la única región donde la incidencia de anomalías (51,65%) supera a los estudios normales. Esto sugiere una mayor prevalencia de problemas en esta área. Aunque la incidencia de anomalías en el húmero es menor que en el hombro, sigue siendo relativamente alta (45,78%) en comparación con otras regiones, lo que indica que el húmero también es una región con una notable prevalencia de anomalías.

En contraste, la mano presenta la menor incidencia de anomalías (25,82%) entre todas las regiones analizadas, lo que sugiere que los problemas en la mano son menos comunes o detectables en esta base de datos. El antebrazo también muestra una incidencia relativamente baja de anomalías (32,73%), lo que indica una menor prevalencia de problemas en esta región en comparación con otras partes del cuerpo.

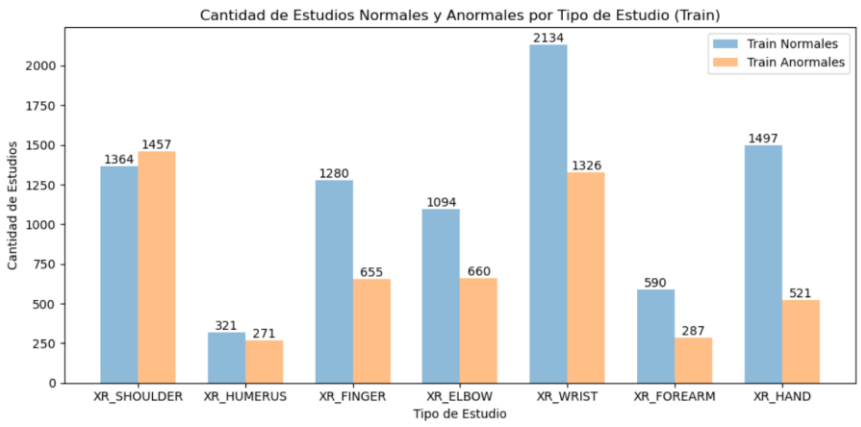


Figura 16. Cantidad de estudios normales y anormales por tipo de estudio (Train)

A continuación, se realizará el mismo análisis para el conjunto de validación (véase Figura 17).

- Hombro: 99 (51,03%) estudios normales, 95 (48,97%) estudios anormales.
- Húmero: 68 (50,37%) estudios normales, 67 (49,63%) estudios anormales.
- Codo: 92 (58,23%) estudios normales, 66 (41,77%) estudios anormales.
- Antebrazo: 69 (51,88%) estudios normales, 64 (48,12%) estudios anormales.
- Muñeca: 140 (59,07%) estudios normales, 97 /40,93%) estudios anormales.
- Mano: 101 (60,48%) estudios normales, 66 (39,52%) estudios anormales.
- Dedo: 92 (52,57%) estudios normales, 83 (47,43%) estudios anormales.

En el conjunto de validación se puede observar como en el conjunto de entrenamiento que las áreas que muestran mayor número de anomalías son el hombro y el húmero, ambas regiones muestran una alta prevalencia de anomalías, con incidencias cercanas al 50% (48,97% y 49,63%, respectivamente).

En cambio, para los estudios con menor incidencia de anomalías, se obtienen datos diferentes, siendo la mano nuevamente la que presenta menos problemas con un 39,52%, seguida de la muñeca (40,93%).

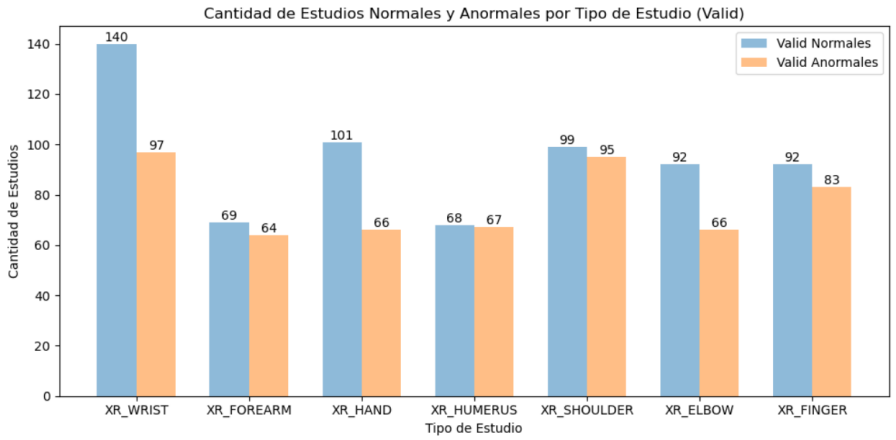


Figura 17. Cantidad de estudios normales y anormales por tipo de estudio (Valid)

5.1.2. Visualización de datos

La visualización de datos es fundamental en la investigación de inteligencia artificial aplicada a la clasificación de imágenes médicas. Esta sección se centra en la representación

de ejemplos visuales que ilustran tanto estudios normales como anormales en diversas regiones anatómicas.

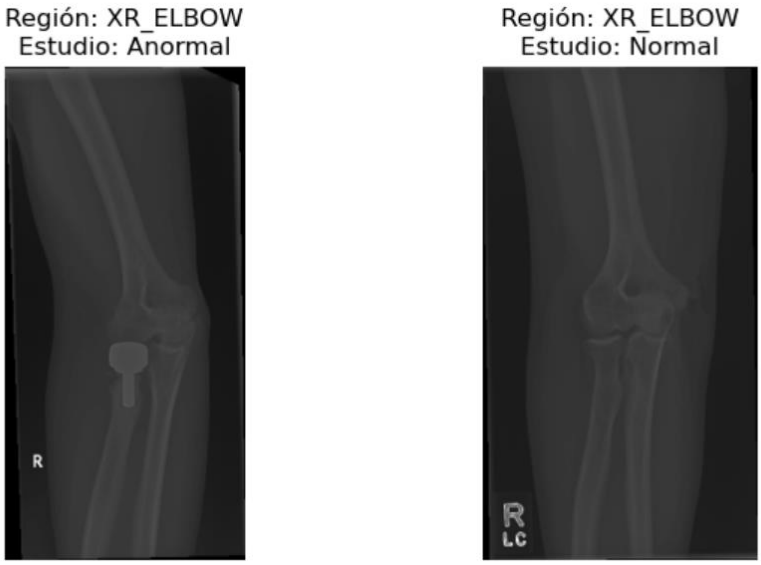


Figura 18. Estudio anormal (izq.) y normal (dcha.) del codo

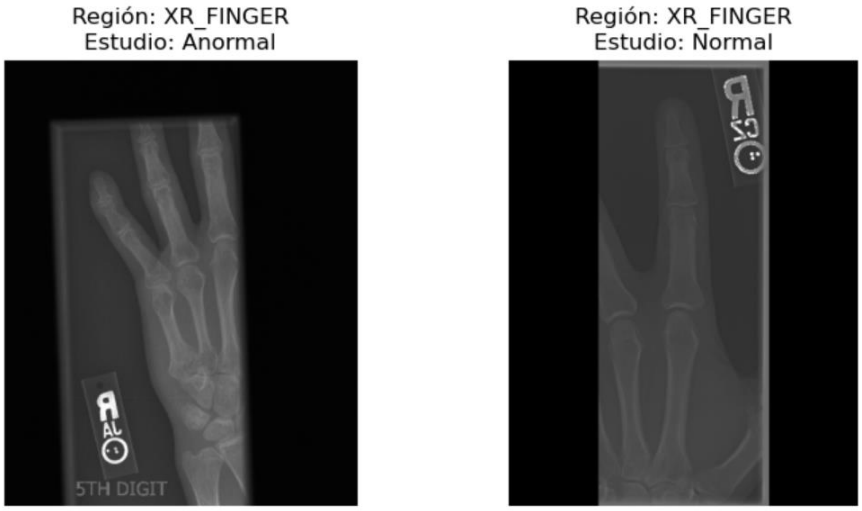


Figura 19. Estudio anormal (izq.) y normal (dcha.) del dedo

Figura 20. Estudio anormal (izq.) y normal (dcha.) del antebrazo

Figura 21. Estudio anormal (izq.) y normal (dcha.) de la mano

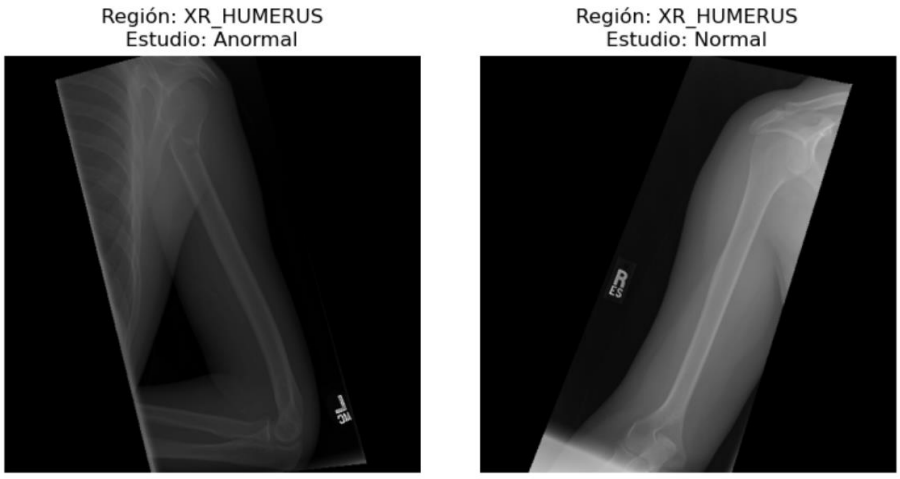


Figura 22. Estudio anormal (izq.) y normal (dcha.) del húmero

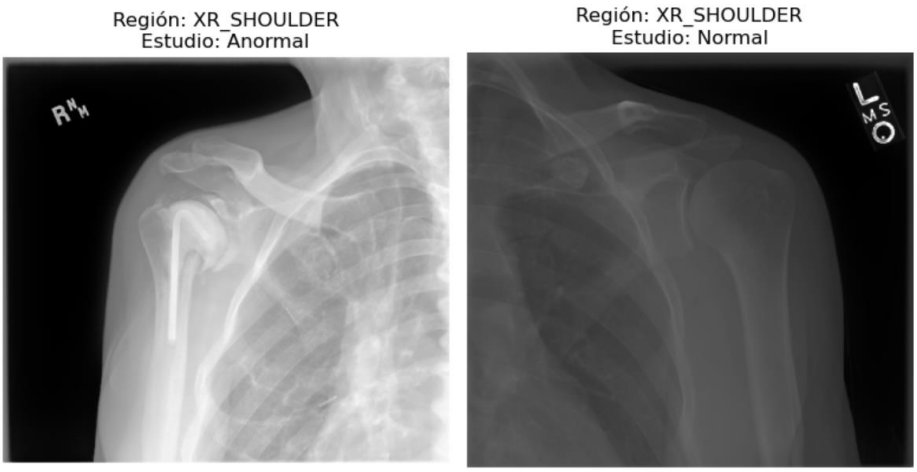


Figura 23. Estudio anormal (izq.) y normal (dcha.) del hombro

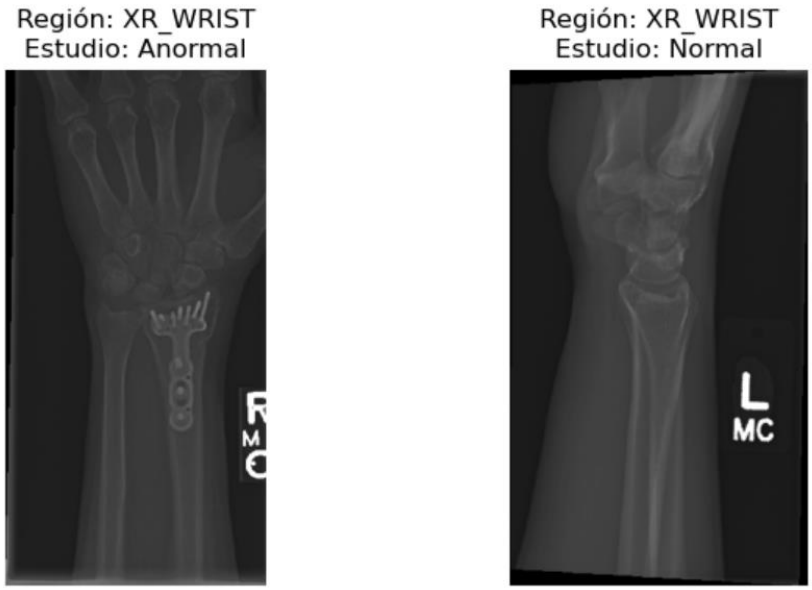


Figura 24. Estudio anormal (izq.) y normal (dcha.) de la muñeca

5.1.3. Descripción estadística

En este apartado, se presenta un análisis estadístico detallado de los datos recogidos para comprender las características y la distribución de las variables estudiadas. Para ajustar este proyecto al tiempo disponible, se ha decidido centrar el trabajo en el análisis de un tipo de estudio, eligiendo la muñeca como enfoque principal. El código correspondiente a este apartado se puede observar en el Anexo 11.2.

