UNMdP - Facultad de Ingeniería - Departamento de Informática

Sistema de reconocimiento de estados pulmonares en ecografías mediante Inteligencia Artificial

Proyecto final para optar al grado de Ingeniero en Informática

Autores

- Pruis, Ramiro
- Garcia, Mariano Enrique

Director

- Meschino, Gustavo

Codirector

- Finochietto, Mariano

1. Agradecimientos

A nuestras familias y amigos, por ser un pilar fundamental no solo durante el desarrollo de este proyecto, sino a lo largo de toda nuestra formación profesional. Sin su apoyo incondicional, este camino no habría sido posible.

A nuestro director, Gustavo Meschino, por la voluntad para acercarnos a este proyecto, tomar el rol de dirigirlo y estar siempre a disposición para el desarrollo del mismo sin importar fecha ni horario.

A la Facultad de Ingeniería y a todos los docentes que nos formaron durante estos años, por brindarnos los conocimientos, las herramientas y la oportunidad de culminar esta etapa y dar inicio a una nueva como profesionales.

2. Índice

3. Resumen

Este proyecto propone el desarrollo de una herramienta de software que, mediante inteligencia artificial, analiza y predice los estados de aireación pulmonar a partir de videos de ecografías pulmonares. La iniciativa surge de la necesidad de mejorar la precisión diagnóstica en contextos clínicos donde factores como la baja calidad de imagen o la variabilidad del operador dificultan la evaluación. El sistema identifica patrones asociados a distintos estados de aireación pulmonar (Normal, B1, B2 y Consolidación), posteriormente genera métricas visuales y permite emitir informes automáticos que apoyan el diagnóstico médico.

El trabajo se basa en investigaciones previas del Laboratorio de Bioingeniería de la Universidad Nacional de Mar del Plata y del Hospital Privado de la Comunidad, y plantea una solución integral: una aplicación de escritorio local, desarrollada con tecnologías modernas como React, Node.js, Electron y Python, que permite cargar, recortar y analizar múltiples videos de forma eficiente.

Se evaluaron distintos modelos de inteligencia artificial pre entrenados y se aplicaron técnicas de transferencia de aprendizaje para lograr resultados coherentes con los datos utilizados. El sistema permite además la carga de otros modelos de *deep learning*, aparte del entregado, lo que posibilita su mejora y expansión futura.

La aplicación ofrece una visualización detallada e interactiva de los resultados, permite generar informes sobre los estudios ingresados y facilita el procesamiento masivo de videos para fines de investigación. El informe detalla las decisiones tomadas con relación al entrenamiento del modelo de IA y al desarrollo del sistema, y plantea desarrollos futuros orientados a mejorar la experiencia del usuario y optimizar la capacidad de procesamiento y predicción de la red neuronal frente a nuevos datos.

4. Introducción

La ecografía pulmonar ha emergido en las últimas décadas como una técnica no invasiva, económica y efectiva para evaluar de manera eficaz el estado respiratorio de pacientes en contextos clínicos rutinarios y críticos. En este contexto, es importante comprender el rol de los especialistas, quienes deben contar con los criterios necesarios y la experiencia correspondiente para realizar diagnósticos exitosos. Sin embargo, se debe reconocer que hay factores que pueden afectar seriamente el análisis por parte de los médicos: baja calidad de las imágenes, ecógrafos con potencia insuficiente, dependencia de la ubicación del transductor del equipo, diferencias entre prestaciones de ecógrafos de diferentes marcas, entre otros.

Frente a la situación mencionada anteriormente, la incorporación de Inteligencia Artificial (IA) representa una oportunidad significativa para objetivar y estandarizar la evaluación diagnóstica, disminuyendo la tasa de errores y proporcionando un instrumento adicional a los profesionales.

Este proyecto propone desarrollar un sistema computacional, basado en Inteligencia Artificial, capaz de analizar automáticamente videos provenientes de ecografías de pulmón. El sistema identificará patrones específicos que indican distintos estados pulmonares basados en el comportamiento y estado de la pleura . Además, ofrecerá una interfaz intuitiva que facilite la interpretación de los resultados a través de informes visuales y métricas claras. Al automatizar este análisis, se busca no solo apoyar a especialistas en diagnóstico por imágenes, sino también fomentar nuevas investigaciones médicas.

La implementación de la tecnología propuesta promete impactar positivamente en la precisión diagnóstica y en la formación de futuros médicos especialistas, contribuyendo al avance de la medicina respiratoria. También busca generar un impacto positivo en la sociedad, promoviendo la creación de nuevas herramientas que faciliten los procesos de diagnóstico, alentando la inclusión de la Inteligencia Artificial en diversos entornos.

5. Objetivos generales

El objetivo principal del proyecto es desarrollar e implementar un sistema informático basado en un modelo preliminar de Inteligencia Artificial, orientado al análisis automático de videos provenientes de ecografías pulmonares. Mediante esta tecnología, se busca entrenar y validar un clasificador de aprendizaje profundo capaz de identificar los cuatro estados de aireación pulmonar definidos (Normal, B1, B2 y Consolidación) en imágenes individuales extraídas de ecografías.

El sistema está diseñado para procesar múltiples videos de forma secuencial y generar reportes de probabilidad por frame, proporcionando una base tecnológica para futuros desarrollos en diagnóstico automatizado.

Como objetivos secundarios, el proyecto busca establecer una metodología replicable para el procesamiento de datos ecográficos y la implementación de modelos de clasificación en este dominio específico. Se espera que este desarrollo preliminar sirva como punto de partida para investigaciones futuras en el campo de la medicina respiratoria asistida por IA, proporcionando a investigadores una herramienta de análisis que permita la evaluación sistemática de patrones ecográficos pulmonares con fines de investigación y validación clínica.

6. Análisis previo

6.1 Introducción al contexto de investigación

En los últimos años, el uso de la ecografía pulmonar como herramienta para el diagnóstico dinámico y no invasivo de patologías respiratorias ha cobrado un interés creciente en el ámbito clínico (Soldati et al., 2019). Particularmente, en pacientes sometidos a anestesia general o a ventilación mecánica, la aparición de atelectasia —colapso parcial o total de las unidades alveolares— representa una complicación frecuente que compromete el intercambio gaseoso y la mecánica respiratoria, además de incrementar el riesgo de daño pulmonar inducido por la ventilación (Tusman et al., 2012).

Frente a esta problemática, el desarrollo de maniobras de reclutamiento pulmonar ha buscado revertir las atelectasias mediante incrementos controlados de presión en las vías aéreas. Sin embargo, la falta de herramientas prácticas que permitieran evaluar en tiempo

real la eficacia de dichas maniobras constituía hasta hace poco un obstáculo en su implementación clínica. En este contexto, el Laboratorio de Bioingeniería del ICYTE (Universidad Nacional de Mar del Plata y CONICET) y el Hospital Privado de Comunidad, con la participación de los expertos Dr. Gerardo Tusman y Dra. Cecilia Acosta, realizaron investigaciones pioneras para aplicar técnicas de ultrasonido pulmonar en la valoración dinámica del estado de aireación pulmonar (Tusman et al., 2017; Acosta et al., 2014).

El ultrasonido pulmonar demostró ser una técnica altamente sensible y específica para detectar colapso pulmonar (Xirouchaki et al., 2011), con la ventaja adicional de ser no invasiva, portátil y realizable a pie de cama. Estas características la posicionan como una herramienta ideal para la evaluación instantánea de la condición pulmonar en pacientes críticos, abriendo nuevas posibilidades para la personalización de estrategias ventilatorias y la reducción de complicaciones asociadas.

6.2 Clasificación de estados de aireación pulmonar

Como parte de estas investigaciones, se consolidó una clasificación estandarizada de los patrones de aireación pulmonar observables mediante ecografía (Bouhemad et al., 2007). Esta clasificación permite segmentar el estado de los pulmones en cuatro categorías principales:

- Estado Normal (líneas A): Corresponde a la presencia de líneas horizontales paralelas a la pleura, indicativas de un pulmón bien aireado y en condiciones normales.

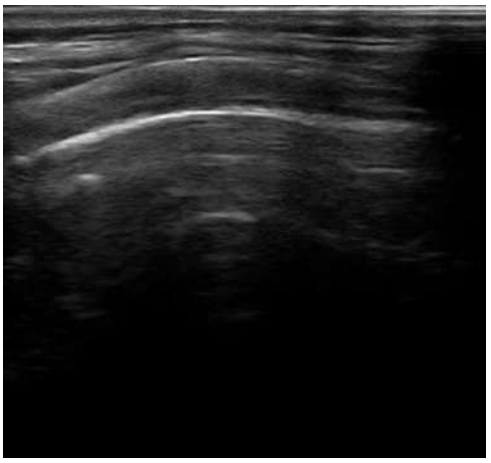

Figura 1. Ecografía marcada como Normal
(ausencia de patologías respiratorias, pulmón correctamente aireado).

- Pérdida moderada de aireación (líneas B1): Caracterizada por múltiples líneas B dispersas e irregulares, que representan un grado inicial de pérdida de aireación, con zonas parcialmente colapsadas.

Figura 2. Ecografía marcada como B1
(patrón de líneas B dispersas, indicativas de una pérdida moderada de aireación.).

- Pérdida severa de aireación (líneas B2): Definida por la aparición de líneas B coalescentes, donde las zonas sin aireación ocupan una proporción mucho mayor del parénquima pulmonar.

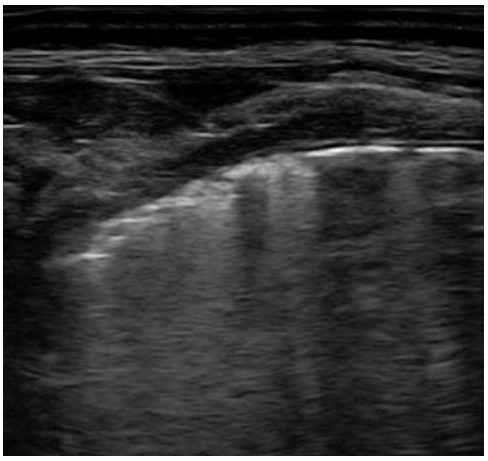

Figura 3. Ecografía clasificada como B2
(patrón de líneas que denotan una severa pérdida de aireación).

- Pérdida completa de aireación (Consolidación): Representa la transformación completa del tejido pulmonar aireado en un patrón ecográfico similar al del tejido hepático, reflejando colapso alveolar total.

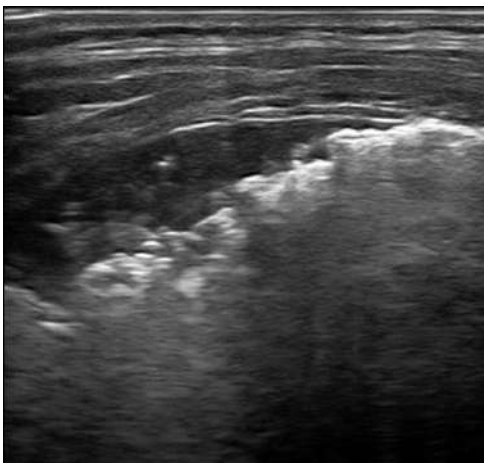

Figura 4. Ecografía clasificada como Consolidación
(patrón que denota una pérdida completa de aireación).