5.1.3.1. Medidas descriptivas

En esta sección se presentan las medidas estadísticas básicas de las diferentes clases y regiones analizadas en el estudio. Estas medidas se enfocan específicamente en las imágenes correspondientes a la región de la muñeca. Se aborda tanto la escala de 0 a 255 como la escala normalizada de 0 a 1 para las intensidades de los píxeles.

- Escala de 0 a 255

  En el contexto de imágenes digitales, especialmente en formatos como imágenes en escala de grises (8 bits por canal), los valores de los píxeles varían de 0 (negro) a 255 (blanco). Cada valor intermedio representa una escala de grises entre el negro y el blanco absolutos.

  - Valores máximos y mínimos:
    - Máximo: 255
    - Mínimo:0

    Estos valores corresponden a la escala de intensidad de los píxeles en las imágenes de la muñeca, donde 0 representa el valor más oscuro y 255 el más claro.

  - Número total de píxeles y suma total:
    - Número total de píxeles en todas las imágenes: 7130201600
    - Suma total de los valores (intensidades) de los píxeles en todas las imágenes: 375043758839
  - Media muestral:

$$Media\ (\bar{x}) = \frac{\sum_{i=1}^{N} x_i}{N}$$

Ecuación 2. Media muestral

Donde $\sum_{i=1}^{N} x_i$ es la suma total de las intensidades de los píxeles y N es el número total de píxeles.

- Media: 52,5993

  - Varianza muestral:

$$Varianza(s^2) = \frac{\sum_{i=1}^{N}(x_i - \bar{x})^2}{N-1}$$

Ecuación 3. Varianza muestral

- Varianza: 114,2305

- Desviación típica:

$$Desviación\ típica\ (s) = \sqrt{Varianza}$$

Ecuación 4. Desviación típica

- Desviación típica: 10,6879

- Escala de 0 a 1 (Normalizada)

  Para este proyecto, se ha optado por normalizar los valores de los píxeles de las imágenes a una escala de 0 a 1. Esta normalización se logra dividiendo cada valor de píxel por 255, transformando así los valores originales que van de 0 a 255 a una escala donde 0 representa la mínima intensidad (negro) y 1 la máxima intensidad (blanco).

  La normalización desempeña un papel crucial al estandarizar los datos, lo que facilita la comparación entre diferentes conjuntos de imágenes o características dentro de una misma imagen. Este proceso es beneficioso en el contexto de algoritmos de aprendizaje automático y redes neuronales.

  Al trabajar con datos normalizados, estos algoritmos pueden experimentar mejoras significativas en la convergencia y estabilidad del modelo. La reducción de la escala de los datos a un rango uniforme de 0 a 1 ayuda a evitar problemas como gradientes que se vuelven demasiado grandes o pequeños, lo cual puede ralentizar significativamente la convergencia del modelo. Además, la normalización contribuye a mantener una propagación de gradientes adecuada a lo largo de las capas de redes neuronales profundas, mejorando así la estabilidad general del modelo.

  En resumen, usar valores normalizados en este proyecto facilita la manipulación y comparación de datos y mejora la eficiencia y efectividad de los algoritmos de aprendizaje automático empleados, asegurando resultados más robustos y consistentes en el análisis de las imágenes de la muñeca.

  - Media normalizada: 0,2063
  - Desviación típica normalizada: 0,0419

5.2. Preparación de datos

El código correspondiente a este apartado se encuentra en el Anexo 11.3.

5.2.1. Implementación de la clase MuraDataset

En esta sección se detalla la implementación de la clase `MuraDataset`, utilizada para gestionar el conjunto de datos MURA (Musculoskeletal Radiographs) en este proyecto. Esta clase es crucial para la carga, preprocesamiento y manejo de las imágenes, así como de las etiquetas asociadas a ellas. La implementación está basada en la estructura `torch.utils.data.Dataset`, una clase base para manejar datasets en PyTorch. A continuación se detallan los componentes claves:

- Variables de clase: Definen constantes como el nombre del dataset, archivos de anotaciones y carpetas de división de datos.
- Método `__init__`: Inicializa la clase con parámetros como el directorio raíz, tipo de estudio, si se utiliza para entrenamiento o prueba, y opciones de transformación.
- Método `__len__`: Devuelve el número de elementos en el dataset.
- Método `__getitem__`: Carga y devuelve una imagen y su etiqueta correspondiente según el índice proporcionado.

Esta clase facilita la carga y preprocesamiento de las imágenes, permitiendo un manejo eficiente del conjunto de datos MURA en el proyecto.

5.2.2. Preprocesamiento de imágenes

Antes de la fase de análisis, todas las imágenes de la muñeca fueron sometidas a un riguroso proceso de preprocesamiento para garantizar la calidad y uniformidad de los datos. Este proceso incluyó varias etapas de transformación y ajuste que son fundamentales para preparar adecuadamente las imágenes para su posterior análisis mediante algoritmos de aprendizaje automático.

- Redimensionamiento de imágenes

  Las imágenes fueron redimensionadas para ajustarse al tamaño de entrada esperado por la red neuronal. En este caso, se utilizó un tamaño de 224x224 píxeles, compatible con la arquitectura de la red DenseNet empleada.
- Conversión a punto flotante

  Originalmente en formato entero, todas las imágenes fueron convertidas a punto flotante (float) para facilitar operaciones matemáticas y cálculos precisos durante el procesamiento.

- Normalización de imágenes

  Para mejorar la convergencia y estabilidad del modelo durante el entrenamiento, las intensidades de píxeles en las imágenes fueron normalizadas. Se aplicó una normalización específica con media y desviación estándar calculadas para los datos del repositorio. En este caso particular, se utilizó una media de [0.206, 0.206, 0.206] y una desviación estándar de [0.0419, 0.0419, 0.0419] para cada uno de los tres canales RGB.

Este proceso asegura que todas las imágenes estén homogeneizadas y preparadas de manera óptima para ser utilizadas en el modelo de aprendizaje automático, garantizando resultados consistentes y confiables en el análisis de imágenes de la muñeca.

5.2.3. Creación de conjuntos de datos de entrenamiento y prueba

Después del preprocesamiento, las imágenes fueron organizadas en dos conjuntos: el conjunto de entrenamiento y el conjunto de prueba. Para esta organización, se utilizó una clase especializada denominada MuraDataset, que facilita la gestión y transformación de las imágenes para su uso.

- Conjunto de datos de entrenamiento

  Las imágenes destinadas al entrenamiento del modelo fueron recopiladas y transformadas según las especificaciones de preprocesamiento mencionadas anteriormente. Este conjunto de datos incluye imágenes de la muñeca en formato RGB y preprocesadas para asegurar su compatibilidad con la red neuronal.

  El tamaño total del conjunto de datos de entrenamiento es de 9752, proporcionando una base sólida y amplia para el aprendizaje del modelo.

- Conjunto de datos de prueba

  Similar al conjunto de entrenamiento, las imágenes de prueba fueron preprocesadas y organizadas adecuadamente. Este conjunto se utiliza para evaluar el rendimiento del modelo una vez entrenado, asegurando que los resultados sean precisos y generalizables.

  El tamaño total del conjunto de datos de prueba es de 659, permitiendo una evaluación exhaustiva del modelo en datos no vistos previamente.

Preparar y organizar estos datos asegura que las imágenes estén listas para ser usadas de manera efectiva durante el entrenamiento y la evaluación del modelo de aprendizaje automático, lo que a su vez contribuye a un análisis de imágenes de muñecas más confiable y sólido.

5.2.4. Inspección y visualización de imágenes preprocesadas

Para asegurar que el preprocesamiento de las imágenes se realizó correctamente, se implementó una función de visualización de las imágenes tensoriales. Esta función corrige la normalización y considera el número de canales para proporcionar una representación visual precisa de las imágenes preprocesadas. Se utilizó una cuadrícula de 3x3 para mostrar aleatoriamente nueve imágenes del conjunto de entrenamiento, permitiendo una inspección visual detallada.

Durante esta inspección, se visualizan detalles importantes como la forma de la imagen, el tipo de dato y los valores de píxeles arbitrarios, asegurando que las imágenes estén en el formato esperado. Además, se muestra información adicional sobre cada imagen, como la etiqueta y el identificador del paciente, para una mejor comprensión y verificación.

Este paso es crucial para validar que las imágenes se han preprocesado correctamente y están listas para ser utilizadas en el entrenamiento del modelo de aprendizaje automático.

Figura 25. Visualización de imágenes preprocesadas

5.3.Selección del modelo

En este estudio, elegir el modelo adecuado es un paso clave para analizar las imágenes de la muñeca. Se evaluaron diferentes arquitecturas de redes neuronales para encontrar la que mejor se adapta a las necesidades del proyecto, considerando factores como precisión, eficiencia computacional y capacidad de generalización del modelo.

5.3.1. Arquitecturas de redes neuronales analizadas

Se analizaron varias arquitecturas de redes neuronales convolucionales (CNN), que son especialmente efectivas para tareas de visión por computador.

DenseNet: Es una arquitectura de red neuronal convolucional (CNN) conocida por su enfoque innovador en la conexión entre capas. A diferencia de las redes convolucionales tradicionales, donde cada capa se conecta solo a la siguiente, DenseNet introduce conexiones densas en las que cada capa está conectada directamente con todas las capas subsiguientes en un bloque. Esta estructura resulta en L(L+1)/2 conexiones directas en lugar de las L conexiones típicas en redes convolucionales de L capas

Esta arquitectura presenta varias ventajas significativas: mitiga el problema del gradiente decreciente, refuerza la propagación de características a través de las capas, fomenta la reutilización de características aprendidas y reduce sustancialmente el número de parámetros en comparación con las redes convencionales. DenseNet utiliza operaciones de concatenación en lugar de sumar o promediar características, lo que ayuda a preservar y combinar información detallada en cada capa.

DenseNet se basa en bloques de convolución compuestos por capas de convolución, normalización por lotes y funciones de activación. A medida que la red se profundiza, el número de bloques densos aumenta, permitiendo la captura progresiva de características complejas y de alto nivel. Esta estructura densa promueve la reutilización de características y facilita el flujo de gradientes a través de la red, mejorando tanto el rendimiento como la eficiencia del entrenamiento. DenseNet ofrece ventajas clave en términos de eficiencia y precisión en comparación con otras arquitecturas convencionales [18] [19].

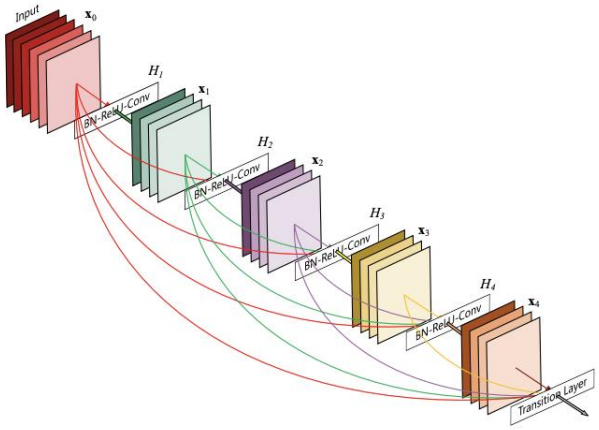


Figura 26. Ejemplo arquitectura DenseNet

Fuente: G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely Connected Convolutional Networks"

ResNet: es una arquitectura de red neuronal convolucional (CNN) destacada por su capacidad para entrenar redes extremadamente profundas con éxito. Esta aborda el problema del desvanecimiento del gradiente mediante el uso de conexiones residuales. Desarrollada para la clasificación de imágenes, ResNet utiliza bloques residuales que permiten que las capas aprendan las funciones residuales en lugar de las funciones originales. Esto facilita el entrenamiento de redes profundas sin enfrentar problemas significativos de degradación del rendimiento.

La arquitectura ResNet se basa en capas convolucionales básicas (generalmente 3x3) y utiliza max-pooling para reducir las dimensiones de las características. A medida que la red se profundiza, el número de bloques residuales aumenta, permitiendo la captura progresiva de características complejas y de alto nivel [20].

VGG: es una arquitectura de red neuronal convolucional (CNN) conocida por su estructura profunda y su diseño simple. Utiliza convoluciones 3x3 y max-pooling 2x2 en capas repetidas para profundizar la red. A medida que la red se hace más profunda, el número de filtros en las capas convolucionales aumenta, permitiendo capturar características de alto nivel y complejidad.

Esta estructura con capas pequeñas (3x3) permite a VGG aprender representaciones detalladas y abstractas de las imágenes, crucial para tareas como la clasificación de objetos en imágenes médicas. El uso extensivo de capas convolucionales y max-pooling facilita la extracción jerárquica de características, mejorando la capacidad del modelo para reconocer patrones complejos [21].

5.3.2. Justificación de la selección del modelo

Después de analizar las diversas opciones posibles, la elección definitiva ha sido DenseNet-169 debido a varias consideraciones estratégicas y técnicas que hacen que este modelo sea particularmente aplicable y efectivo en el análisis de imágenes médicas.

DenseNet-169 es una arquitectura profundamente conectada que se distingue por su estructura densa de capas. Con un total de 169 capas, esta red neuronal convolucional permite capturar y reutilizar características a múltiples niveles de abstracción. Esta característica es crucial para la detección precisa de patrones en imágenes médicas de alta

resolución, facilitando la tarea de distinguir entre imágenes normales y anormales con gran detalle y precisión.

Comparado con otras arquitecturas de redes neuronales convolucionales, DenseNet tiende a ser más eficiente en términos de uso de parámetros. Las conexiones residuales en DenseNet permiten mitigar el problema de degradación de la red a medida que se incrementa su profundidad, facilitando el entrenamiento de redes neuronales profundas sin sufrir una pérdida significativa en el rendimiento. Esto es beneficioso para este estudio, donde se requiere una red robusta capaz de aprender representaciones complejas de datos médicos.

Aunque DenseNet es conocida por su adaptabilidad a diferentes dominios y problemas, esta característica también es beneficiosa en el contexto específico de este estudio. Si bien la tarea principal es la clasificación binaria de imágenes radiográficas, la adaptabilidad de DenseNet permite su potencial uso en futuras extensiones del modelo, como la clasificación de diferentes tipos de anomalías o el análisis de otras regiones anatómicas.

En conclusión, la selección de DenseNet-169 como la arquitectura de red neuronal convolucional para este estudio se basa en su capacidad demostrada para mejorar la precisión de clasificación de imágenes médicas, su eficiencia en el uso de recursos computacionales y su adaptabilidad. Estas características hacen de DenseNet-169 una opción adecuada y prometedora para los objetivos de investigación, enfocándose específicamente en la tarea de determinar si una imagen radiográfica es normal o anormal.

5.3.3. Configuración del modelo

Para adaptar DenseNet-169 a la tarea específica de clasificación binaria de imágenes radiográficas, se realizó un ajuste de las salidas del clasificador, puesto que el número predeterminado de salidas es 1000 (véase Figura 27). Se modificó el número de salidas del clasificador utilizando “model.classifier.out_features = 2” (véase Figura 28). Esta configuración asegura que el modelo pueda generar predicciones precisas para las dos clases objetivo: imágenes normales y anormales.

Esta adaptación es esencial para alinear el modelo con los requisitos específicos del problema de clasificación binaria, facilitando la interpretación y evaluación de las predicciones del modelo en el contexto médico.

```
    )
    (classifier): Linear(in_features=1664, out_features=1000, bias=True)
  )
```

Figura 27. Clasificador con mil características de salida

```
    )
    (classifier): Linear(in_features=1664, out_features=2, bias=True)
  )
```

Figura 28. Clasificador con dos características de salida

5.4. Entrenamiento del modelo

El código de este apartado se puede observar en el Anexo 11.4.

5.4.1. Configuración de hiperparámetros

Antes de proceder con el entrenamiento del modelo, es fundamental configurar adecuadamente los hiperparámetros y los mecanismos que aseguran una gestión eficiente de los datos. En este contexto, se utilizaron DataLoaders para manejar los conjuntos de datos de entrenamiento y prueba. Estos DataLoaders son responsables de ordenar aleatoriamente las imágenes y agruparlas en lotes ("batches"), lo cual es esencial para el proceso de entrenamiento basado en gradiente estocástico.

1. Creación de DataLoaders

   Se utilizaron DataLoaders de la biblioteca PyTorch para organizar y gestionar los datos de entrenamiento y prueba. Los DataLoaders aseguran que las imágenes se presenten al modelo en lotes, facilitando el cálculo de métricas de entrenamiento y la optimización del modelo.
2. Ordenación aleatoria y agrupación en batches

   Los DataLoaders se configuraron para ordenar aleatoriamente las imágenes en cada iteración y agruparlas en lotes de 64 imágenes. Este proceso ayuda a mejorar la eficiencia del entrenamiento y a asegurar que el modelo no aprenda patrones específicos del orden de los datos.

3. Dimensiones de los batches

   En cada iteración, el DataLoader proporciona un lote completo, donde la primera dimensión del tensor representa el tamaño del lote (batch size), seguido por las dimensiones de los canales, altura y anchura de las imágenes (B, C, H, W).

- DataLoader de entrenamiento:

  El conjunto de datos de entrenamiento se gestionó utilizando un DataLoader configurado con un tamaño de lote de 64 y con la opción de ordenación aleatoria habilitada.

  - Número de batches de entrenamiento: 153

- DataLoader de Prueba:

  De manera similar, el conjunto de datos de prueba se gestionó con un DataLoader configurado con un tamaño de lote de 64 y con ordenación aleatoria.

  - Número de batches de prueba: 11

Esta configuración asegura que el modelo reciba los datos de manera eficiente y adecuada durante el proceso de entrenamiento y evaluación, optimizando el uso de los recursos computacionales y mejorando la estabilidad del entrenamiento.

5.4.2. Configuración del entorno de ejecución

Para el entrenamiento del modelo, es crucial seleccionar y configurar adecuadamente el entorno de ejecución, optimizando el uso de los recursos computacionales disponibles.

El modelo se ha configurado para ejecutarse en una GPU cuando está disponible. El uso de la GPU es preferible debido a su capacidad para manejar operaciones en paralelo, lo que acelera significativamente el entrenamiento del modelo. Si una GPU no está disponible, el modelo se ejecuta en la CPU. Esta flexibilidad en la selección del dispositivo de ejecución asegura que el código sea compatible y funcional en cualquier entorno, ya sea de alto rendimiento con GPU o más limitado con CPU.

Para determinar el dispositivo de ejecución, se verifica la disponibilidad de una GPU. Si se detecta una GPU, se utiliza para el entrenamiento; de lo contrario, se emplea la CPU. Esta configuración permite aprovechar al máximo los recursos computacionales disponibles,

mejorando la eficiencia del proceso de entrenamiento y reduciendo los tiempos de procesamiento.

5.4.3. Configuración para el entrenamiento

Para el entrenamiento del modelo es esencial una configuración específica que comprende elementos que determinan cómo el modelo aprende de los datos disponibles y optimiza su rendimiento. Cada aspecto de esta configuración desempeña un papel en el proceso de entrenamiento, influenciando directamente la capacidad del modelo para realizar predicciones precisas.

- Función de pérdida (‘criterion’)

  La función de pérdida `nn.CrossEntropyLoss()` se utiliza para calcular la discrepancia entre las predicciones del modelo y las etiquetas reales durante el entrenamiento. Aunque comúnmente se asocia con problemas de clasificación multiclase, donde hay más de dos clases posibles, también es aplicable a problemas de clasificación binaria. En este caso específico, `nn.CrossEntropyLoss()` gestiona internamente la función softmax y la pérdida de entropía cruzada, ayudando al modelo a optimizar sus predicciones entre dos clases: imágenes normales y anormales.

- Optimizador (‘optimizer’)

  El optimizador `torch.optim.SGD`, configurado con una tasa de aprendizaje inicial de `lr=0.001`, determina cómo se ajustan los pesos del modelo en respuesta a la pérdida calculada. Utilizando el gradiente descendente estocástico, este optimizador realiza actualizaciones de los parámetros del modelo en dirección al gradiente negativo multiplicado por la tasa de aprendizaje, optimizando así la precisión del modelo durante el entrenamiento.

- Scheduler de tasa de aprendizaje (‘exp_lr_scheduler’)

  El scheduler de tasa de aprendizaje `lr_scheduler.StepLR` desempeña un papel crucial al ajustar dinámicamente la tasa de aprendizaje a lo largo del entrenamiento. Aunque típicamente se emplea en problemas multiclase, su aplicación en problemas binarios como el presente puede mejorar la estabilidad del entrenamiento y facilitar

la convergencia del modelo hacia soluciones óptimas. En este caso, el scheduler reduce la tasa de aprendizaje por un factor de `gamma=0.1` cada `step_size=7` epochs.

5.4.4. Proceso de entrenamiento del modelo

En este apartado se describe el proceso de entrenamiento del modelo de clasificación binaria, optimizado para diferenciar entre imágenes normales y anormales. El entrenamiento se realiza utilizando PyTorch y se implementa mediante la función train_model.

La función train_model realiza el entrenamiento y la evaluación del modelo a través de múltiples epochs, ajustando los parámetros del modelo para minimizar la pérdida y maximizar la precisión. A continuación, se resumen los componentes del proceso de entrenamiento:

1. Configuración inicial
   - El modelo y los datos se trasladan al dispositivo adecuado (GPU o CPU).
   - Se crea un directorio temporal para almacenar los checkpoints del modelo, permitiendo guardar y recuperar el mejor modelo basado en su rendimiento durante la evaluación.

2. Iteración por Epochs
   - El entrenamiento se realiza a lo largo de un número definido de epochs (`num_epochs`), utilizándose en este caso 25. Cada epoch incluye una fase de entrenamiento y una fase de evaluación (validación).

3. Fase de entrenamiento
   - En la fase de entrenamiento, el modelo se ajusta mediante la función de pérdida `nn.CrossEntropyLoss()`, que calcula la discrepancia entre las predicciones del modelo y las etiquetas reales.
   - El optimizador `torch.optim.SGD` actualiza los pesos del modelo en función del gradiente descendente estocástico.
   - Se utiliza un scheduler (`lr_scheduler.StepLR`) para ajustar dinámicamente la tasa de aprendizaje, mejorando la estabilidad y la eficiencia del entrenamiento.

4. Fase de evaluación
    - En la fase de evaluación, se calcula la precisión del modelo utilizando un conjunto de datos de prueba. Si la precisión supera la mejor precisión registrada, se guarda el estado actual del modelo.

5. Optimización y guardado del mejor modelo
    - A lo largo de las epochs, se supervisan y registran la pérdida y la precisión tanto en las fases de entrenamiento como de evaluación.
    - Al final del entrenamiento, el modelo con la mejor precisión en la fase de evaluación se guarda y se restaura para su uso futuro.

El proceso completo asegura que el modelo pueda generalizar correctamente y hacer predicciones precisas sobre nuevas imágenes, optimizando su rendimiento en la clasificación binaria de imágenes radiográficas.

5.4.5. Visualización de resultados del modelo

La función `visualize_model` se utiliza para inspeccionar visualmente las predicciones del modelo en el conjunto de datos de prueba. Este proceso es crucial para evaluar de manera cualitativa cómo el modelo está funcionando y para identificar posibles áreas de mejora. A continuación, se describe el proceso y los componentes clave de la función de visualización:

1. Configuración inicial
    - El modelo se configura para el modo de evaluación utilizando `model.eval()`, y se asegura que esté en el dispositivo adecuado (GPU o CPU).
    - Se inicializa una figura para la visualización utilizando matplotlib.

2. Iteración sobre el conjunto de datos de prueba
    - Se itera sobre los datos de prueba (`test_dataloader`) sin calcular gradientes (`torch.no_grad()`), lo cual es más eficiente y adecuado para la inferencia.
    - Para cada lote de imágenes, se obtienen las predicciones del modelo y se calculan las probabilidades utilizando la función `softmax`.

3. Visualización de predicciones
    - Para cada imagen en el lote, se muestran las predicciones del modelo junto con la probabilidad asociada.
    - La imagen y su predicción se visualizan en una cuadrícula mediante matplotlib, con el título de cada subplot indicando la clase predicha y la probabilidad correspondiente (véase Figura 29).