Estos patrones no solo permiten identificar la presencia de atelectasias, sino también cuantificar su severidad mediante la asignación de "scores de aireación" (LUS scores), brindando así un marco sistemático para la evaluación y seguimiento de los pacientes. La evolución de estos scores en el tiempo constituye un parámetro objetivo para valorar la respuesta de los pulmones a las intervenciones clínicas, particularmente a las maniobras de reclutamiento alveolar.

6.3 Resultados de las investigaciones previas

Los trabajos de investigación realizados permitieron validar la capacidad del ultrasonido pulmonar para guiar maniobras de reclutamiento de aire y personalizar la selección de parámetros ventilatorios en pacientes bajo asistencia respiratoria. Además, se exploró la aplicación de técnicas de aprendizaje automático y aprendizaje profundo (*Deep Learning*) para automatizar la clasificación de los patrones.

En las investigaciones tecnológicas se evaluaron modelos de redes neuronales convolucionales (CNN), destacándose el uso preliminar de la arquitectura MobileNet mediante estrategias de *transfer-learning* (Howard et al., 2019).

Los resultados mostraron una alta correlación entre las predicciones automáticas y la evaluación realizada por especialistas médicos, lo que consolidó el potencial de la inteligencia artificial como herramienta de apoyo al diagnóstico dinámico pulmonar. No obstante, dichos resultados permanecieron circunscriptos al ámbito experimental y académico, sin haberse traducido en una herramienta computacional integrada, documentada y validada para su uso en entornos clínicos reales.

El presente proyecto toma como base esa experiencia previa, proponiendo una solución completa que integre un modelo de clasificación robusto dentro de un sistema informático operable, con validaciones más estructuradas y documentación orientada a su aplicación en entornos reales.

6.3.1 Limitaciones iniciales

Al iniciar el proyecto, se identificaron varias limitaciones inherentes que condicionaron el desarrollo tecnológico. En primer lugar era necesario diseñar y desarrollar un modelo de Inteligencia Artificial operativo que pudiera ser incorporado directamente al software propuesto. Los prototipos previos, realizados por el personal del Laboratorio de Bioingeniería, no habían sido formalmente implementados ni adaptados para su transferencia tecnológica, lo que obligó a replantear todo el proceso de entrenamiento desde cero.

Por otra parte, el conjunto de datos disponible para el entrenamiento del modelo era limitado en volumen y presentaba alta heterogeneidad. A diferencia de las grandes bases de datos comúnmente empleadas en aplicaciones de Deep Learning, que requieren decenas o cientos de miles de imágenes para lograr un entrenamiento robusto, el dataset inicial consistía en una cantidad acotada de videos e imágenes etiquetadas manualmente, provenientes de diferentes pacientes y condiciones clínicas.

La etiquetación manual de cada imagen o secuencia de video, realizada por especialistas médicos, constituía otro factor limitante, dado que este proceso intensivo demandaba una importante dedicación de tiempo y recursos humanos altamente capacitados. Esto restringía la velocidad con la que podía ampliarse el conjunto de entrenamiento y aumentaba la posibilidad de sesgos o inconsistencias en la asignación de etiquetas.

Ante estas restricciones, se optó por implementar estrategias específicas para optimizar el uso de los datos disponibles. Se utilizó aprendizaje por transferencia sobre modelos pre entrenados, reduciendo así la cantidad de datos necesaria para alcanzar una capacidad predictiva aceptable. Además, se aplicaron técnicas de data augmentation (Shorten & Khoshgoftaar, 2019), para incrementar la diversidad del conjunto y mejorar la generalización del modelo.

Estas limitaciones iniciales configuraron un escenario de alta complejidad, en el que la construcción del modelo predictivo requería no solo habilidades técnicas avanzadas, sino también una estrategia de trabajo cuidadosa, que permitiera maximizar la eficiencia del

proceso de entrenamiento y validación, garantizando al mismo tiempo un nivel de desempeño clínicamente aceptable.

7. Planificación inicial

7.1 Planteo de la solución inicial

El proyecto tuvo como objetivo principal el desarrollo de un sistema informático que integrara modelos de inteligencia artificial para analizar automáticamente videos de ecografías pulmonares, detectando patrones de aireación y generando reportes interpretables de manera rápida y objetiva.

La solución planteada contempla un flujo de trabajo sencillo y eficiente: el usuario podrá cargar uno o múltiples videos de ecografías al sistema, seleccionar las zonas de interés mediante una herramienta de recorte incorporada, y obtener en pocos segundos un análisis detallado que indique el estado de aireación pulmonar en cada región evaluada.

Desde el punto de vista técnico, se planteó la construcción de una aplicación de escritorio local mediante el uso de Electron, un *framework* que encapsula tanto el *frontend* como el *backend* de la aplicación. Esta estructura permitió simular una separación entre la interfaz de usuario (desarrollada en React) y la lógica de procesamiento (implementada en Node.js), manteniendo internamente una organización similar al patrón cliente-servidor, pero con ejecución completamente local.

El análisis automático de imágenes se planificó mediante scripts desarrollados en Python que implementaron modelos de aprendizaje profundo. Estos scripts serían invocados posteriormente desde el backend local para procesar los frames extraídos de los videos, generando como resultado un archivo en formato JSON con la clasificación correspondiente a cada imagen.

7.2 Cronograma Previsto

Dentro del proceso de planificación inicial, se elaboró un cronograma de trabajo detallado con el objetivo de organizar y estructurar las actividades requeridas para alcanzar los objetivos planteados. La asignación de tiempos y la secuenciación de las tareas se definieron en función de los conocimientos previos del equipo de trabajo, tomando como

referencia la estructura general que implica un desarrollo de software completo, desde la fase de análisis hasta la entrega final del producto.

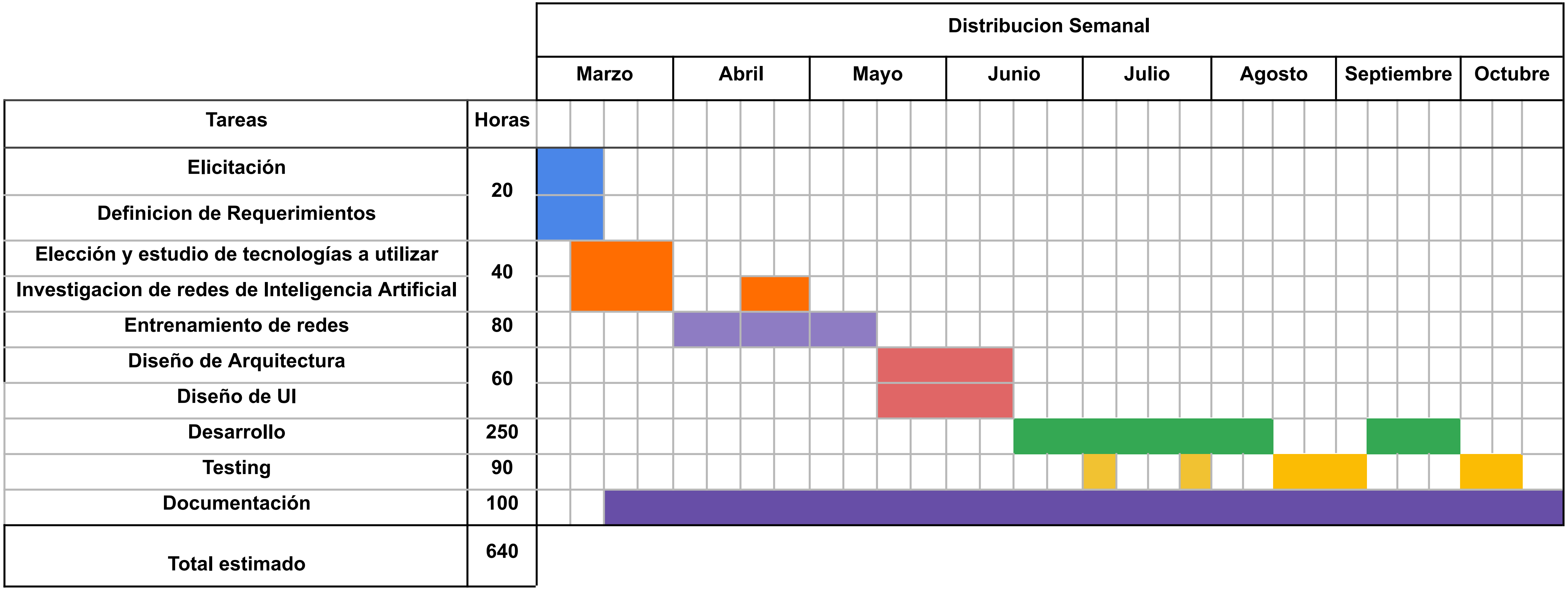

Figura 5. Diagrama de Gantt presentado en el protocolo.

7.3 Análisis FODA

A la par de la planificación, se realizó un análisis FODA con el propósito de identificar las principales fortalezas, debilidades, oportunidades y amenazas que podrían influir en el desarrollo y la implementación del proyecto. Cabe destacar que, al momento de efectuar este análisis inicial, no se contemplaban algunas de las limitaciones prácticas que surgieron posteriormente durante la ejecución. Por este motivo, al finalizar el desarrollo, se llevó a cabo una nueva evaluación FODA que incorporó las restricciones y desafíos que se presentaron a lo largo del proceso.

Factores internos

Fortalezas

- **F1:** Experiencia previa en el análisis de algoritmos de aprendizaje automático.
- **F2:** Laboratorio de Bioingeniería altamente capacitado y especializado en inteligencia artificial.
- **F3:** Acceso a un amplio conjunto de datos de ecografías para el entrenamiento inicial de la red neuronal.
- **F4:** Capacidad para generar conclusiones gráficas y numéricas detalladas a partir de los datos analizados.
- **F5:** Posibilidad de generar informes automáticos y analizar el estado de las ecografías frame por frame, mejorando la precisión del diagnóstico.

Debilidades

- **D1:** Dependencia de un dataset específico para el entrenamiento del modelo, limitando la generalización inicial.
- **D2:** Necesidad de definir y ajustar adecuadamente el modelo de entrenamiento para lograr resultados clínicamente precisos.
- **D3:** Posibles limitaciones técnicas en la arquitectura y diseño general del sistema debido a restricciones de tiempo y recursos.

  D4: Requerimiento de recursos humanos altamente especializados en inteligencia artificial y bioingeniería para etapas clave del desarrollo.
- **D5:** Potencial necesidad de recurrir a consultoría externa para resolver desafíos técnicos específicos fuera del alcance del equipo.

Factores externos

Oportunidades

- **O1:** Demanda creciente de soluciones de inteligencia artificial en el campo de la medicina, especialmente en el diagnóstico asistido por imágenes.
- **O2:** Posibilidad de colaboración con instituciones médicas y profesionales del área de la bioingeniería para validar y expandir el proyecto.
- **O3:** Acceso a avances tecnológicos constantes que pueden mejorar el rendimiento, la precisión y la capacidad de escalado del sistema.
- **O4:** Potencial de expansión del proyecto a nivel internacional, dado el interés global en aplicaciones de IA para el área médica.

Amenazas

- **A1:** Competencia creciente en el mercado de inteligencia artificial aplicada a la medicina, con aparición constante de nuevas soluciones.
- **A2:** Riesgo de cambios regulatorios que puedan restringir la implementación o comercialización de sistemas de diagnóstico asistido por IA.
- **A3:** Posibles limitaciones presupuestarias que afecten la continuidad del desarrollo, mantenimiento y mejora del producto.
- **A4:** Riesgo de falta de aceptación o adopción por parte de profesionales médicos, por desconfianza hacia sistemas automatizados.
- **A5:** Posible escasez de recursos tecnológicos o infraestructura adecuada para el despliegue y uso óptimo del sistema en algunos entornos clínicos.

7.4 Análisis de riesgos

En base a las amenazas identificadas y al análisis de la naturaleza del proyecto, se determinaron los factores que podrían poner en riesgo su finalización o generar demoras en el proceso de desarrollo. Para cada riesgo se asignaron valores de probabilidad de ocurrencia (OP) e impacto (I), utilizando una escala de 1 a 3 en ambos casos.

El peso de cada riesgo (W) se calculó como el producto entre la probabilidad de ocurrencia y su impacto ($W = OP \times I$).

Para aquellos riesgos cuyo peso fue mayor o igual a 6, se elaboraron planes de contingencia que permitan anticipar su ocurrencia y establecer acciones concretas para mitigarlos en caso de materializarse.

Tabla 1. Análisis de riesgos

Riesgo	Descripción	Consecuencia	OP	I	W
RI 01	Falta de datos suficientes para el entrenamiento del modelo	El modelo no arroja resultados precisos No comprende los datos de entrada	2	3	6
RI 02	Falta de experiencia técnica en el equipo de desarrollo	Retrasos en el desarrollo Necesidad de aprendizaje	2	2	4
RI 03	Falta de aceptación por parte de los usuarios o profesionales médicos	Los clientes no hacen uso del producto	2	3	6
RI 04	Escasez de recursos de hardware adecuados	No se puede desarrollar un correcto entrenamiento del modelo de IA	2	3	6
RI 05	Cambios en la tecnología o avances científicos que puedan hacer que el software se vuelva obsoleto	Problemas de rendimiento	1	2	2
RI 06	Fallas en el test de integración	Debuggear para encontrar errores Corrección de los mismos Retrasos en el cronograma	3	2	6
RI 07	Cambios en la disponibilidad de los integrantes del equipo	Retrasos en el cronograma	3	2	6

7.5 Planes de contingencia

RI01. Realizar una exhaustiva búsqueda y recopilación de datos pulmonares relevantes. Considerar la colaboración con instituciones médicas para acceder a conjuntos de datos más amplios.

RI03. Realizar pruebas de concepto o demostraciones con potenciales usuarios durante el desarrollo, para recolectar feedback anticipado y adaptar la herramienta a sus expectativas.

RI04. Priorizar modelos de IA ligeros y optimizados. En caso de limitaciones críticas, utilizar servicios en la nube de forma puntual para completar las etapas de entrenamiento o análisis más demandantes

RI 06. Si la corrección de los errores toma más tiempo del establecido, el equipo trabajará tiempo extra para resolver los problemas

RI 07. Reorganizar las tareas de forma que no queden exclusivamente asignadas a un único integrante. Priorizar entregables parciales que aseguren avances concretos aunque se reduzca temporalmente la disponibilidad de los miembros del equipo.

8. Análisis

8.1 Definición de la estrategia de procesamiento de datos

Durante las etapas iniciales del proyecto, uno de los primeros aspectos analizados fue la forma de abordar los datos de entrada provenientes de las ecografías pulmonares. Se evaluaron dos alternativas principales: procesar los videos de forma integral, considerando su secuencia temporal completa, o analizar individualmente cada fotograma de los videos, tratándolos como instancias independientes de clasificación.

Esta decisión estratégica resultaba fundamental, ya que influía de manera directa sobre la complejidad del desarrollo de la herramienta de clasificación, los requerimientos de procesamiento computacional y el diseño general del sistema.

Dado que el proyecto no contaba con un modelo pre-entrenado capaz de clasificar secuencias de video completas, y considerando además que el conjunto de datos disponible estaba compuesto por fotogramas individuales de ecografías ya etiquetados según su estado pulmonar, se optó por la segunda alternativa: construir un modelo de clasificación basado en el análisis de imágenes.

Esta elección permitió estructurar el sistema en torno a una red neuronal convolucional optimizada para trabajar con imágenes estáticas. Cada *frame* extraído de un video sería procesado de manera independiente para determinar su correspondiente estado de aireación pulmonar.

La decisión de realizar la clasificación de esa manera ofreció varias ventajas prácticas para el desarrollo del sistema:

- Facilitó la adaptación del proceso de entrenamiento, al poder aprovechar directamente el dataset existente sin necesidad de generar secuencias sintéticas o reestructurar los datos disponibles.

- Redujo la complejidad del modelo, evitando la necesidad de emplear arquitecturas específicas para procesamiento de secuencias temporales (como redes LSTM, GRU o Transformers).
- Permitió mantener una mayor modularidad en el diseño del sistema, ya que cada fotograma podría ser analizado de manera autónoma, y luego consolidar los resultados a nivel global.

Esta definición temprana también permitió clarificar y delimitar con mayor precisión los requerimientos funcionales y no funcionales, asegurando que el desarrollo posterior del sistema se adecuará de manera coherente a las necesidades técnicas del proyecto.

8.2 Metodología de Desarrollo

Para la gestión y planificación de las tareas a lo largo del proyecto se adoptó la metodología ágil Kanban. Su uso permitió visualizar el estado de las actividades, gestionar su creación y asignación, y realizar estimaciones de manera continua a medida que avanzaba el proyecto.

El tablero Kanban fue implementado en la plataforma Trello, donde se organizaron las tareas en distintas columnas representando los estados de avance: tareas por hacer, en progreso y finalizadas. Este enfoque facilitó la adaptación a cambios en las prioridades y permitió mantener una organización visual y sencilla del flujo de trabajo.

El desarrollo de la aplicación se realizó utilizando Git como sistema de control de versiones, alojando los repositorios en la plataforma GitHub. Esta herramienta permitió gestionar de manera efectiva las distintas versiones del código, integrar cambios de forma controlada y mantener un historial detallado de las modificaciones realizadas.

Para asegurar la estandarización de los mensajes de cambios y facilitar su interpretación, se aplicó el estándar *Conventional Commits*, que establece reglas específicas para los mensajes de confirmación de cambios, utilizando prefijos cómo *feat* para nuevas funcionalidades, *fix* para correcciones, *chore* para tareas de mantenimiento, entre otros.

8.3 Requerimientos funcionales

Los requerimientos funcionales corresponden a las acciones y comportamientos específicos que el sistema debe soportar para satisfacer las necesidades planteadas. A continuación, se detallan los requisitos fundamentales del sistema en cuestión:

Código del Requerimiento	RF01
Nombre	Carga de videos
Propósito	Permitir la carga de videos de ecografías pulmonares
Descripción	El usuario debe contar con la posibilidad de subir 1 o más videos referidos a un mismo paciente. Esto es así ya que un análisis estricto cuenta con 6 zonas analizadas por lado del pulmón, siendo el máximo de videos 12.
Entrada	Videos en formatos a definir.
Salida	Videos cargados en la aplicación listos para ser recortados.
Prioridad de Requerimiento	Alta

Código del Requerimiento	RF02
Nombre	Recorte de videos
Propósito	Darle la posibilidad al usuario de que recorte los videos luego de cargarlos.
Descripción	Como los vídeos producidos por los ecógrafos suelen generar metadatos en la imagen, el usuario debe contar con la posibilidad de seleccionar una zona del video para recortarla a su gusto.
Entrada	Videos previamente cargados en el software.
Salida	Videos recortado, quedando la zona de interés a analizar.
Prioridad de Requerimiento	Alta

Código del Requerimiento	RF03
Nombre	Cálculo de métricas
Propósito	Realizar cálculo de métricas a partir de las ecografías analizadas.
Descripción	A partir del análisis realizado por el modelo de Inteligencia Artificial, se deben calcular métricas asociadas al estudio. Debe permitir al usuario elegir el método de cálculo.
Entrada	Método de cálculo de métricas
Salida	Métricas calculadas
Prioridad de Requerimiento	Alta

Código del Requerimiento	RF04
Nombre	Informe de resultados
Propósito	Brindar un método de exportar el análisis
Descripción	Construir en forma automática un informe de los vídeos analizados. El formato podrá ser .PDF, .XLS dependiendo del requisito (estética, facilidad de visualización)
Entrada	Videos analizados por el modelo de inteligencia artificial
Salida	Informe generado en formato especificado
Prioridad de Requerimiento	Media

Código del Requerimiento	RF05
Nombre	Análisis en Batch
Propósito	Permitir el análisis de múltiples videos en simultáneo y en forma rápida.
Descripción	El usuario puede optar por ingresar los videos en batch, recortarlos y obtener los informes de cada uno, sin necesidad de visualizar en detalle el análisis realizado.
Entrada	Videos en batch
Salida	Informes
Prioridad de Requerimiento	Media

Código del Requerimiento	RF06
Nombre	Analisis frame a frame
Propósito	Proporcionar al usuario la posibilidad de visualizar el video en frames luego del análisis.
Descripción	El análisis frame a frame permite analizar en detalle las ecografías, brindando la posibilidad de contemplar los resultados arrojados por el modelo de inteligencia artificial.
Entrada	Video de ecografía recortado
Salida	Video con visualizador, gráfico con información del frame actual y del video completo
Prioridad de Requerimiento	Alta

Código del Requerimiento	RF07
Nombre	Elección de método de cálculo
Propósito	Proporcionar al usuario la posibilidad de elegir el método para el cálculo de las métricas derivadas de los resultados de la predicción del modelo.
Descripción	Al haber distintos criterios para la obtención de las métricas de aireación pulmonar, se desea dar la posibilidad elegir cuál de los provistos desea utilizar.
Entrada	Método de cálculo, Matriz de valores Softmax para cada video
Salida	Métrica calculada
Prioridad de Requerimiento	Media

8.4 Requerimientos no funcionales

Además de las funcionalidades específicas, se establecieron requerimientos no funcionales que definen atributos de calidad que el sistema debe cumplir.