4. Restauración del estado del modelo
    - Después de la visualización, el modelo se restaura a su modo original de entrenamiento o evaluación (`model.train(mode=was_training)`), asegurando que este proceso de visualización no afecte su estado para futuras operaciones.

La visualización de las predicciones del modelo proporciona una forma intuitiva de evaluar su rendimiento y ayuda a identificar patrones o errores que pueden no ser evidentes a partir de métricas cuantitativas.

Figura 29. Ejemplo de dos imágenes con su predicción de clasificación y correspondiente probabilidad.

6. Resultados

El código de este apartado se puede observar en el Anexo 11.5.

6.1. Evaluación del modelo

6.1.1. Métricas de evaluación

Para evaluar el rendimiento de este modelo de clasificación binaria de imágenes radiográficas, se utilizan diversas métricas que proporcionan una visión integral de su eficacia. Las métricas más relevantes incluyen la precisión (precisión), la sensibilidad (recall), la exactitud (accuracy), y el área bajo la curva (AUC) de la característica operativa del receptor (ROC). Estas métricas ayudan a entender tanto la capacidad del modelo para realizar predicciones correctas como su comportamiento ante falsos positivos y negativos.

- Precisión (Precision)

  Indica cuán precisa es la predicción del modelo cuando dice que una imagen es anormal. Es la proporción de verdaderos positivos entre todos los ejemplos que el modelo clasificó como positivos.

$$Precision = \frac{TP}{TP + FP}$$

Ecuación 5. Precisión

- Sensibilidad (Recall)

  Mide la capacidad del modelo para identificar correctamente las imágenes anormales. Es la proporción de verdaderos positivos entre todos los ejemplos realmente positivos.

$$Recall = \frac{TP}{TP + FN}$$

Ecuación 6. Sensibilidad

- Exactitud (Accuracy)

  Proporciona una visión general del rendimiento del modelo en todas las clases. Es la proporción de todas las predicciones correctas (tanto verdaderos positivos como verdaderos negativos) entre el total de ejemplos.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Ecuación 7. Exactitud

- Área Bajo la Curva (AUC) de la Curva ROC

  Un AUC cercano a 1 indica un modelo excelente, mientras que un AUC de 0.5 indica un rendimiento no mejor que el azar. Es una medida de la capacidad del modelo para distinguir entre clases. Se calcula a partir de la curva ROC, que traza la tasa de verdaderos positivos (TPR) frente a la tasa de falsos positivos (FPR).

6.2. Resultados obtenidos

En esta sección, se presentan los resultados obtenidos tras la evaluación del modelo de clasificación binaria de imágenes radiográficas de la muñeca. Se incluyen las métricas de rendimiento mencionadas anteriormente y la visualización de la curva ROC. Estos resultados proporcionan una visión detallada de la capacidad del modelo para clasificar imágenes radiográficas de la muñeca en normales y anormales.

Los resultados obtenidos para el modelo de clasificación binaria en el conjunto de datos de prueba son los siguientes:

- True Positive (TP): 207

- False Positive (FP): 38
- True Negative (TN): 326
- False Negative (FN): 88

Estos valores representan el rendimiento del modelo en términos de verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos, respectivamente.

La matriz de confusión (véase Figura 18) muestra la distribución de las predicciones correctas e incorrectas del modelo. Esta matriz es una herramienta esencial para evaluar el rendimiento de un modelo de clasificación, ya que permite visualizar no solo las tasas de aciertos sino también los errores cometidos por el modelo en cada clase.

En la matriz de confusión, los valores en la diagonal principal representan las predicciones correctas:

- Clase Positiva (True Positives): El valor de 0,9 en la celda correspondiente a la clase positiva en ambas dimensiones (actual y predicha) indica que el 90% de las imágenes que son realmente positivas (anormales) fueron correctamente clasificadas por el modelo como positivas.

- Clase Negativa (True Negatives): El valor de 0,7 en la celda correspondiente a la clase negativa en ambas dimensiones indica que el 70% de las imágenes que son realmente negativas (normales) fueron correctamente clasificadas por el modelo como negativas.

Los valores fuera de la diagonal principal representan las predicciones incorrectas:

- Falsos Positivos (False Positives): El valor de 0,1 en la celda donde las predicciones son positivas pero las etiquetas reales son negativas indica que el 10% de las imágenes que son realmente negativas fueron incorrectamente clasificadas como positivas por el modelo. Esto implica que el modelo cometió errores al identificar imágenes normales como anormales.

- Falsos Negativos (False Negatives): El valor de 0,3 en la celda donde las predicciones son negativas pero las etiquetas reales son positivas indica que el

30% de las imágenes que son realmente positivas fueron incorrectamente clasificadas como negativas por el modelo.

La matriz de confusión es especialmente útil porque proporciona una visión detallada de dónde y cómo el modelo está cometiendo errores. En este caso, observamos que el modelo tiene una alta tasa de aciertos para la clase positiva (90%) y una buena pero menor tasa de aciertos para la clase negativa (70%).

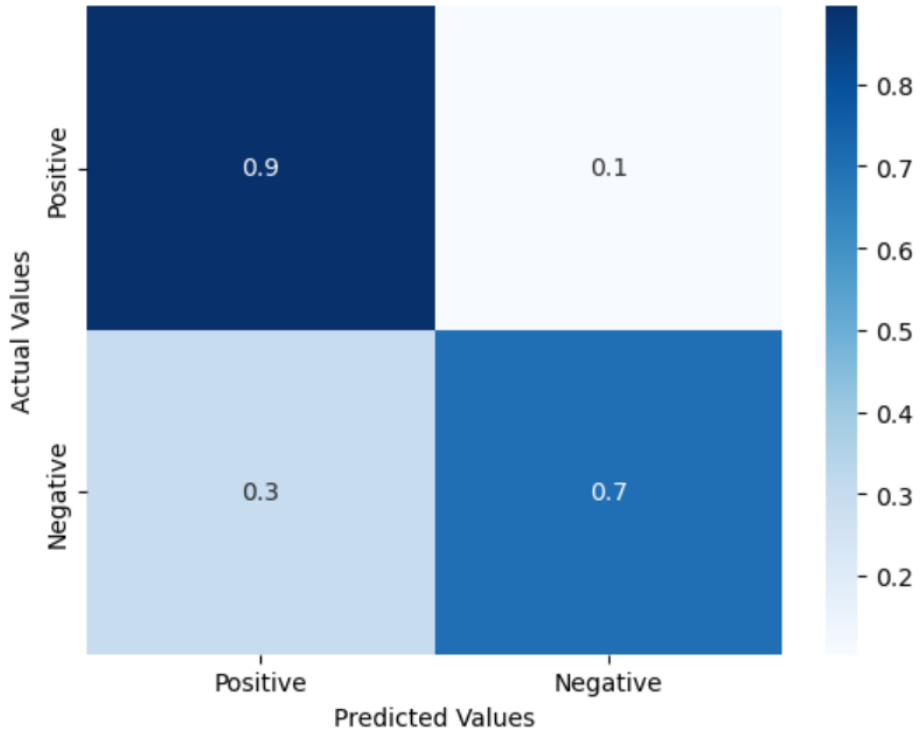


Figura 30. Matriz de confusión del modelo de la muñeca

RESULTADOS OBTENIDOS	
Precisión	0,8449
Sensibilidad	0,7017
Exactitud	0,8088
AUC	0,8741

Tabla 2. Resultados de las métricas de evaluación para el modelo de clasificación binaria de imágenes radiográficas de la muñeca.

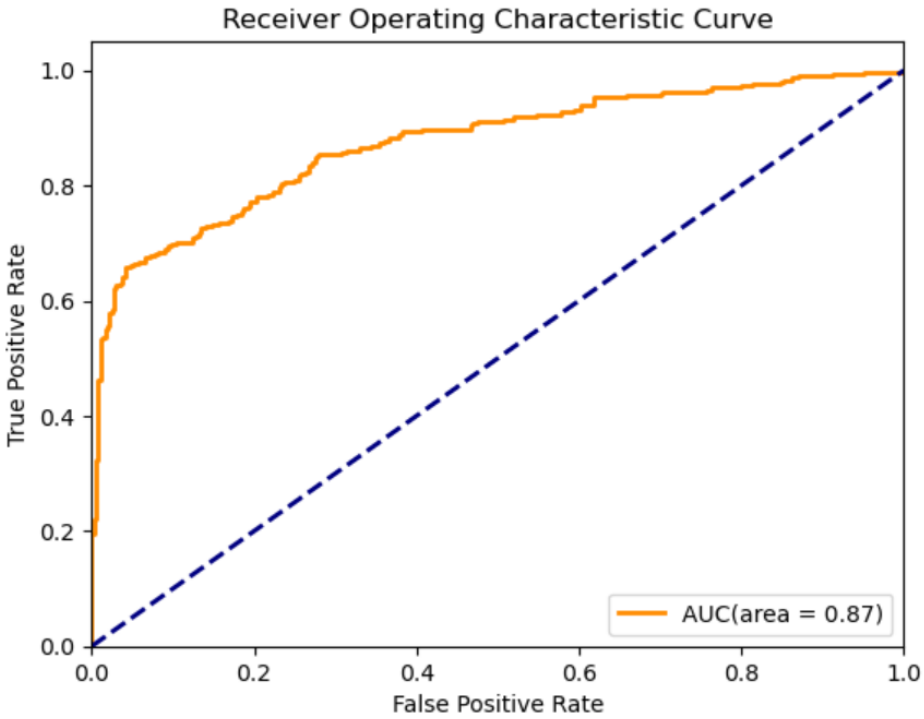


Figura 31. Curva ROC del modelo de la muñeca

Los resultados de la evaluación del modelo indican que el modelo tiene una alta precisión, sensibilidad y exactitud, lo que sugiere que es eficaz para la tarea de clasificación binaria de imágenes radiográficas de la muñeca. La precisión (0,8449) refleja que el 84,49% de las veces en que el modelo predijo una imagen como anormal, esta predicción fue correcta. Esto es crucial en un contexto clínico, ya que un alto valor de precisión minimiza el número de falsos positivos, reduciendo la posibilidad de diagnósticos erróneos que podrían llevar a procedimientos innecesarios.

La sensibilidad (0,7017), o recall, indica que el modelo identificó correctamente el 70,17% de las imágenes realmente anormales. Una alta sensibilidad es vital en aplicaciones médicas, ya que asegura que la mayoría de los casos anormales sean detectados, lo cual es fundamental para un diagnóstico temprano y un tratamiento oportuno.

La exactitud (0,8088) refleja que el 80,88% de todas las predicciones del modelo, tanto para imágenes normales como anormales, fueron correctas. Esto da una visión general del rendimiento del modelo en el conjunto de datos y sugiere que es fiable para clasificar imágenes radiográficas de la muñeca.

El área bajo la curva (AUC) de la curva ROC, con un valor de 0,87, sugiere que el modelo tiene una excelente capacidad para diferenciar entre imágenes normales y anormales. El AUC es una métrica que resume la capacidad del modelo para discriminar entre las dos clases en un rango de umbrales de decisión. Un AUC cercano a 1 indica un modelo con una muy buena capacidad de clasificación. En este caso, un AUC de 0,87 es indicativo de un rendimiento robusto, mostrando que el modelo es eficaz en identificar correctamente las clases en una variedad de umbrales.

En conclusión, el modelo tiene un rendimiento prometedor, muy sensible y exactitud, y una excelente AUC, lo que sugiere que es eficaz para clasificar imágenes radiográficas de la muñeca.

6.3. Comparación con estudios previos

En este apartado, se realiza una comparación detallada entre los resultados obtenidos en este proyecto y los resultados reportados en el estudio previo, específicamente el artículo oficial de la base de datos MURA. Es crucial destacar que mientras el estudio previo abarca diversos tipos de estudios radiográficos, mi investigación se centra exclusivamente en imágenes de la muñeca.

Los resultados de evaluación del modelo de este proyecto indican que, aunque el modelo tiene un buen rendimiento, los resultados son inferiores a los reportados en un estudio previo, el artículo oficial de la base de datos MURA. Mi modelo mostró una sensibilidad de 0,702, y un AUC de 0,87. En comparación, el estudio previo reportó una sensibilidad de 0,815 y un AUC de 0,929. La comparación muestra que mi modelo tiene una menor sensibilidad y un AUC más bajo que los reportados en el estudio oficial, por ello es crucial analizar las diferencias metodológicas que pueden haber contribuido a estos resultados.

Primero, es importante destacar que este trabajo se centró exclusivamente en imágenes individuales, ya que se pretendía observar cuánto de efectivo podía llegar a ser comparado con el estudio previo, que evaluó su modelo utilizando estudios completos, que incluyen varias imágenes de diferentes vistas de la misma parte del cuerpo. Trabajar con imágenes individuales enfrenta a este modelo a una mayor variabilidad y menos contexto para hacer predicciones precisas en comparación con estudios que contienen múltiples vistas de la misma área, proporcionando información complementaria. Esto permite que el modelo del estudio previo tenga más información contextual y correlacional para hacer predicciones, lo

cual puede mejorar significativamente la sensibilidad y la capacidad de discriminación (AUC).

Segundo, este estudio se limitó solo a imágenes de la muñeca, mientras que el estudio oficial abarcó diversos tipos de estudios radiográficos de la extremidad superior del cuerpo. Esta diferencia en el alcance de los datos también puede influir en los resultados.

La sensibilidad de 0,7017 en este modelo es inferior a la sensibilidad de 0.815 del estudio oficial, lo cual podría atribuirse a la falta de información adicional de diferentes vistas que el estudio oficial consideró. Un AUC de 0,87 en este modelo, comparado con 0.929 del estudio oficial, indica una menor capacidad de discriminación entre clases, sugiriendo que el modelo del estudio oficial se benefició de una mayor variedad de datos para ajustar mejor los umbrales de decisión.

En esta investigación, se quería evaluar cuán competitivo puede ser un modelo utilizando solo una imagen en comparación con el enfoque de estudios completos del artículo oficial. A pesar de las limitaciones de trabajar con imágenes individuales, los resultados obtenidos aún demuestran un rendimiento significativo, lo que sugiere que incluso con menos información, el modelo puede ser una herramienta útil en la clasificación de imágenes radiográficas.

En conclusión, la diferencia en las métricas de rendimiento entre este modelo y el del estudio previo puede atribuirse principalmente a dos factores: el uso de imágenes individuales versus estudios completos, y la especialización en imágenes de la muñeca en mi investigación versus un enfoque más amplio que incluye diversas partes de la extremidad superior del cuerpo en el estudio oficial. La metodología del estudio oficial proporcionó una ventaja significativa en términos de variedad y cantidad de datos disponibles para la clasificación, lo cual influyó en los resultados superiores reportados en el artículo de la base de datos MURA.

7. Conclusiones

Para llevar a cabo este proyecto, se han seguido los objetivos establecidos en el apartado 1.2. de este documento.

Primero, se hizo una búsqueda y selección de bases de datos públicas con imágenes radiográficas, asegurando su idoneidad para entrenar algoritmos de inteligencia artificial. Tras esta búsqueda, se seleccionó la base de datos MURA (Musculoskeletal Radiographs) como la más adecuada para los objetivos del proyecto. Este proceso fue fundamental para establecer una base sólida para los siguientes pasos del proyecto.

A continuación, se procedió con el entrenamiento de las redes neuronales para la clasificación de las imágenes radiográficas. Este proceso incluyó la preparación de los datos, la selección de la arquitectura de red DenseNet y la optimización de los hiperparámetros para mejorar la precisión y la eficiencia del modelo. La clasificación fue binaria, enfocándose en distinguir entre imágenes normales y anormales, utilizando específicamente radiografías de muñeca.

Por último, se evaluaron los resultados obtenidos mediante métricas de rendimiento estándar en clasificación de imágenes.

Después de abordar y cumplir con estos objetivos, se puede concluir que la inteligencia artificial es una herramienta alentadora para asistir a los médicos de atención primaria en sus diagnósticos, especialmente en situaciones donde los conocimientos especializados pueden ser limitados. Los resultados obtenidos hasta ahora son prometedores, mostrando que la IA puede ayudar a identificar de manera rápida y precisa condiciones anormales en radiografías.

Esta capacidad es crucial para agilizar el proceso diagnóstico y facilitar intervenciones médicas tempranas, mejorando así la gestión y tratamiento de los pacientes.

Sin embargo, es evidente que aún queda trabajo por hacer para perfeccionar el modelo y obtener resultados mejores. Es fundamental continuar mejorando la precisión y confiabilidad del sistema, asegurando también su aplicación efectiva en una variedad de casos clínicos y condiciones médicas.

En resumen, la integración de la inteligencia artificial en la clasificación de imágenes radiográficas puede transformar la forma en que se realizan los diagnósticos en la atención primaria, con el objetivo final de mejorar la calidad de vida de los pacientes y optimizar los recursos en los servicios de salud.

8. Líneas futuras

Teniendo en cuenta las conclusiones extraídas de este proyecto, se proponen algunas mejoras para el futuro:

- Ampliación del estudio a toda la extremidad superior: Realizar el mismo estudio que se ha hecho para la muñeca, pero extendiéndolo a todas las partes de la misma contenidas en la base de datos MURA, incluyendo el codo, el hombro, el húmero, el antebrazo, la mano y los dedos. Esto permitirá evaluar el rendimiento del modelo en diferentes regiones anatómicas y proporcionar un análisis más completo.

- Uso de una base de datos diferente: Utilizar una base de datos distinta a MURA que incluya imágenes de otras partes del sistema musculoesquelético, no solo de la extremidad superior. Este enfoque permitirá generalizar el clasificador a una variedad más amplia de imágenes médicas, mejorando su aplicabilidad en contextos clínicos más diversos.

- Desarrollo de una aplicación independiente: Implementar el modelo en una aplicación independiente, en lugar de utilizar notebooks de Jupyter. Esta aplicación facilitará la integración del clasificador en los sistemas informáticos de atención primaria, permitiendo a los profesionales de la salud acceder y utilizar la herramienta de manera más eficiente y práctica en su rutina diaria.

Estas mejoras tienen el potencial de incrementar la eficacia y la aplicabilidad del modelo, extendiendo su utilidad en el diagnóstico y tratamiento en entornos clínicos variados.

9. Presupuesto

Para desarrollar este apartado del proyecto, se seguirán las directrices establecidas por el Colegio Oficial de Ingenieros Técnicos de Telecomunicaciones (COITT), y el precio final obtenido estará sujeto a la aprobación y aceptación del cliente. El presupuesto se presentará desglosado en varios apartados.

- Trabajo tarifado por tiempo empleado.
- Amortización del inmovilizado material.
- Redacción de la documentación.
- Derechos de visado del Colegio Oficial de Ingenieros Técnicos de Telecomunicación (COITT).
- Gastos de tramitación y envío.
- Aplicación de impuestos y coste total.

9.1. Trabajo tarifado por tiempo empleado

Este apartado del presupuesto se enfoca en contabilizar los gastos asociados a la mano de obra, basándose en el salario correspondiente a las horas trabajadas por un Ingeniero Técnico de Telecomunicaciones. Para calcular estos honorarios, y considerando que el proyecto se realizó bajo la tutela de la Universidad de Las Palmas de Gran Canaria, se ha consultado el documento "Clasificación y retribución del personal contratado con cargo a proyectos, programas, convenios y contratos" [37], específicamente la modificación de 2023. Según este documento, el salario mensual del personal técnico con una titulación mínima de Grado es de 745,15€ con una dedicación de 20 horas semanales, lo que equivale a un total de 9,31€.

Dado que la asignatura de Trabajo Fin de Grado (TFG) tiene una duración total de 300 horas, la tarifa por el tiempo empleado se calculará utilizando la siguiente ecuación.

$$Tarifa\ por\ tiempo\ empleado = 9{,}31\ €/h \times 300h = 2.793\ €$$

Ecuación 8. Honorarios por tiempo empleado

9.2. Amortización del inmovilizado material

En este apartado se realizan los cálculos de amortización de los recursos de hardware y software utilizados para la realización de este proyecto.

Para hacer este cálculo, se usa un sistema de amortización lineal, donde se asume que el inmovilizado material se desprecia de forma constante a lo largo de su vida útil. De esto sale la ecuación que se usa para calcular el coste de amortización.

$$Coste\ de\ amortización = \frac{Valor\ de\ adquisición - Valor\ residual}{Años\ de\ vida\ útil}$$

Ecuación 9. Coste de amortización

En este caso el valor residual es el valor que tendrá un bien después de su vida útil. Para el cálculo de este parámetro, se hace uso de la tabla de coeficientes de amortización lineal de la Agencia Tributaria (AEAT) [22], de la cual se ha extraído la siguiente información.

Tipo de elemento	Coeficiente lineal máximo	Periodo máximo
Equipo para procesos de información	25%	8 años
Sistemas y programas informáticos	33%	6 años

Tabla 3. Extracción de la tabla de coeficientes de amortización lineal de la Agencia Tributaria (AEAT)

9.2.1. Amortización del material hardware

La tabla 4 muestra la amortización del material hardware.

Recurso Hardware	Valor de adquisición	Valor residual	Vida útil (años)	Coste de amortización
Ordenador portátil Asus Zenbook UX425EA	1.198,84 €	299,71 €	5	179,83 €
Ordenador DELL Precision 5820	5.100 €	1.275 €	7	546,43 €
TOTAL				**726,26 €**

Tabla 4. Coste de amortización del hardware

9.2.2. Amortización del material software

La tabla siguiente muestra la amortización del material software.

Recurso Software	Coste de amortización
Pytorch	0,00 €
Jupyter Notebooks	0,00 €
Anaconda	0,00 €
GitHub	0,00 €
Microsoft Office 365	0,00 €
TOTAL	**0,00 €**

Tabla 5. Coste de amortización del software

Todas las herramientas de software utilizadas son gratuitas, excepto Microsoft Office 365, cuya licencia es proporcionada gratuitamente por la Universidad de Las Palmas de Gran Canaria (ULPGC).

9.3. Costes asociados a la redacción del documento

Para determinar el coste asociado a la redacción del proyecto se empleará la siguiente ecuación.

$$R = 0{,}07 \times P \times C_n$$

Ecuación 10. Coste por redacción del proyecto

Donde:

- R: Honorarios por la redacción del documento.
- P: Presupuesto del proyecto.
- C_n: Coeficiente de ponderación en función del presupuesto.

Para calcular P, se suma el trabajo tarifado por el tiempo empleado y la amortización total del inmovilizado material (veáse Ecuación 11).

$$P = C_{hardware} + C_{software} + Tarifa_{tiempo\_empleado}$$

Ecuación 11. Honorarios por la redacción del proyecto

Presupuesto del proyecto (P)	**Coste**
Amortización trabajo tarifado por el tiempo empleado	2.793 €
Amortización del material hardware	726,26 €
Amortización del material software	0,00 €
TOTAL	**3.519,26 €**

Tabla 6. Cálculo de P derivada de la ecuación de coste por redacción

Por otro lado, para el cálculo de C_n se sigue lo estipulado por el COITT. Según estas directrices, para presupuestos inferiores a 30.050,00 € se tiene un $C_n = 1$. Con estos datos se obtiene lo siguiente.

$$R = 0{,}07 \times 3.519{,}26\ € \times 1 = 246{,}35\ €$$

Ecuación 12. Resultado de los honorarios por la redacción del proyecto

El coste calculado hasta este punto está completamente libre de impuestos.

9.4. Derechos de visado del COITT

Los gastos de visado del COITT se calculan mediante la siguiente ecuación.

$$V = 0{,}006 \times P_1 \times C_1 + 0{,}003 \times P_2 \times C_2$$

Ecuación 13. Cálculo de los gastos de visado del COITT

Donde:

- V: Costes del visado del trabajo.
- P_1: Presupuesto del proyecto.
- C_1: Coeficiente reductor dado en función del presupuesto de trabajo.
- P_2: Presupuesto de ejecución material correspondiente a la obra civil.
- C_2: Coeficiente reductor dado en función de P_2.

El valor de P_1 será el previamente calculado P. Se asigna nuevamente el valor de $C_1 = 1$, conforme a la premisa establecida para proyectos con presupuestos inferiores a 30.050,00 €. Finalmente, P_2 tiene un valor de 0,00 € dado que no se lleva a cabo ninguna obra. Con estos datos, se obtiene el siguiente resultado.

$$V = 0{,}006 \times 3.519{,}26\ € \times 1 + 0{,}003 \times 0{,}00\ € \times 1 = 21{,}12\ €$$

Ecuación 14. Resultado del cálculo de los gastos de visado del COITT

9.5. Gastos de tramitación y envío

Cada documento enviado de forma telemática conlleva un cargo por tramitación de seis euros (6,00€).