Código del Requerimiento	RNF01
Nombre	Documentación
Propósito	Proporcionar la documentación adecuada para la posible continuación del proyecto
Descripción	Generar tanto la documentación necesaria para el uso del programa a desarrollar como la documentación del código a entregar en caso que se deseen realizar modificaciones o expansiones al mismo
Prioridad de Requerimiento	Media

Código del Requerimiento	RNF02
Nombre	Interfaz de usuario
Propósito	Crear una interfaz de usuario amigable
Descripción	Vincular el modelo de inteligencia artificial con una interfaz de usuario amigable que permita navegar y observar los resultados.
Prioridad de Requerimiento	Alta

Código del Requerimiento	RNF03
Nombre	Rendimiento
Propósito	Optimizar los recursos para obtener resultados en un tiempo acorde a lo esperado
Descripción	Utilizar distintas técnicas de optimización en el análisis de las ecografías para obtener resultados en un tiempo esperable
Prioridad de Requerimiento	Medio

9. Definición de la arquitectura

Durante la fase de diseño del sistema, uno de los aspectos fundamentales fue la elección de la arquitectura que mejor se adaptara a los requerimientos funcionales, no funcionales y a las limitaciones detectadas en las etapas de análisis. La arquitectura debía contemplar la necesidad de procesar grandes volúmenes de datos (imágenes de ecografías pulmonares), integrar un modelo de inteligencia artificial, ofrecer una experiencia de usuario moderna y ágil, y garantizar la operatividad autónoma del sistema, sin depender de infraestructura externa.

9.1 Alternativas consideradas

Inicialmente se contemplaron tres enfoques arquitectónicos posibles:

1. **Arquitectura web tradicional (cliente-servidor remoto):**
   Este enfoque implicaba desarrollar una aplicación web donde el procesamiento de los videos y la ejecución del modelo de inteligencia artificial se realizaría en un servidor remoto.

   - *Ventajas:* facilidad de actualización centralizada, escalabilidad para múltiples usuarios concurrentes, y el uso de tecnologías de procesamiento más avanzadas. En este aspecto, se consideró incluso la eventual implementación de técnicas como el aprendizaje federado, que permitiría a un modelo de inteligencia artificial re-entrenarse de forma continua a partir del feedback recibido por parte de los usuarios.
   - *Desventajas:* necesidad de una conexión permanente a internet, mayores tiempos de latencia, costos asociados al mantenimiento de infraestructura en la nube, y limitaciones técnicas para integrar modelos de procesamiento intensivo como redes neuronales.

2. **Aplicación de escritorio monolítica:**
   Se consideró el desarrollo de una aplicación de escritorio clásica, utilizando lenguajes como Java o C#, integrando en el mismo ejecutable tanto la interfaz gráfica como las rutinas de procesamiento.

   - *Ventajas:* Operación totalmente offline, control total sobre los recursos locales.
   - *Desventajas:* Mayor complejidad de desarrollo y mantenimiento, menor flexibilidad para integrar modelos de machine learning, y mayor rigidez en la actualización y expansión de funcionalidades.

3. **Arquitectura híbrida local basada en tecnologías web modernas:**
   Esta alternativa proponía utilizar tecnologías como React, Node.js y Electron para crear una aplicación de escritorio que simulara una arquitectura cliente-servidor internamente, ejecutándose de forma completamente local en el equipo del usuario.

- *Ventajas*: Operatividad offline sin necesidad de infraestructura externa, integración directa con modelos de machine learning desarrollados en otros lenguajes, experiencia de usuario moderna, fluida y modular, flexibilidad y escalabilidad para futuras actualizaciones.
- *Desventajas:* Tamaño de aplicación superior a una solución puramente web, necesidad de gestión cuidadosa de procesos locales.

9.2 Decisión sobre la arquitectura del sistema

Considerando todas las alternativas, se adoptó finalmente una arquitectura híbrida local basada en las siguientes tecnologías:

- **React:** Seleccionado para el desarrollo de la interfaz de usuario debido a su capacidad para diseñar y construir aplicaciones modernas, modulares y altamente interactivas. Su enfoque basado en componentes facilita la reutilización de código y la escalabilidad de la aplicación.

- **Node.js:** Utilizado como backend local para gestionar la lógica de negocio, coordinar la comunicación entre la interfaz y los procesos externos, y ejecutar de manera controlada los programas de procesamiento en Python.

- **Electron:** Empleado para empaquetar la aplicación como un ejecutable de escritorio multiplataforma (Electron, s.f.), permitiendo integrar frontend y backend en una única solución local que puede instalarse y ejecutarse de manera independiente del navegador web.

- **Vite:** Herramienta de construcción elegida para el proyecto debido a su rapidez, su soporte nativo para React y su eficiencia en tiempos de compilación y recarga en caliente durante el desarrollo (Vite, s.f.).

La integración de estas tecnologías permite implementar una arquitectura cliente-servidor moderna ejecutada completamente en el equipo local del usuario, optimizando el uso de los recursos de hardware disponibles y garantizando un funcionamiento eficiente y autónomo. La adopción de JavaScript como lenguaje común para el desarrollo del frontend y backend proporciona consistencia arquitectónica, facilita el mantenimiento del código y permite una implementación más precisa de los prototipos de interfaz diseñados.

9.3 Decisión sobre el lenguaje y el entorno de Machine Learning

Para el desarrollo del modelo de inteligencia artificial encargado de la clasificación automática de las imágenes, se evaluaron distintas alternativas de lenguajes y entornos de *machine learning*.

Se optó por utilizar Python como lenguaje de programación principal debido a su posición de liderazgo en el campo del aprendizaje automático y la ciencia de datos (Python, s.f.). Es, además, ampliamente adoptado por la comunidad científica y de ingeniería debido a su sintaxis sencilla, su ecosistema de librerías especializadas y su constante actualización con los últimos avances en inteligencia artificial.

Dentro del ecosistema de Python, se seleccionó TensorFlow como framework principal para la implementación y entrenamiento del modelo de red neuronal (TensorFlow, s.f.), con base en los siguientes factores clave:

- **Amplia documentación y soporte:** cuenta con una extensa documentación oficial, numerosos tutoriales, ejemplos de implementación y una comunidad de usuarios muy activa, facilitando la resolución de problemas y acelerando el proceso de desarrollo.
- **Acceso a modelos pre entrenados:** el framework ofrece una biblioteca robusta de modelos pre-entrenados, lo que permite aplicar estrategias de transferencia de aprendizaje de forma sencilla, optimizando los tiempos de entrenamiento y mejorando la precisión del modelo final.
- **Compatibilidad y escalabilidad:** permite integrar modelos en distintos entornos, incluyendo la ejecución local de scripts, adaptándose perfectamente a la arquitectura elegida para el sistema.

La combinación de Python y TensorFlow permitió desarrollar un modelo de clasificación confiable, fácilmente integrable con el resto de la aplicación, y alineado con los estándares actuales de la industria.

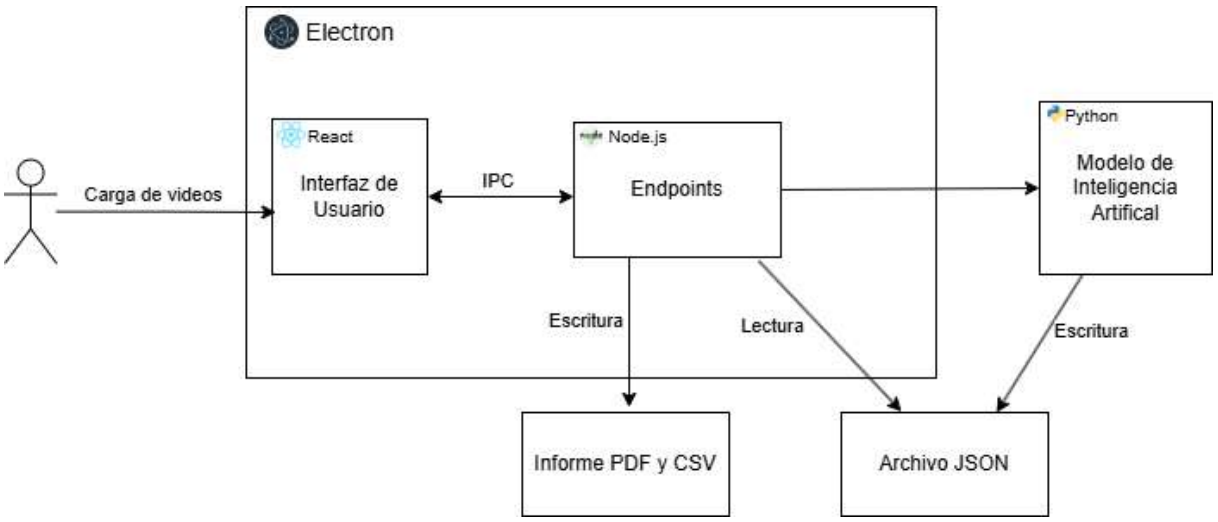

Figura 6. Diagrama de arquitectura del sistema.

10. Desarrollo del sistema

En el planteo inicial del proyecto, se había previsto comenzar las tareas de desarrollo enfocándose en el entrenamiento y la prueba del modelo de inteligencia artificial encargado de realizar el análisis automático de las ecografías pulmonares. La idea era construir primero el núcleo de procesamiento del sistema, para luego avanzar con la implementación de las demás funcionalidades.

Sin embargo, al iniciar esta etapa, surgieron dos problemáticas principales que llevaron a replantear la estrategia de trabajo. En primer lugar, la experiencia previa sobre los algoritmos de aprendizaje automático no abarcaba las técnicas específicas requeridas para el entrenamiento efectivo de redes neuronales convolucionales aplicadas a imágenes médicas. A pesar de la familiaridad teórica general con machine learning, la práctica concreta de seleccionar arquitecturas, configurar hiper parámetros, realizar preprocesamiento de datos y evaluar métricas específicas resultó ser más compleja de lo previsto.

En segundo lugar, a pesar de los esfuerzos, no se lograron obtener resultados concretos durante las primeras pruebas de entrenamiento. La combinación de un conjunto de datos limitado, la dificultad para ajustar correctamente los modelos y el tiempo de iteración elevado en los entrenamientos hizo que el progreso en este área fuera más lento de lo esperado.

Frente a estas dificultades, y para no detener el avance global del proyecto, se decidió posponer temporalmente el entrenamiento efectivo del modelo y priorizar el avance en otras

áreas del desarrollo. Se enfocó el trabajo en el diseño de las interfaces de usuario y en la implementación de las funcionalidades principales del sistema.

Para ello, se diseñaron *mocks* que simularon el comportamiento esperado del modelo de inteligencia artificial. Así, se generaron salidas de clasificación consistentes, permitiendo avanzar en:

- El flujo de carga y recorte de videos.
- La estructura de análisis frame a frame.
- El cálculo de métricas basado en resultados simulados.
- La construcción automática de informes.