9.6. Aplicación de impuestos

Los costes totales de este proyecto incluyen el Impuesto General Indirecto Canario (IGIC), que actualmente se aplica a una tasa del siete por ciento (7%). La tabla siguiente presenta el presupuesto final con la inclusión del impuesto correspondiente.

Concepto	Coste
Trabajo tarifado por tiempo empleado	2.793 €
Amortización del material hardware	726,26 €
Amortización del material software	0,00 €
Costes asociados a la redacción del documento	246,35 €
Derechos de visado del COITT	21,12 €
Gastos de tramitación y envío	6,00 €
SUBTOTAL	**3.792,73 €**
IGIC (7%)	**265,49 €**
TOTAL	**4.058,22 €**

Tabla 7. Coste total del proyecto con los impuestos correspondientes aplicados

El presupuesto total del proyecto "Clasificación de imágenes radiográficas utilizando inteligencia artificial" asciende a cuatro mil cincuenta y ocho euros con veintidós céntimos (4.058,22 €).

Fdo.: Paula Rodríguez Álvarez

En Las Palmas de Gran Canaria a 20 de julio de 2024

10. Bibliografía

[1] Mayo Clinic, "Radiografía: Estudio de diagnóstico por imágenes que ayuda a diagnosticar con rapidez - Mayo Clinic," Mayo Clinic. Accessed: Feb. 06, 2024. [Online]. Available: https://www.mayoclinic.org/es/tests-procedures/x-ray/about/pac-20395303

[2] NUBIX, "Uso de la Inteligencia Artificial para mejorar la precisión en la detección de enfermedades en radiografías," NUBIX. Accessed: Feb. 06, 2024. [Online]. Available: https://nubix.cloud/radiologia/como-se-esta-utilizando-la-inteligencia-artificial-para-mejorar-la-precision-en-la-deteccion-de-enfermedades-en-radiografias

[3] Dr. Fabián Vítolo; Dra. Elisa Gancedo, "ERRORES DIAGNÓSTICOS EN RADIOLOGÍA," *Biblioteca Virtual NOBLE*, 2009.

[4] "Cómo se obtienen las imágenes radiográficas." Accessed: Jun. 12, 2024. [Online]. Available: https://www.cursosfemxa.es/blog/imagenes-radiograficas-rayos-x

[5] L. Ford, "Issue 2 1 Perspective Citation: Ford L, The importance and applications of radiography in modern medicine," *J Biomed Imag Bioeng*, vol. 7, no. 2, p. 174, 2023, doi: 10.35841/aabib-7.2.174.

[6] "¿Qué es el Sistema Músculo-Esquelético? Diccionario Médico. Clínica U. Navarra." Accessed: May 13, 2024. [Online]. Available: https://www.cun.es/diccionario-medico/terminos/sistema-musculo-esqueletico

[7] "Trastornos musculoesqueléticos." Accessed: Jun. 17, 2024. [Online]. Available: https://www.who.int/es/news-room/fact-sheets/detail/musculoskeletal-conditions

[8] “Pruebas para el diagnóstico de trastornos musculoesqueléticos - Trastornos de los huesos, articulaciones y músculos - Manual MSD versión para público general.” Accessed: Jun. 17, 2024. [Online]. Available: https://www.msdmanuals.com/es-es/hogar/trastornos-de-los-huesos,-articulaciones-y-m%C3%BAsculos/diagn%C3%B3stico-de-los-trastornos-musculoesquel%C3%A9ticos/pruebas-para-el-diagn%C3%B3stico-de-trastornos-musculoesquel%C3%A9ticos

[9] “Radiografía músculo-esquelética | Aula de Pacientes.” Accessed: Jun. 12, 2024. [Online]. Available: https://www.saludcastillayleon.es/AulaPacientes/es/cuida-salud-16ad6f/pruebas-diagnosticas/pruebas-diagnostico-imagen/pruebas/radiologia-convencional/radiografia-musculo-esqueletica

[10] Dª María Francisca Cegarra Navarro, “Radiología Músculo-Esquelética en Atención Primaria y Especializada: Resultados de una Intervención,” UNIVERSIDAD DE MURCIA, 2017.

[11] “Introducción a las Redes Neuronales: Conceptos Básicos y Aplicaciones.” Accessed: Jun. 20, 2024. [Online]. Available: https://data-universe.org/introduccion-a-las-redes-neuronales-conceptos-basicos-y-aplicaciones/

[12] Edgar Serna M., *DESARROLLO E INNOVACIÓN EN INGENIERÍA*. Medellín, Antioquia: Editorial IAI, 2017.

[13] “¿Qué es PyTorch? | IBM.” Accessed: Jul. 08, 2024. [Online]. Available: https://www.ibm.com/es-es/topics/pytorch

[14] “Descubre Jupyter Notebook: imprescindible para ciencia de datos.” Accessed: Jul. 09, 2024. [Online]. Available: https://aprenderbigdata.com/jupyter-notebook/

[15] “Qué es Github: para qué sirve, cómo funciona y cómo usarlo.” Accessed: Jul. 09, 2024. [Online]. Available: https://ebac.mx/blog/que-es-github

[16] “Stanford University.” Accessed: Jul. 18, 2024. [Online]. Available: https://www.stanford.edu/

[17] P. Rajpurkar *et al.*, “MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs”, Accessed: Jun. 25, 2024. [Online]. Available: http://stanfordmlgroup.github.io/competitions/mura.

[18] “Densenet | PyTorch.” Accessed: Jul. 07, 2024. [Online]. Available: https://pytorch.org/hub/pytorch_vision_densenet/

[19] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, “Densely Connected Convolutional Networks”

[20] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition”

[21] K. Simonyan and A. Zisserman, “VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION,” 2015

[22] “Agencia Tributaria: 1. Por coeficientes de amortización lineal.” Accessed: Jul. 17, 2024. [Online]. Available: https://sede.agenciatributaria.gob.es/Sede/ayuda/manuales-videos-folletos/manuales-practicos/irpf-2020/capitulo-7-rendimientos-actividades-economicas-directa/fase-1-determinacion-rendimiento-neto/amortizaciones-dotaciones-ejercicio-fiscalmente-deducibles/requisitos-generales/coeficientes-amortizacion-lineal.html

11. Anexos

11.1. Análisis exploratorio de los datos

```
def contar_pacientes(archivo):
    pacientes = set()

    # Leer el archivo CSV
    with open(archivo, mode='r') as file:
        csv_reader = csv.reader(file)
        for row in csv_reader:
            # Extraer la ruta de la fila
            ruta = row[0]

            # Extraer el identificador del paciente de la ruta
            partes = ruta.split('/')
            for parte in partes:
                if parte.startswith("patient"):
                    pacientes.add(parte)
                    break

    # Contar el número de pacientes únicos
    num_pacientes = len(pacientes)
    return num_pacientes

# Llamada a la función y mostrar el resultado
numero_de_pacientes_train = contar_pacientes(train_labeled_studies_fn)
print(f'Cantidad de pacientes en conjunto de entrenamiento:
{numero_de_pacientes_train}')

numero_de_pacientes_valid = contar_pacientes(valid_labeled_studies_fn)
print(f'Cantidad de pacientes en conjunto de validación:
{numero_de_pacientes_valid}')
```

Código 1. Cantidad de pacientes

```
def contar_imagenes_desde_csv(archivo_csv):
    cantidad_imagenes = 0

    # Leer el archivo CSV
    with open(archivo_csv, mode='r') as file:
        csv_reader = csv.reader(file)
        for row in csv_reader:
            # Extraer la ruta de la imagen de la fila
            ruta_imagen = row[0]

            # Verificar si la ruta termina con una extensión de imagen
válida
            if ruta_imagen.endswith('.png') or
ruta_imagen.endswith('.jpg') or ruta_imagen.endswith('.jpeg'):
                cantidad_imagenes += 1

    return cantidad_imagenes




# Llamada a la función para contar imágenes
numero_de_imagenes_train = contar_imagenes_desde_csv(train_img_paths_fn)
print(f'Cantidad de imágenes en el conjunto de entrenamiento:
{numero_de_imagenes_train}')

numero_de_imagenes_valid = contar_imagenes_desde_csv(valid_img_paths_fn)
print(f'Cantidad de imágenes en el conjunto de validación:
{numero_de_imagenes_valid}')
```

Código 2. Cantidad de imágenes

```
# Inicializar contadores
train_positivos = 0
train_negativos = 0

# Leer el archivo CSV
with open(train_labeled_studies_fn, mode='r') as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        # Extraer el número que indica si es positivo (1) o negativo (0)
        etiqueta = row[1]
        if etiqueta == '1':
           train_positivos += 1
        elif etiqueta == '0':
            train_negativos += 1

# Calcular totales y porcentajes
total_estudios = train_positivos + train_negativos
porcentaje_positivos = (train_positivos / total_estudios) * 100
porcentaje_negativos = (train_negativos / total_estudios) * 100

view = {
    'Estudios': {
        'Normales': train_negativos,
        'Anormales': train_positivos,
        'Porcentaje Normales': porcentaje_negativos,
        'Porcentaje Anormales': porcentaje_positivos
    }
}

# Mostrar los resultados
print(f'Cantidad de estudios anormales: {train_positivos}')
print(f'Cantidad de estudios normales: {train_negativos}')
print(f'Porcentaje de estudios anormales: {porcentaje_positivos:.2f}%')
print(f'Porcentaje de estudios normales: {porcentaje_negativos:.2f}%')
```

Código 3. Cantidad de estudios train

```
# Inicializar contadores
valid_positivos = 0
valid_negativos = 0

# Leer el archivo CSV
with open(valid_labeled_studies_fn, mode='r') as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        # Extraer el número que indica si es positivo (1) o negativo (0)
        etiqueta = row[1]
        if etiqueta == '1':
            valid_positivos += 1
        elif etiqueta == '0':
            valid_negativos += 1
# Calcular totales y porcentajes
total_estudios = valid_positivos + valid_negativos
porcentaje_positivos = (valid_positivos / total_estudios) * 100
porcentaje_negativos = (valid_negativos / total_estudios) * 100

view = {
    'Estudios': {'Normales':valid_negativos, 'Anormales':
valid_positivos,
                'Porcentaje Normales': porcentaje_negativos,
        'Porcentaje Anormales': porcentaje_positivos}
}
# Mostrar los resultados
print(f'Cantidad de estudios anormales: {valid_positivos}')
print(f'Cantidad de estudios normales: {valid_negativos}')
print(f'Porcentaje de estudios anormales: {porcentaje_positivos:.2f}%')
print(f'Porcentaje de estudios normales: {porcentaje_negativos:.2f}%')
```

Código 4. Cantidad de estudios valid

```
# Datos de entrenamiento y validación
train_labels = ['Normales', 'Anormales']
train_counts = [train_negativos, train_positivos]

valid_labels = ['Normales', 'Anormales']
valid_counts = [valid_negativos, valid_positivos]

# Configuración de la figura y ejes
fig, ax = plt.subplots()

# Ancho de las barras
bar_width = 0.35

# Posiciones de los grupos de barras
x_pos_train = np.arange(len(train_labels))
x_pos_valid = [x + bar_width for x in x_pos_train]

# Graficar barras para entrenamiento y validación
bars_train = ax.bar(x_pos_train, train_counts, bar_width,
align='center', alpha=0.5, label='Train')
bars_valid = ax.bar(x_pos_valid, valid_counts, bar_width,
align='center', alpha=0.5, label='Valid')

# Etiquetas en los ejes x
ax.set_xticks(x_pos_train + bar_width / 2)
ax.set_xticklabels(train_labels)

# Etiquetas y título
ax.set_xlabel('Resultado')
ax.set_ylabel('Cantidad de Estudios')
ax.set_title('Cantidad de Estudios Normales y Anormales')

# Mostrar leyenda
ax.legend()
# Mostrar los valores de cada barra
def autolabel(bars):
    for bar in bars:
        yval = bar.get_height()
        ax.text(bar.get_x() + bar.get_width() / 2.0, yval, int(yval),
ha='center', va='bottom')

autolabel(bars_train)
autolabel(bars_valid)

# Mostrar la gráfica
plt.show()
```

Código 5. Gráfica estudios normales y anormales

```
# Inicializar un diccionario para contar los estudios por tipo y estado
(normal/anormal)
tipo_estudio_count = defaultdict(lambda: {'Normales': 0, 'Anormales':
0})
# Leer el archivo CSV
with open(train_labeled_studies_fn, mode='r') as file:
    csv_reader = csv.reader(file)
    for row in csv_reader:
        # Extraer la ruta completa
        ruta = row[0]
        # Asumiendo que el tipo de estudio es la tercera parte de la
ruta, separar por '/'
        partes_ruta = ruta.split('/')
        tipo_estudio = partes_ruta[2]  # Ajustar este índice según la
estructura real de tu ruta
        # Extraer la etiqueta que indica si es normal (0) o anormal (1)
        etiqueta = row[1]
        # Incrementar el contador correspondiente
        if etiqueta == '0':
            tipo_estudio_count[tipo_estudio]['Normales'] += 1
        elif etiqueta == '1':
            tipo_estudio_count[tipo_estudio]['Anormales'] += 1
# Mostrar los resultados por cada tipo de estudio
for tipo, counts in tipo_estudio_count.items():
    total = counts["Normales"] + counts["Anormales"]
    porcentaje_normales = (counts["Normales"] / total) * 100 if total >
0 else 0
    porcentaje_anormales = (counts["Anormales"] / total) * 100 if total
> 0 else 0
    print(f'Tipo de estudio: {tipo}')
    print(f'   Normales:
{counts["Normales"]}({porcentaje_normales:.2f}%)')
    print(f'   Anormales:
{counts["Anormales"]}({porcentaje_anormales:.2f}%)')
    print(f'   Total: {total}')
# También puedes acceder a los totales de normales y anormales en todo
el conjunto de datos
total_normales = sum(counts['Normales'] for counts in
tipo_estudio_count.values())
total_anormales = sum(counts['Anormales'] for counts in
tipo_estudio_count.values())
print(f'Total de estudios normales en todo el conjunto:
{total_normales}')
print(f'Total de estudios anormales en todo el conjunto:
{total_anormales}')
```

Código 6. Cantidad de Estudios Normales y Anormales por Tipo de Estudio (Train)

Para validación el código utilizado es idéntico

```
# Función para contar estudios normales y anormales por tipo de estudio
en train
def contar_estudios_normales_anormales_train(csv_filename):
    tipo_estudio_count = defaultdict(lambda: {'Normales': 0,
'Anormales': 0})

    # Leer el archivo CSV y contar estudios normales y anormales por
tipo de estudio en train
    with open(csv_filename, mode='r') as file:
        csv_reader = csv.reader(file)
        for row in csv_reader:
            ruta = row[0]
            etiqueta = row[1]

            partes_ruta = ruta.split('/')
            if len(partes_ruta) >= 3:  # Verificar que haya al menos
tres partes en la ruta
                tipo_estudio = partes_ruta[2]  # Ajustar este índice
según la estructura real de tu ruta
                if etiqueta == '0':
                    tipo_estudio_count[tipo_estudio]['Normales'] += 1
                elif etiqueta == '1':
                    tipo_estudio_count[tipo_estudio]['Anormales'] += 1
            else:
                print(f"Ruta incompleta: {ruta}")

    return tipo_estudio_count


# Obtener conteos de estudios normales y anormales por tipo de estudio
para train
train_tipo_estudio_count =
contar_estudios_normales_anormales_train(train_labeled_studies_fn)

# Preparar datos para la gráfica
tipos_estudio = list(train_tipo_estudio_count.keys())
normales_train = [train_tipo_estudio_count[tipo]['Normales'] for tipo in
tipos_estudio]
anormales_train = [train_tipo_estudio_count[tipo]['Anormales'] for tipo
in tipos_estudio]

# Configuración de la gráfica
plt.figure(figsize=(10, 5))  # Ajusta el tamaño de la figura según sea
necesario

# Ancho de las barras
bar_width = 0.35
```

```
# Posiciones de las barras para train
x_pos = np.arange(len(tipos_estudio))
x_pos_normales = x_pos - bar_width/2
x_pos_anormales = x_pos + bar_width/2

# Crear las barras para train
bars_train_normales = plt.bar(x_pos_normales, normales_train, bar_width,
align='center', alpha=0.5, label='Train Normales')
bars_train_anormales = plt.bar(x_pos_anormales, anormales_train,
bar_width, align='center', alpha=0.5, label='Train Anormales')

# Etiquetas de conteo dentro de las barras para train Normales
for bar, count in zip(bars_train_normales, normales_train):
    plt.text(bar.get_x() + bar.get_width()/2., bar.get_height(), count,
ha='center', va='bottom')

# Etiquetas de conteo dentro de las barras para train Anormales
for bar, count in zip(bars_train_anormales, anormales_train):
    plt.text(bar.get_x() + bar.get_width()/2., bar.get_height(), count,
ha='center', va='bottom')

# Configuración de la gráfica
plt.xlabel('Tipo de Estudio')
plt.ylabel('Cantidad de Estudios')
plt.title('Cantidad de Estudios Normales y Anormales por Tipo de Estudio
(Train)')
plt.xticks(x_pos, tipos_estudio)
plt.legend()  # Mostrar la leyenda con los colores
plt.tight_layout()
plt.show()
```

Código 7. Gráfica cantidad de estudios normales y anormales por tipo de estudio (Train)

Para validación el código utilizado es idéntico

```
# Ruta base donde se encuentran las imágenes
base_path = base_dir + 'MURA-v1.1/train/'

# Lista de todas las regiones disponibles
regiones = ['XR_ELBOW', 'XR_FINGER', 'XR_FOREARM', 'XR_HAND',
'XR_HUMERUS', 'XR_SHOULDER', 'XR_WRIST']

def buscar_pacientes_con_estudio(base_path, region, estudio):
    # Obtener la lista de pacientes en la región seleccionada
    pacientes_path = os.path.join(base_path, region)
    pacientes = os.listdir(pacientes_path)

    # Filtrar pacientes que tengan el estudio especificado (positive o
negative)
    pacientes_con_estudio = []
    for paciente in pacientes:
        estudio_path = os.path.join(base_path, region, paciente,
f'study1_{estudio}', 'image1.png')
        if os.path.exists(estudio_path):
            pacientes_con_estudio.append(paciente)

    return pacientes_con_estudio

def mostrar_imagenes_aleatorias(base_path, region):
    # Obtener dos pacientes aleatorios, uno con estudio positivo y otro
con estudio negativo
    pacientes_positivos = buscar_pacientes_con_estudio(base_path,
region, 'positive')
    pacientes_negativos = buscar_pacientes_con_estudio(base_path,
region, 'negative')

    if len(pacientes_positivos) < 1 or len(pacientes_negativos) < 1:
        print(f"No se encontraron pacientes con ambos estudios (positive
y negative) en la región {region}")
        return

    paciente_positivo = random.choice(pacientes_positivos)
    paciente_negativo = random.choice(pacientes_negativos)
```

```
 # Mostrar las imágenes de los pacientes seleccionados
    fig, axs = plt.subplots(1, 2, figsize=(8, 4))

    # Mostrar imagen del paciente con estudio positivo (Anormal)
    imagen_positiva_path = os.path.join(base_path, region,
paciente_positivo, 'study1_positive', 'image1.png')
    imagen_positiva = Image.open(imagen_positiva_path)
    axs[0].imshow(imagen_positiva, cmap='gray')
    axs[0].axis('off')
    axs[0].set_title(f'Región: {region}\nEstudio: Anormal', fontsize=12)

    # Mostrar imagen del paciente con estudio negativo (Normal)
    imagen_negativa_path = os.path.join(base_path, region,
paciente_negativo, 'study1_negative', 'image1.png')
    imagen_negativa = Image.open(imagen_negativa_path)
    axs[1].imshow(imagen_negativa,cmap='gray')
    axs[1].axis('off')
    axs[1].set_title(f'Región: {region}\nEstudio: Normal', fontsize=12)

    plt.tight_layout()
    plt.show()

# Llamar a la función para mostrar una imagen positiva y una negativa
por cada región del cuerpo
for region in regiones:
    print(f"Mostrando imágenes para la región: {region}")
    mostrar_imagenes_aleatorias(base_path, region)
```

Código 8. Visualización ejemplo estudio normal y anormal por tipo de estudio

11.2. Estadística

```
# Estadísticos (máximo y mínimo) del repositorio
# con TODAS las imágenes de entrenamiento
# max = 255 min =0 - tarda 4 minutos en CPU
img_data.rgb=False

max, min = 0, 0
for i in range(len(img_data)):
    img, _ = img_data[i]

    aux = torch.max(img)
    max = aux if aux > max else max

    aux = torch.min(img)
    min = aux if aux < min else min


print('Mínimo: ', min, '\tMáximo: ', max)
```

Código 9

```
# Número total de píxeles y suma total para calcular media muestral
# Tarda dos minutos y medio en CPU

media  = 0
n_pxls = 0
for i in range(len(img_data)):
    img, _ = img_data[i]

    media = media + torch.sum(img)
    n_pxls = n_pxls + img.shape[0]*img.shape[1]*img.shape[2]

print('Número de píxeles: ', n_pxls, '\tSuma total: ', media.item(),
'\tMedia: ', (media/n_pxls).item())
```

Código 10

```
# Varianza muestral
# Tarda dos minutos y 50 segs en CPU

var  = 0
for i in range(len(img_data)):
    img, _ = img_data[i]

    var = var + torch.sum(torch.pow(img-media, 2))


print('Varianza muestral: ', (var/(n_pxls-1)).item(), 'Desviación típica
muestral: ', torch.pow(var/(n_pxls-1), 0.5).item())
```

Código 11

```
media = (media/n_pxls).item()
sigma = torch.pow(var/(n_pxls-1), 0.5).item()

print('Media (0-255): ', media, '\tDesviación típica (0-255): ', sigma)
print('Media (0-1): ', media/255, '\tDesviación típica (0-1): ',
sigma/255)
```

Código 12

11.3. Preparación de datos

```
# Defnición de clase MuraDataset para poner en un módulo que se importará
# El módulo es un fichero .py (por ejemplo muradataset.py) accesible
(para simplificar, en misma carpeta)

class MuraDataset(Dataset):
    # La clase hereda de torch,utils.data.Dataset
    # Los datasets en pytorch deben tener Dataset como ancestro

    # Variables de clase, comunes para todos los objetos de la misma
    dataset_name       = 'MURA-v1.1'
    annotation_files   = {'train':'train_labeled.csv',
'test':'valid_labeled.csv'}
    data_split_folders = {'train': 'train', 'test':'valid'}
    labels_map         = {0: 'negativo', 1:'positivo'}


# Método de inicialización de los objetos de la clase. Sobreescribe al
de Dataset
    def __init__(self, root_dir, study_type='XR_WRIST', train=True,
rgb=True, transform=None, target_transform=None):

        self.root_dir = root_dir
        self.train = train
        self.rgb = rgb
        #self.img_dir = os.path.normpath(
        #    (os.path.join(root_dir, dataset_name,
data_split_folders['train'], study_type) if train
        #                else os.path.join(root_dir, dataset_name,
data_split_folders['test'], study_type)))

        self.annotations_file =  os.path.normpath(
            (os.path.join(root_dir, dataset_name,
annotation_files['train']) if train else
                                  os.path.join(root_dir, dataset_name,
annotation_files['test'])))
        all_img_labels  = pd.read_csv(self.annotations_file) # Se carga
el fichero con todos los estudios
        # Se selecciona el tipo de estudio. EL data frame tiene la ruta
a cada imagen y su etiqueta
        self.img_labels = all_img_labels if study_type=='ALL' \
            else
all_img_labels.iloc[(all_img_labels['Path'].str.contains(study_type)).to
_numpy()].copy()
        self.transform = transform
        self.target_transform = target_transform
```

```
  # Método de longitud del dataset. Sobreescribe al de Dataset
    def __len__(self):
        return len(self.img_labels)

    # Método para obtener una única imagen y su etiqueta correspondiente
    def __getitem__(self, idx):
        # Ruta al fichero con la imagen
        img_path = os.path.join(self.root_dir, self.img_labels.iloc[idx,
0])

        im_mode= ImageReadMode.RGB if self.rgb else ImageReadMode.GRAY
        image = read_image(img_path, mode=im_mode)

        # Se obtiene la etiqueta correspondiente
        label = self.img_labels.iloc[idx, 1]

        if self.transform:
            image = self.transform(image)
        if self.target_transform:
            label = self.target_transform(label)
        return image, label
```