Esta decisión estratégica permitió consolidar una base funcional robusta, asegurando que el desarrollo del sistema pudiera continuar de manera ordenada y sostenida, a pesar de las dificultades iniciales en el área de machine learning.

10.1 Diseño de interfaces gráficas de usuario

Para la definición visual de la aplicación, se elaboraron prototipos de las interfaces gráficas utilizando la plataforma Figma. Se construyeron representaciones detalladas de las pantallas, contemplando la disposición de los elementos interactivos y los flujos de navegación entre las distintas funcionalidades del sistema.

Una vez confeccionados los prototipos, se procedió a su validación directa con usuarios finales, representados en este caso por referentes funcionales del proyecto. Durante esta instancia se verificó que las funcionalidades planteadas fueran adecuadas a las necesidades reales, permitiendo detectar ajustes necesarios en los flujos de trabajo y en la disposición de algunos componentes.

La validación temprana de los prototipos garantizó que, al momento de la implementación, las interfaces reflejaran de manera precisa los requerimientos operativos planteados, optimizando así el desarrollo posterior del sistema.

Figura 7. Ventana inicial del programa

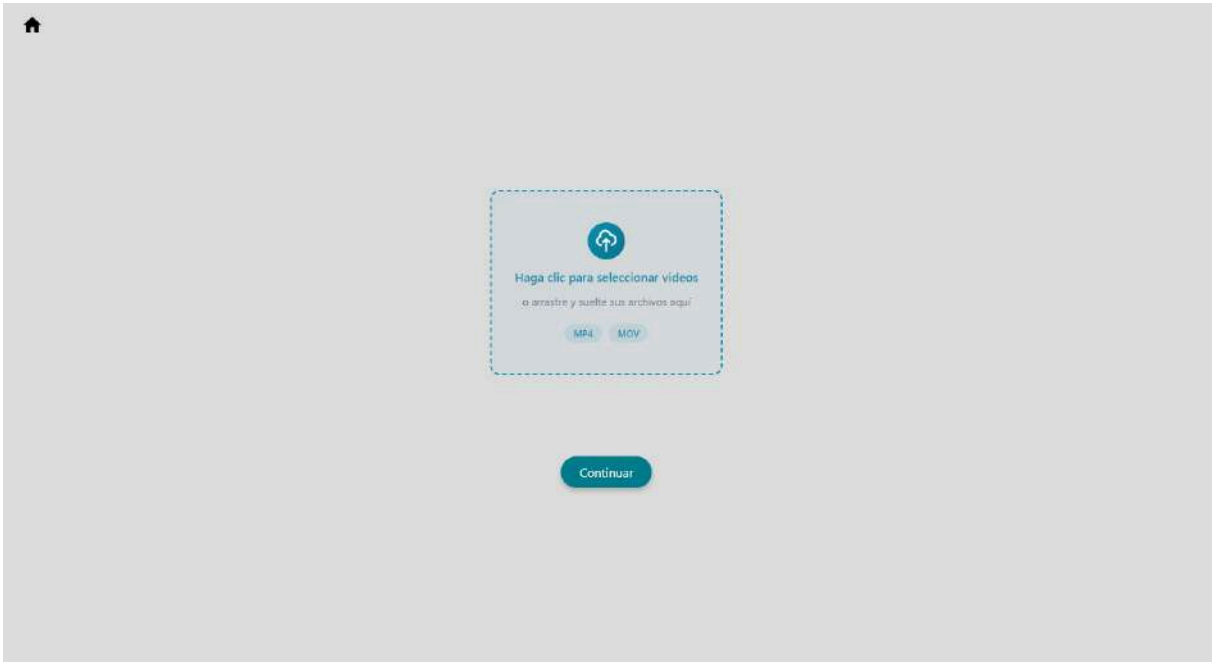

Figura 8. Ventana de carga y recorte de videos (sin videos cargados).

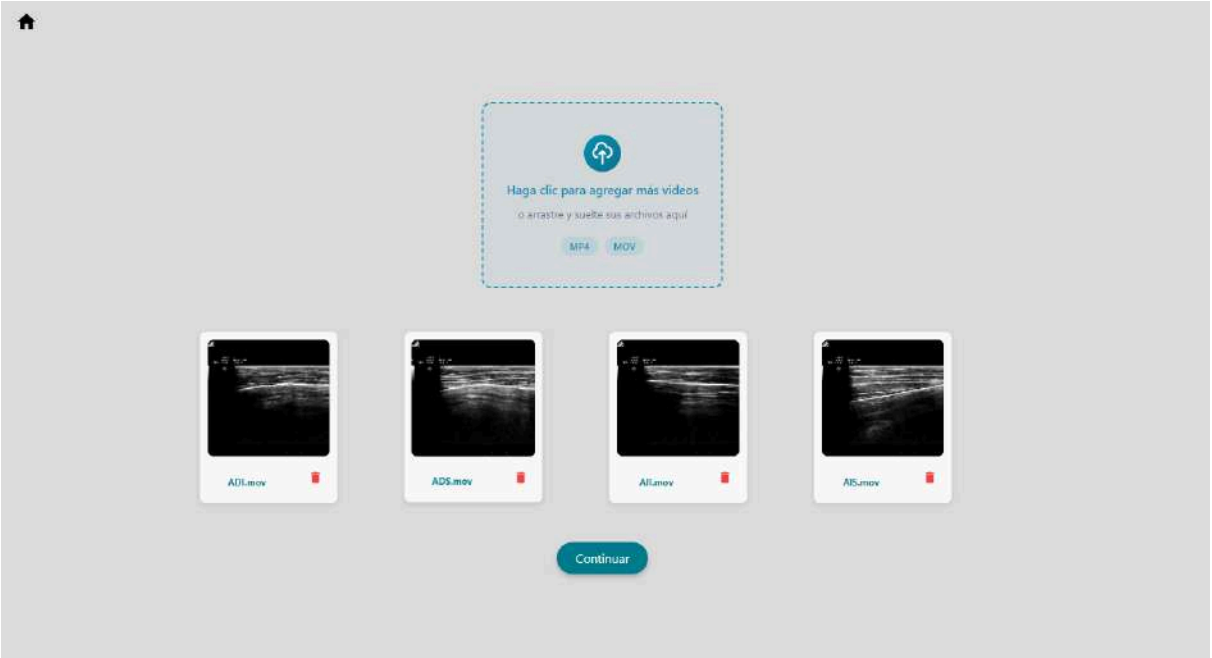

Figura 9. Ventana de carga y recorte de videos

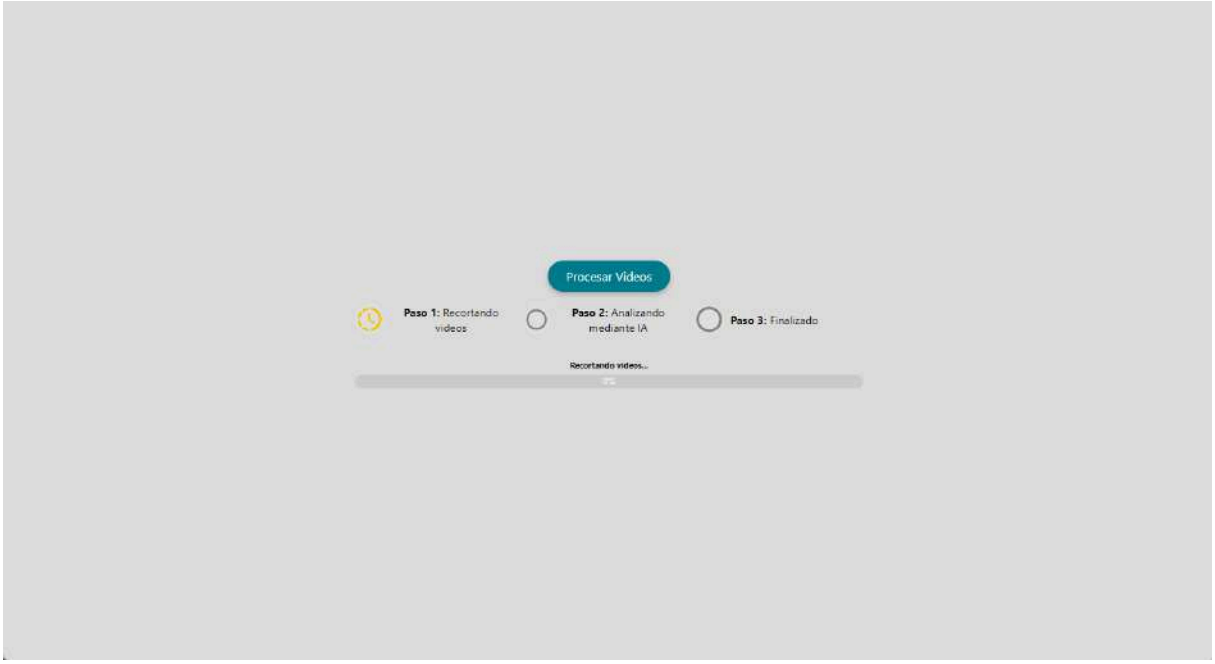

Figura 10. Ventana de procesamiento de videos.

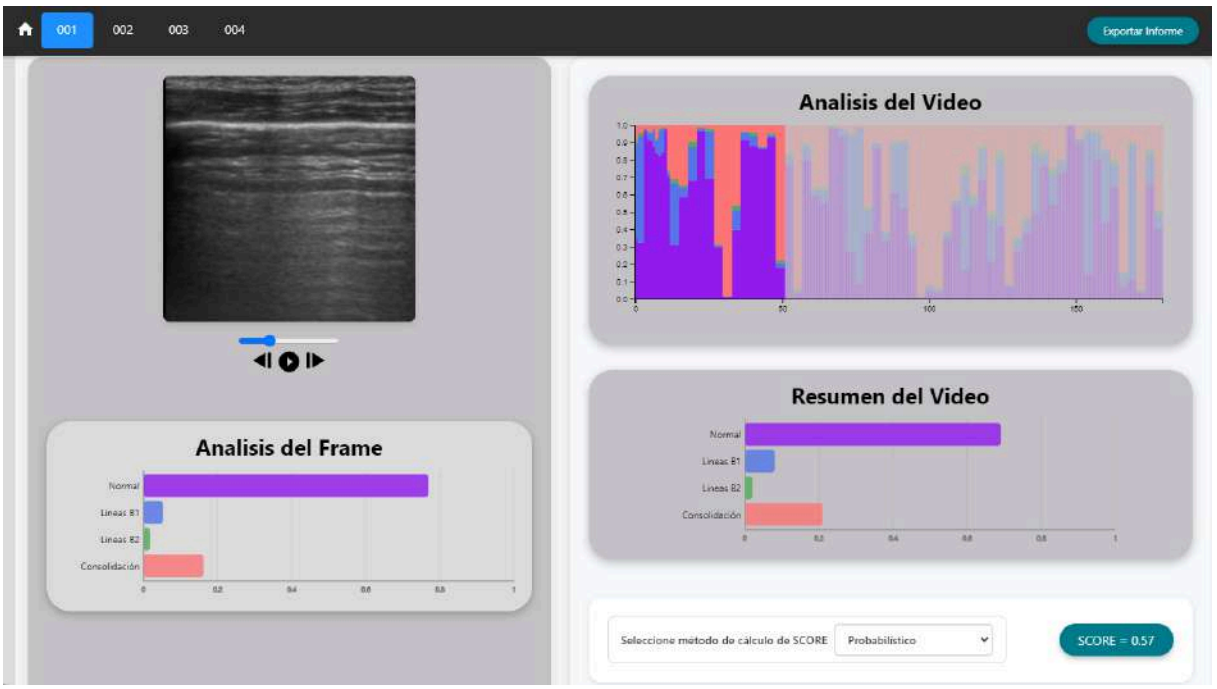

Figura 11. Ventana de videos procesados y gráficos interactivos

Además de las ventanas principales, se generaron los prototipos para las funcionalidades que permiten modificar parámetros al sistema, tanto en la parte visual como en el modelo a utilizar para el procesamiento de ecografías en caso de querer utilizar uno propio.

Figura 12. Ventana de elección del modelo a utilizar para procesar los videos.

Figura 13. Ventana de generación del informe en formato PDF

Figura 14. Ventana de elección de paleta de colores para la visualización gráfica

También se hicieron informes de ejemplo que luego serán los que permita generar el sistema en base a múltiples ecografías analizadas. A continuación se muestra un ejemplo visual:

Figura 15. Visualización de informe generado en formato PDF

10.2 Frontend

El *frontend* de la aplicación fue desarrollado utilizando React, uno de los *frameworks* más extendidos para el desarrollo de interfaces dinámicas y escalables. React permitió construir una interfaz modular, eficiente en la gestión de cambios de estado y adaptada a la interacción continua con datos provenientes del análisis de ecografías pulmonares.

El enfoque adoptado para el desarrollo de la interfaz gráfica se basó en la composición de componentes reutilizables, que permiten dividir la complejidad de la aplicación en partes independientes y fácilmente mantenibles. Cada funcionalidad visual, ya sea de visualización de datos, interacción con el usuario o gestión de archivos multimedia, se encapsuló dentro de un componente específico, siguiendo un principio de alta cohesión y bajo acoplamiento.

El estado de la aplicación, así como el de los distintos elementos multimedia y resultados de procesamiento, se gestionó utilizando patrón de observador, mediante la librería MobX. De esta forma, los componentes de la interfaz reaccionan automáticamente a los cambios de estado definidos en contenedores (*stores*), garantizando una sincronización continua entre los datos y la representación visual.

La arquitectura del frontend aplicó los siguientes patrones de diseño de software:

- **Composición de Componentes:**
  Cada unidad funcional de la interfaz fue implementada como un componente independiente, lo que facilita su prueba, extensión y reutilización en distintas partes del sistema.

- **Observador-Reacción (Observer Pattern):**
  A través de MobX, los componentes observan los cambios en los contenedores de información, actualizando automáticamente su representación cuando se produce un cambio. Las *stores* definidas en la aplicación permiten organizar y centralizar el estado de distintos elementos clave del sistema. Se gestionan eventos como la carga y reproducción de videos, el avance de etapas en el flujo de análisis y el control de ejecuciones en lote. Esta organización favorece una estructura clara y mantenible, donde cada componente de la interfaz puede reaccionar automáticamente a los cambios relevantes, asegurando la coherencia entre los datos y su visualización.

10.3 Backend

La implementación del *backend* del sistema estuvo enfocada en la gestión centralizada de la lógica de negocio, la coordinación de las distintas operaciones del sistema, y la comunicación eficiente entre la interfaz gráfica y el modelo de inteligencia artificial.

El *backend* fue desarrollado utilizando Node.js, en combinación con Electron, permitiendo así estructurar una aplicación de escritorio de arquitectura híbrida que integra procesamiento de datos, gestión de archivos y ejecución de procesos externos de forma local.

10.3.1 Funcionalidades principales

Entre las funcionalidades principales que gestiona el *backend* se destacan:

- Carga y procesamiento de videos de ecografías pulmonares: gestión de archivos, recorte automático del área ecográfica útil para eliminar metadatos del equipo de adquisición (configuración, versión del ecógrafo, parámetros de adquisición) utilizando una librería específica (FFMPEG), subdivisión del video en fotogramas, and almacenamiento de los resultados en directorios estructurados.
- Ejecución del modelo de inteligencia artificial: coordinación de la ejecución de *scripts* de Python mediante comunicación entre procesos, enviando los parámetros de entrada necesarios, recibiendo información relacionada al estado del proceso, y recuperando los resultados de análisis en formato JSON.
- Cálculo de métricas: procesamiento posterior de los resultados de clasificación para calcular métricas de aireación pulmonar frame a frame y de manera global.
- Generación de informes: creación de archivos CSV y PDF a partir de los resultados de análisis, permitiendo su exportación y almacenamiento.
- Gestión de modelos de IA: registro, carga y selección de diferentes modelos de machine learning para utilizar en el análisis de ecografías.

Las características implementadas en el sistema se comunican con la lógica del backend mediante canales de comunicación interna (IPC, por sus siglas en inglés). Esto permite que la interfaz gráfica desarrollada en React pueda enviar y recibir información de manera sencilla y asincrónica, garantizando una interacción fluida entre la vista y el núcleo de la aplicación. En el Anexo I se detallan las interfaces expuestas por el backend..

A continuación, se adjunta el diagrama de componentes del sistema:

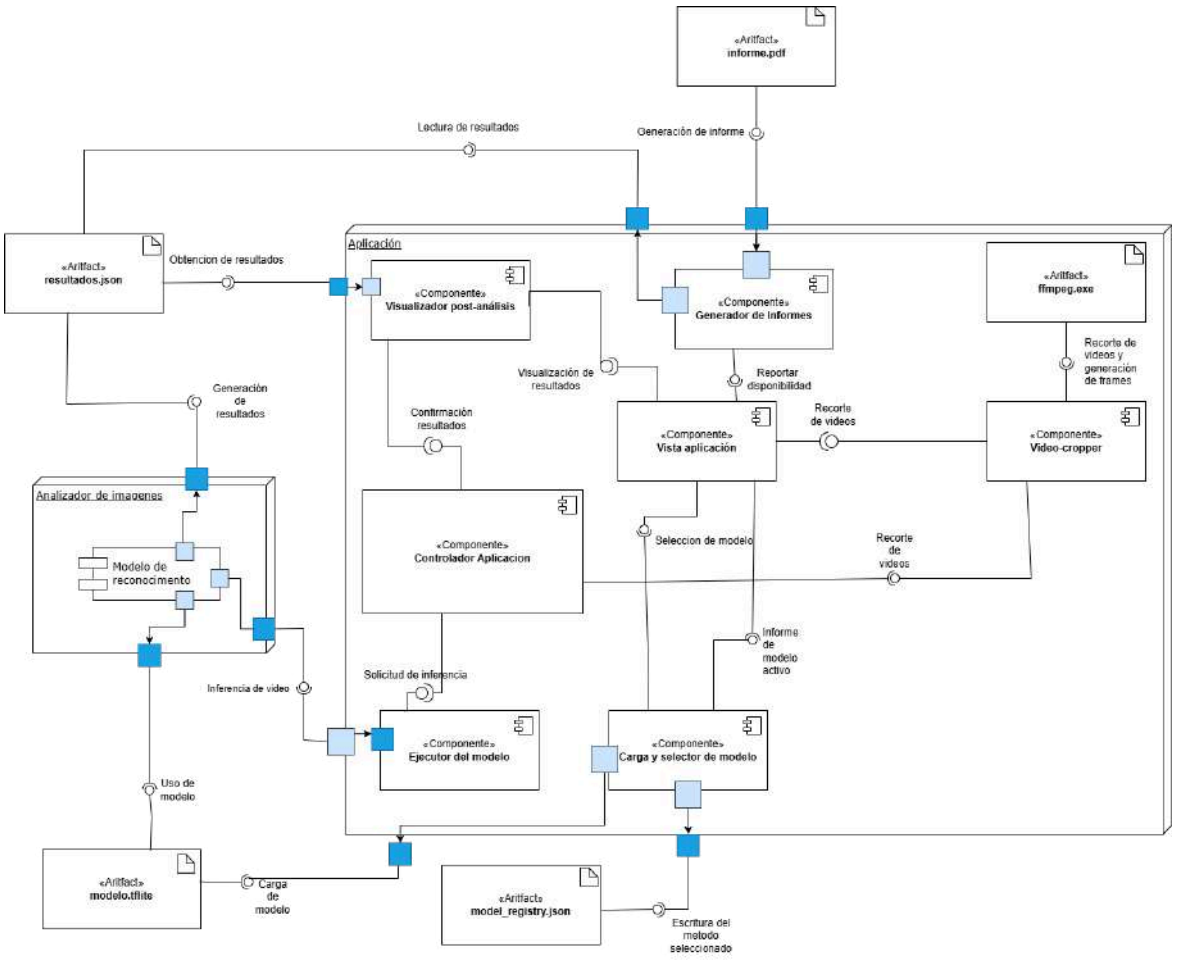

Figura 16. Diagrama de componentes

10.3.2 Patrones de diseño aplicados

Durante la implementación del *backend* se aplicaron distintos patrones de diseño de *software* para garantizar modularidad, mantenimiento y escalabilidad:

- **Separación de responsabilidades**:
  Cada bloque funcional del sistema (gestión de archivos, ejecución de scripts, procesamiento de resultados, comunicación IPC) está organizado de manera independiente, evitando acoplamientos innecesarios.

- **Patrón Proxy para ejecución de procesos externos**:
  El backend actúa como un intermediario ("proxy") entre la interfaz gráfica y el script Python que implementa el modelo de IA, aislando al frontend de la complejidad de ejecución de procesos del sistema operativo.

- **Patrón de Orquestador**:
  El backend centraliza la coordinación de múltiples tareas (recorte de video, análisis, cálculo de métricas, generación de reportes) asegurando que se ejecuten en el orden adecuado y manejando estados intermedios.

- **Uso de promesas y asincronía**:
  Todas las operaciones del backend que involucran entrada/salida (IO) o procesos externos son manejadas de manera asíncrona mediante Promise y async/await, permitiendo un flujo de ejecución no bloqueante y eficiente.

10.4 Comunicación entre Node.js y Python

La integración del modelo de inteligencia artificial con la aplicación principal desarrollada en Node.js se resolvió mediante una combinación de herramientas provistas por el entorno de ejecución y mecanismos de intercambio de información basados en archivos. La ejecución del script de Python se realizó a través de la creación de un proceso separado utilizando herramientas del entorno, lo que permitió que ambos programas pudieran ejecutarse de forma paralela e intercambiar información sin bloquear el flujo principal de la aplicación. La coordinación general de la comunicación entre procesos se manejó utilizando los canales internos de Electron mediante el módulo de *ipcMain* (framework para comunicación entre procesos), garantizando una sincronización fluida con la interfaz de usuario.