Código 13

```
# Comprobamos que pueden leerse las imágenes en gris y en RGB
# en Pytorch las bandas son C (color),H (altura),W (anchura)
# Mientras que normalmente es H, W, C. Hay que tenerlo en cuenta para
visualizar
img_data = MuraDataset(root_dir, study_type='ALL', rgb=False)
sample_idx = torch.randint(len(img_data), size=(1,)).item()
img, label = img_data[sample_idx]

print(img.shape, '\t', img.dtype)
print('Pixel arbitrario:\t', img[:,10,150])
```

Código 14

```
# Convertimos imagen de tensores a imagen PIL para representarla
plt.imshow(F.to_pil_image(img), cmap='gray')
patient = re.search(r'\bpatient\d+\b',
img_data.img_labels.iloc[sample_idx,0]).group()

plt.xlabel(patient+'\n'+
os.path.basename(img_data.img_labels.iloc[sample_idx,0]))
plt.title(img_data.labels_map[label])
plt.show()
```

Código 15

```
# Explícitamente pedimos que se lea como RGB, pues estaba puesto como
gris
img_data.rgb=True
img, label = img_data[sample_idx]

print(img.shape, '\t', img.dtype)
print('Pixel arbitrario:\t', img[:,10,150])
```

Código 16

```
# Convertimos imagen de tensores a imagen PIL para representarla
plt.imshow(F.to_pil_image(img), cmap='gray')

patient = re.search(r'\bpatient\d+\b',
img_data.img_labels.iloc[sample_idx,0]).group()
plt.xlabel(patient+'\n'+
os.path.basename(img_data.img_labels.iloc[sample_idx,0]))
plt.title(img_data.labels_map[label])
plt.show()
```

Código 17

```
# La clase Dataset permite transformar las imágenes, para
# adecuarlas a la cadena posterior de procesado
# En el caso de DenseNet, podemos tomar de referencia
# https://pytorch.org/hub/pytorch_vision_densenet/
# La red espera imágenes RGB

transform = Compose([
    # Se ajustan las dimensiones de la imagen al tamaño que espera la
red neuronal
    Resize(size=(224,224)),
    # Las imágenes se convierten a punto flotante (eran enteras)
    ConvertImageDtype(torch.float),
    # Se normalizan las imágenes, para que resulten con media cero y
sigma=1
    # Suelen utilizarse los valores de los canals RGB de ImageNet
    # aunque esto no es realmente consistente. En particular si la
imagen es
    # gris con tres canales, al normalizar resultarán canales con
valores diferentes
    # y se verán en color. Los píxeles pueden tener valores negativos
    #Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
     # Pueden usuarse alternativamente los valores de media y de sigma
     # obtenidos para el repositorio. En nuestro caso
     Normalize(mean=[0.206, 0.206, 0.206], std=[0.0419, 0.0419, 0.0419])
     # o si se usa un solo canal
    #Normalize(mean=0.206, std=0.0419)
     ])
```

Código 18

```
training_data = MuraDataset(
    root_dir=root_dir,
    study_type='XR_WRIST',
    train=True,
    rgb=True,
    transform=transform
    )

print('Tamaño del dataset de entrenamiento:\t', len(training_data))

test_data = MuraDataset(
    root_dir=root_dir,
    study_type='XR_WRIST',
    train=False,
    rgb=True,
    transform=transform
    )

print('Tamaño del dataset de test:\t', len(test_data))
```

Código 19

11.4. Entrenamiento del modelo

```
# Los dataloaders se hacen con la clase de Pytorch
# EL dataloader se encarga de ordenar aleatoriamiente las imágenes
# y de agruparlas en "batches" o grupos. La métrica de entrenamiento
# se calcula para las imágenes del batch (gradiente estocástico)
# En cada iteración el dataloader proporciona un "batch" completo
# El batch corresponde a la primera dimensión del tensor: B,C,H,W

train_dataloader = DataLoader(training_data, batch_size=64,
shuffle=True)
print('Número de batches de entrenamiento:\t', len(train_dataloader))

test_dataloader = DataLoader(test_data, batch_size=64, shuffle=True)
print('Número de batches de test:\t', len(test_dataloader))
```

Código 20

```
# Visualizamos la 0primera imagen del primer batch
# del conjunto de entrenamiento.
# Cada vez que se vuelva a ejecutar pasa al "batch" siguiente
train_imgs, train_labels = next(iter(train_dataloader))

print(f"Batch shape (images): {train_imgs.size()}")
print(f"Batch shape (labels): {train_labels.size()}")

img = train_imgs[0]
label = train_labels[0]

plt.imshow(F.to_pil_image(img*0.0419+0.206), cmap="gray")
plt.title(training_data.labels_map[label.item()])
plt.show()
```

Código 21

```
# Para continuar con el pipeline hay que recrear los dataloaders
# Pues se han consumido parcialmente al correr antes
# train_imgs, train_labels = next(iter(train_dataloader))

train_dataloader = DataLoader(training_data, batch_size=64,
shuffle=True)
print('Número de batches de entrenamiento:\t', len(train_dataloader))

test_dataloader = DataLoader(test_data, batch_size=64, shuffle=True)
print('Número de batches de test:\t', len(test_dataloader))
```

Código 22

```
device = ('cuda' if torch.cuda.is_available()
  else 'cpu')

print('Using device: ', device)
```

Código 23

```
model = torch.hub.load('pytorch/vision:v0.10.0', model='densenet169',
weights='DenseNet169_Weights.DEFAULT')
```

Código 24

```
model.classifier.out_features = 2
```

Código 25

```
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(params = model.parameters(), lr = 0.001)
exp_lr_scheduler = lr_scheduler.StepLR(optimizer, step_size=7,
gamma=0.1)
```

Código 26

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()
    model.to(device)
    # Create a temporary directory to save training checkpoints
    with TemporaryDirectory() as tempdir:
        best_model_params_path = os.path.join(tempdir,
'best_model_params.pt')

        torch.save(model.state_dict(), best_model_params_path)
        best_acc = 0.0

        for epoch in range(num_epochs):
            print(f'Epoch {epoch}/{num_epochs - 1}')
            print('-' * 10)
```

```
 # Each epoch has a training and validation phase
            for phase in ['train', 'test']:
                if phase == 'train':
                    model.train()  # Set model to training mode
                    dataloader = train_dataloader
                else:
                    model.eval()   # Set model to evaluate mode
                    dataloader = test_dataloader

                running_loss = 0.0
                running_corrects = 0

                # Iterate over data.
                for inputs, labels in dataloader:
                    inputs = inputs.to(device)
                    labels = labels.to(device)

                    # zero the parameter gradients
                    optimizer.zero_grad()

                   # forward
                    # track history if only in train
                    with torch.set_grad_enabled(phase == 'train'):
                        outputs = model(inputs)
                        _, preds = torch.max(outputs, 1)
                        loss = criterion(outputs, labels)

                        # backward + optimize only if in training phase
                        if phase == 'train':
                            loss.backward()
                            optimizer.step()
                  # statistics
                    running_loss += loss.item() * inputs.size(0)
                    running_corrects += torch.sum(preds == labels.data)
                if phase == 'train':
                    scheduler.step()

                epoch_loss = running_loss / len (dataloader.dataset)
                epoch_acc = running_corrects.double() / len
(dataloader.dataset)

                  print(f'{phase} Loss: {epoch_loss:.4f} Acc:
            {epoch_acc:.4f}')
```

```
# deep copy the model
                if phase == 'test' and epoch_acc > best_acc:
                    best_acc = epoch_acc
                    torch.save(model.state_dict(),
best_model_params_path)

            print()

        time_elapsed = time.time() - since
        print(f'Training complete in {time_elapsed // 60:.0f}m
{time_elapsed % 60:.0f}s')
        print(f'Best val Acc: {best_acc:4f}')

        model.load_state_dict(torch.load(best_model_params_path))
    return model
```

Código 27

```
def      visualize_model(model,      test_dataloader,labels_map,device,
num_images=6):
    was_training = model.training
    model.eval()
    model.to(device)
    images_so_far = 0
    fig = plt.figure()



    with torch.no_grad():
        for i, (inputs, labels) in enumerate(test_dataloader):
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            probs = torch.softmax(outputs, dim=1)
            _, preds = torch.max(outputs, 1)




            for j in range(inputs.size()[0]):
                images_so_far += 1
                ax = plt.subplot(num_images//2, 2, images_so_far)
                ax.axis('off')
                prob = probs[j, preds[j]].item()
                label_idx = preds[j].item()
                label_name = labels_map[label_idx]
                ax.set_title(f'predicted: {label_name}\nprobability:
{prob:.4f}')

                imshow(inputs.cpu().data[j])

                if images_so_far == num_images:
                    model.train(mode=was_training)
                    return
        model.train(mode=was_training)
```

Código 28

```
model = train_model(model, criterion, optimizer, exp_lr_scheduler,
                    num_epochs=25)
```

Código 29

```
labels_map          = {0: 'negativo', 1:'positivo'}
```

Código 30

```
visualize_model(model, test_dataloader, labels_map, device, num_images=6)
```

Código 31

11.5. Evaluación del modelo y obtención de resultados

```
def evaluate_model(model, dataloader, device):
    model.eval()
    tp, fp, tn, fn = 0, 0, 0, 0
    y_true = []
    y_scores = []

    with torch.no_grad():
        for inputs, labels in dataloader:
            inputs = inputs.to(device)
            labels = labels.to(device)

            outputs = model(inputs)
            _, preds = torch.max(outputs, 1)

            tp += ((preds == 1) & (labels == 1)).sum().item()
            fp += ((preds == 1) & (labels == 0)).sum().item()
            tn += ((preds == 0) & (labels == 0)).sum().item()
            fn += ((preds == 0) & (labels == 1)).sum().item()

  # Guardar las etiquetas verdaderas y las puntuaciones de predicción
para calcular el AUC
            y_true.extend(labels.cpu().numpy())
            y_scores.extend(torch.softmax(outputs, dim=1)[:,
1].cpu().numpy())

    accuracy = (tp + tn) / (tp + tn + fp + fn)
    precision = tp / (tp + fp)
    recall = tp / (tp + fn)
```

```
    if len(y_true) > 0:  # Verificar si y_true no está vacío
        # Calcular AUC
        y_true = np.array(y_true)
        y_scores = np.array(y_scores)
        fpr, tpr, _ = roc_curve(y_true, y_scores, pos_label=1)  #
Especificar pos_label
        roc_auc = auc(fpr, tpr)
    else:
        roc_auc = 0.5  # AUC predeterminado si no hay muestras positivas

    print("True Positive:", tp)
    print("False Positive:", fp)
    print("True Negative:", tn)
    print("False Negative:", fn)
    print("Precision:", precision)
    print("Recall:", recall)
    print("Accuracy:", accuracy)
    print("AUC:", roc_auc)

    # Plot ROC curve
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange', lw=lw, label='AUC(area =
%0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic Curve')
    plt.legend(loc="lower right")
    plt.show()



evaluate_model(model, test_dataloader, device)
```

Código 32

```
# Set the model to evaluation mode
model.eval()

# Lists to store true labels and predicted labels
y_true = []
y_pred = []

# Disable gradient computation
with torch.no_grad():
    for test_data in test_dataloader: # Iterate over the test dataloader
to get test images and labels
        test_images, test_labels = (test_data[0].to(device),
                                    test_data[1].to(device))
        # Forward pass through the model to get predictions
        output = model(test_images)

        # Compute the predicted labels by taking the argmax of the
softmax output
        pred_label = torch.softmax(output, dim = 1).argmax(dim = 1)

        # Append the true labels and predicted labels to their
respective lists
        y_true.append(test_labels.cpu())
        y_pred.append(pred_label.cpu())

# Convert the lists of true labels and predicted labels to NumPy arrays
y_true = torch.cat(y_true).numpy()
y_pred = torch.cat(y_pred).numpy()
```

Código 33

```
# Generate the classification report and print it
report = classification_report(y_true,
                               y_pred,
                               target_names = ["Positive", "Negative"],
                               digits = 4)
print(report)
```

Código 34

```
# Generate the confusion matrix
cmat = confusion_matrix(y_true, y_pred, normalize = 'true')

# Create a heatmap for the confusion matrix visualization
ax = sns.heatmap(cmat, annot = True, cmap = 'Blues')

# Set labels for x-axis and y-axis
ax.set_xlabel('Predicted Values')
ax.set_ylabel('Actual Values')

# Set the labels
ax.xaxis.set_ticklabels(["Positive","Negative"])
ax.yaxis.set_ticklabels(["Positive","Negative"])

# Display the visualization of the Confusion Matrix
plt.show()
```

Código 35