El intercambio de información estructurada entre ambos entornos se realizó a través de un archivo en formato JSON. Se utilizaron archivos para especificar el modelo de inteligencia artificial seleccionado y para recibir los resultados generados tras el procesamiento. De esta manera, ambos procesos podrían acceder y modificar un recurso compartido en el sistema de archivos, sin necesidad de establecer una conexión en tiempo real entre ellos.

El flujo completo del proceso se organizó en tres etapas principales:

1. El backend de Node.js invoca los métodos para iniciar la tarea programada en Python, enviando como parámetros el modelo a utilizar, los directorios de entrada (es decir, la ubicación de las imágenes de las ecografías) y la ubicación esperada del archivo de salida.

2. El programa procesa los frames de video y, al finalizar, escribe en la ubicación destino un archivo en formato JSON con los resultados generados.

3. Finalmente, el backend accede al archivo, interpreta los datos contenidos y los utiliza para calcular métricas, enviar la información al frontend y generar informes exportables.

Este esquema de comunicación indirecta ofrece una separación clara de responsabilidades entre los componentes, facilita el mantenimiento del sistema y permite realizar modificaciones sobre el script de Python sin afectar la lógica general de la aplicación, siempre que se mantenga el formato de intercambio definido.

El formato específico del archivo JSON utilizado se encuentra documentado en el Anexo II del presente informe.

10.5 Visualizaciones y cálculos de Probabilidades pulmonares

Se incorporaron visualizaciones interactivas y cálculos estadísticos avanzados con el fin de brindar un análisis dinámico, cuantitativo y comprensible de los estados pulmonares, facilitando su uso en contextos clínicos de monitoreo continuo o de evaluación diagnóstica. El diseño de estos componentes estuvo enfocado en representar con precisión tanto la variabilidad respiratoria como la transición entre distintos patrones ecográficos.

10.5.1 Estimación de probabilidades de pertenencia por frame

Cada imagen individual del video se procesa de forma independiente mediante una función de inferencia que devuelve un vector de probabilidades asociado a las cuatro clases pulmonares definidas por el sistema LUS Score. Dichas clases son:

- Clase 0: Patrón Normal, asociado a la visualización de líneas A.
- Clase 1: B1, correspondiente a pérdida leve o moderada de aireación (líneas B separadas).
- Clase 2: B2, asociada a pérdida severa de aireación (líneas B coalescentes).
- Clase 3: Consolidación, caracterizada por la pérdida completa de aireación y apariencia sólida en el parénquima pulmonar.

El resultado del procesamiento de cada frame es un vector de la forma:

$$\overline{p} = [p_{0,t}, p_{1,t}, p_{2,t}, p_{3,t}]$$

donde cada componente $p_{i,t}$ representa la probabilidad estimada de que el frame t pertenezca a la clase i. Los valores están normalizados, de forma tal que su suma es igual a 1. Este vector probabilístico se convierte en la base para los análisis temporales y los gráficos generados.

10.5.2 Gráfico de probabilidades instantáneas

Con los vectores de probabilidad por frame, se genera un gráfico de evolución temporal en donde el eje X representa la progresión del tiempo en términos de frames, y el eje Y representa la probabilidad correspondiente a cada clase. Se emplean líneas diferenciadas o áreas apiladas para cada clase, lo que permite observar visualmente la evolución relativa de los patrones pulmonares a lo largo del video. Esta representación facilita la identificación de:

- Períodos de estabilidad respiratoria con dominio de una sola clase.
- Fluctuaciones que pueden asociarse a ruido de imagen o a ciclos respiratorios normales.
- Transiciones bruscas entre patrones, que podrían estar vinculadas a eventos clínicos específicos o cambios en la condición pulmonar.

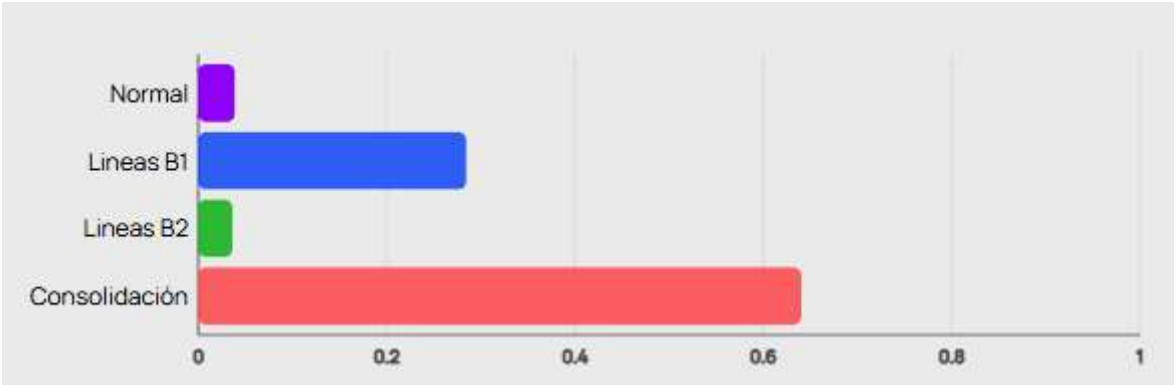

Figura 17. Gráfico de resultados sobre el frame actual.

10.5.3 Timeline de probabilidades apiladas

Para brindar una visualización de contexto global y facilitar la lectura continua, se incorpora un *timeline* apilado, donde cada columna representa un frame y está segmentada verticalmente de acuerdo a la distribución de probabilidades entre las clases. Esto permite observar zonas con alta certeza diagnóstica (una clase dominante) frente a zonas de incertidumbre (probabilidades balanceadas). Además, esta vista resulta especialmente útil para comparar fases respiratorias o efectos de intervenciones clínicas (PEEP).

Sigue la misma estructura del gráfico anterior, pero brinda un panorama más amplio del estado pulmonar, permitiendo detectar los distintos momentos del estudio en los que ocurrieron anomalías o hechos clínicamente relevantes. A su vez, se incorporó un recurso visual dinámico que marca el avance del video: a medida que el usuario reproduce o recorre el estudio, el gráfico diferencia las zonas ya visualizadas mediante un sombreado más opaco, mientras que las secciones aún no exploradas permanecen translúcidas. Esta distinción permite al personal médico mantener un seguimiento claro del progreso del estudio realizado y contextualizar temporalmente los eventos observados.

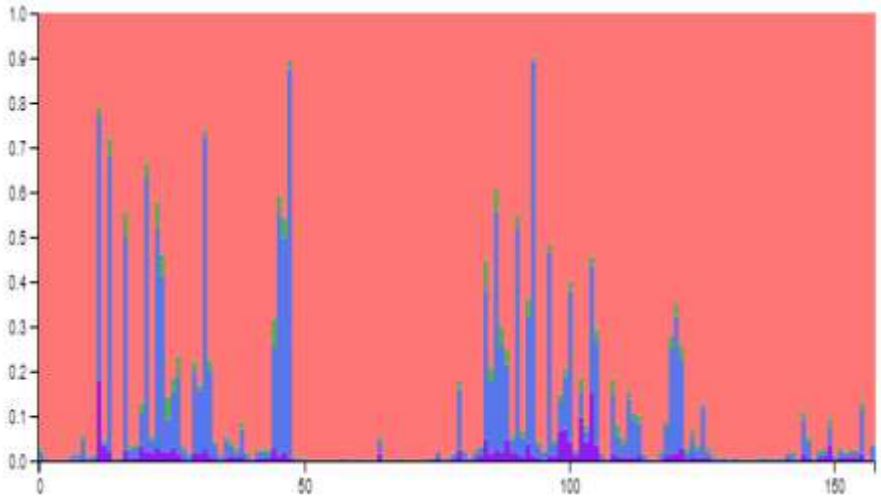

Figura 18 Gráfico “timeline” de probabilidades apiladas.

10.5.4 Cálculo de probabilidades de las clases globales del video

Además de las visualizaciones cuadro a cuadro, se implementa una forma de resumir cuantitativamente el comportamiento general del video. Para ello, se calcula una probabilidad global por clase, utilizando una estrategia estadística robusta. Se consideraron varias opciones, incluyendo el promedio aritmético simple, la aplicación de la función *softmax* y la mediana normalizada. Tras evaluar sus implicancias, se optó por la última opción.

Pasos aplicados en el cálculo:

1. Se computan todos los valores $p_{i,t}$ para cada clase *i* a lo largo de los cuadros *t* del video.
2. Se calcula la mediana m_i de las probabilidades por cada clase *i*, a lo largo de todos los cuadros.

3. Se normalizan las medianas para obtener los valores de probabilidad sumarizados, que mantienen la propiedad de sumar 1:

$$P_i = \frac{m_i}{\sum_{j=0}^{3} m_j}$$

Este vector $\overline{P}$ proporciona una representación probabilística de los patrones pulmonares que ocurrieron durante la totalidad del video.

Durante el desarrollo del proyecto, se consideró el uso de softmax como cálculo alternativo:

$$P_i^{softmax} = \frac{e^{m_i}}{\sum_{j=0}^{3} e^{m_j}}$$

Este método produjo resultados no apropiados, pues asigna probabilidades no nulas a clases que no habían estado presentes en ningún frame. Esta característica, aunque útil en tareas de clasificación general, resulta contraproducente en este contexto donde la ausencia de un patrón debe ser representada explícitamente. En el Anexo III se pueden observar cálculos realizados y la comparativa entre los métodos.

Las ventajas de la mediana normalizada incluyen:

- Capacidad de representar clases ausentes con una probabilidad de 0.
- Menor sensibilidad a valores atípicos o picos espurios en la secuencia de probabilidades.
- Mayor fidelidad en la representación del comportamiento general.

Estos resultados son preliminares. La validez clínica de esta metodología deberá ser comprobada en estudios posteriores, comparando los resultados con evaluaciones realizadas por expertos humanos en un volumen de estudios muy superior al utilizado en este proyecto.

Para la representación gráfica de esta métrica se utiliza un gráfico de barras horizontales, al igual que para la visualización del frame actual, manteniendo la asociación entre los colores y estados pulmonares, con el objetivo de preservar la coherencia visual entre todos los resultados mostrados.

10.5.5 Cálculo de Scores de Aireación

A partir de la serie temporal de probabilidades por clase, se definieron dos métricas adicionales que cuantifican la gravedad del estado pulmonar en términos numéricos:

1. **Score Promedio Ponderado**
   Se obtiene calculando el valor esperado del score para cada frame y luego promediando a lo largo del video:

$$Score_{prom} = \frac{1}{T}\sum_{t=1}^{T}\sum_{i=0}^{3} p_{i,t}\, s_i \,,$$

   donde s_i representa el valor numérico asignado a cada clase (0 para Normal, hasta 3 para Consolidación) y *T* es la cantidad de frames.

2. **Score por Clase Dominante:**
   Considera una clasificación no probabilística por frame. Se asigna a cada frame la clase con mayor probabilidad, y se promedian sus valores:

$$Score_{mayor} = \frac{1}{T}\sum_{t=1}^{T} s_{arg\,max_i(p_{i,t})}.$$

Ambas metodologías permiten realizar comparaciones entre videos, evaluar la evolución de un paciente o cuantificar el efecto de una intervención o maniobra (por ejemplo, el cambio en el valor de parámetros de un sistema de asistencia respiratoria). Ambas estrategias están disponibles para ser utilizadas en el sistema, permitiendo a los usuarios elegir la que mejor se ajuste al estudio en revisión.

10.6 Desarrollo del modelo de Inteligencia Artificial

Una vez definido el alcance del proyecto y los requerimientos técnicos, se dio inicio a la etapa de desarrollo del modelo de Inteligencia Artificial responsable del análisis automático de imágenes de ecografías.

Antes de iniciar el proceso de implementación, se definieron cuatro etapas fundamentales para organizar el trabajo y abordar el problema de manera estructurada:

- Obtención del conjunto de datos
- Análisis preliminar del conjunto de datos
- Entrenamiento supervisado
- Conclusiones

Esta división permitió gestionar el desarrollo de forma eficiente, anticipar posibles desafíos y adaptar el enfoque en función de los hallazgos durante cada fase.

10.6.1 Obtención del conjunto de datos

El laboratorio de bioingeniería contaba con 1001 imágenes de ecografías de pulmón en formato JPG, previamente clasificadas por el Dr. Gerardo Tusman, la Dra. Cecilia Acosta (médicos anestesiólogos) y el Dr. Gustavo Meschino. También, tenían en su poder más de 70 videos de ecografías de pulmón, pero sin etiquetas y no estaban divididos en fotogramas, por lo que se requería un preprocesamiento para la etapa de entrenamiento. Así, el conjunto de datos quedó compuesto únicamente por las imágenes etiquetadas. A continuación, se detalla un ejemplar:

Figura 19 Frame extraído de una ecografía de pulmón

Como se puede observar, la imagen contiene información innecesaria para el entrenamiento de la red neuronal. Esto se repetía en todo el dataset, por lo que se requirió realizar el

recorte de la región de interés de cada imagen, con el fin de optimizar el entrenamiento de la red neuronal.

La distribución de clases quedó de la siguiente manera:

- **Clase B1.** 234 imágenes.
- **Clase B2.** 222 imágenes.
- **Clase C.** 427 imágenes.
- **Clase Normal.** 118 imágenes.

Se evidencia un desbalance en el conjunto. Esto representó un factor fundamental a tener en cuenta a la hora del entrenamiento, pues se considera el riesgo de generar un sesgo hacia la clase más predominante (en este caso, la C). Por otro lado, se tratan de datos reales, lo que otorga riqueza a este trabajo.

10.6.2 Análisis preliminar de pureza mediante clustering

Como pasos iniciales de esta etapa, se llevó a cabo una investigación con el fin de evaluar la pureza del conjunto de datos. Esta fase preliminar consistió en aplicar algoritmos de *clustering* sobre las *features* extraídas automáticamente por distintas arquitecturas de redes convolucionales preentrenadas (VGG16, MobileNet, EfficientNet, ConvNeXt, NasNet, InceptionResNet). Esto es, mediante el uso de filtros y de las primeras capas convolucionales de las redes, obtener las características representativas de cada imagen, y, en base a ellas, ejecutar algoritmos de agrupamiento. El objetivo era evaluar la calidad del conjunto de datos, y respaldar nuestra idea principal de que para el proyecto era necesario acudir a métodos supervisados.

La motivación principal fue determinar en qué medida las imágenes correspondientes a las diferentes clases (Normal, B1, B2, Consolidación) se organizaban naturalmente en agrupamientos separados según sus características internas.

En primer lugar, cada imagen fue redimensionada a la resolución requerida por cada arquitectura y luego normalizada, ajustando los valores de los píxeles según el esquema de preprocesamiento utilizado durante el entrenamiento original de cada red. De esta manera, se asegura la coherencia entre las imágenes procesadas y el espacio de entrada esperado por cada modelo.

Posteriormente, se extrajeron vectores de características (*feature extraction*). Estas representaciones de alto nivel reflejan patrones visuales relevantes (texturas, bordes,

artefactos) aprendidos en ImageNet y permiten caracterizar cada imagen más allá de los píxeles originales.

A partir de los vectores obtenidos se aplicaron algoritmos de clustering. Esta es una aplicación de aprendizaje no-supervisado que busca establecer relaciones entre los datos del conjunto y agruparlos según su semejanza. Se decidió aplicar únicamente KMeans en esta etapa de investigación preliminar al desarrollo del modelo de Deep Learning. En cuanto a los parámetros, el valor de la cantidad de clusters se configuró en cuatro, correspondientes al número de clases del conjunto de datos (*Normal, B1, B2 y C*).

Para visualizar la estructura en un espacio bidimensional se aplicó una reducción de dimensionalidad mediante PCA (Análisis de Componentes Principales).

Finalmente, se construyó una matriz de confusión entre clusters y etiquetas reales y se calculó la pureza, métrica que indica el grado de homogeneidad de cada cluster en relación a la clase predominante. Este análisis permitió examinar en qué medida las distintas arquitecturas pre entrenadas logran separar las clases de forma no supervisada, brindando información valiosa sobre la calidad del dataset y el solapamiento entre categorías clínicas.

En el Anexo IV se visualizan los resultados obtenidos por los modelos pre-entrenados mostrando gráficamente con colores los clusters conformados y con formas la categoría real de cada imagen. Para complementar estos gráficos, se representó la matriz de confusión. Los resultados indicaron que la estructura interna del dataset no permitía una separación clara de las clases únicamente mediante agrupamiento, alcanzando niveles de pureza inferiores al 55% en todos los casos. Esto reforzó la decisión de avanzar con un enfoque supervisado con transfer learning, donde la arquitectura del modelo pueda aprender representaciones más discriminativas a partir de los datos etiquetados.

A continuación, se resumen los valores de pureza obtenidos por cada arquitectura:

Tabla 2. Pureza por arquitectura en análisis no supervisado. Valores de Purity obtenidos al aplicar clustering sobre las features extraídas por cada red pre entrenada.

Arquitectura	Purity (%)
ConvNeXt	53.15
MobileNet	52.35
InceptionResNet	50.15
NasNet	49.75
VGG16	49.15

EfficientNet	44.65

Las matrices de confusión mostraron una fuerte superposición entre clases clínicas, especialmente entre B1, B2 y Consolidación. A modo de ejemplo, la matriz obtenida para MobileNet fue la siguiente:

Tabla 3. Matriz de confusión para red MobileNet.

Clase	Normal	B1	B2	Consolidación
Normal	**9**	16	9	22
B1	106	**106**	113	370
B2	0	49	**69**	20
Consolidación	0	63	31	**15**

Estas observaciones evidenciaron que el problema de clasificación requería el uso de métodos supervisados y arquitecturas entrenables, ya que las representaciones latentes no eran suficientemente discriminativas por sí solas. Este análisis preliminar permitió validar la calidad del dataset desde una perspectiva no supervisada, y contribuyó a establecer una base para las decisiones metodológicas adoptadas en las etapas posteriores.

La visualización de los agrupamientos obtenidos, mediante reducción de dimensionalidad mediante Análisis de Componentes Principales, tomando solo las dos primeras componentes, se encuentra en el Anexo IV.

10.7 Introducción al entrenamiento supervisado

Una vez alcanzado un grado de madurez en el desarrollo del sistema, se retomó el entrenamiento efectivo del modelo de inteligencia artificial, con un mayor conocimiento del dominio y un sistema funcional sobre el cual realizar pruebas Este enfoque escalonado resultó ser acertado, ya que para ese momento se contaba con una comprensión más profunda tanto del problema clínico como de las necesidades prácticas del sistema.

Esta fase tuvo como objetivo principal seleccionar la arquitectura de inteligencia artificial apropiada para el problema de clasificación automática de las imágenes de ecografías de pulmón. Para ello fue necesario definir el paradigma de aprendizaje y la arquitectura específica a implementar. El desarrollo del modelo se fundamenta en el paradigma de aprendizaje supervisado dentro del espectro del Deep Learning, donde el sistema aprende a partir de un conjunto de datos etiquetados para establecer una función de mapeo que relaciona las imágenes con sus correspondientes clasificaciones diagnósticas.

En primer lugar, se definió que se iba a trabajar con mecanismos de Redes Neuronales Convolucionales (CNN), arquitectura especializada para el procesamiento de imágenes que incorpora principios de conectividad local, compartición de pesos e invariancia traslacional, características fundamentales para la extracción automática de patrones visuales en imágenes médicas.

La selección de la arquitectura específica requirió la evaluación de múltiples factores críticos para aplicaciones médicas: eficiencia computacional, robustez al sobreajuste en conjuntos de datos limitados, capacidad de transferencia desde dominios generales, y viabilidad de deployment en entornos con restricciones de hardware.

En relación a la selección arquitectural, se adoptó MobileNet como backbone convolucional fundamentándose en la evidencia empírica acumulada por el Laboratorio de Bioingeniería y en los resultados de las evaluaciones comparativas realizadas, donde esta arquitectura demostró el mejor desempeño en términos de extracción de características discriminativas sin requerimiento de ajuste fino específico al dominio. A pesar de que MobileNet fue originalmente diseñado para clasificación de imágenes RGB mediante *depthwise separable convolutions* (Howard et al., 2017), su arquitectura presenta propiedades ventajosas para este caso, particularmente en lo referente a la eficiencia computacional bajo restricciones de recursos hardware.

La implementación de convoluciones separables por profundidad permite una reducción sustancial del número de parámetros entrenables y operaciones de punto flotante en comparación con arquitecturas convolucionales tradicionales, constituyendo un factor crítico para la viabilidad del entrenamiento en infraestructuras con limitaciones computacionales. Esta decisión se vio respaldada por los resultados del análisis de clustering, donde MobileNet alcanzó el segundo mejor índice de pureza (52.35%), indicando una capacidad levemente superior para generar representaciones latentes más coherentes con la estructura semántica del problema de clasificación planteado.

Se implementaron y evaluaron distintas variantes arquitecturales de redes neuronales convolucionales pre-entrenadas, todas cargadas con pesos provenientes de ImageNet. Se optó por una estrategia de *transfer learning*, incorporando una estructura de clasificación personalizada (ver Anexo V) adaptada específicamente para la discriminación de los cuatro estados diagnósticos definidos en el proyecto.

El modelo final alcanzó un total de 5.3 millones de parámetros con 3.2 millones entrenables, mientras que el resto, correspondientes a la base convolucional, se mantuvieron fijos

durante la primera etapa de entrenamiento. Posteriormente, se experimentó con la habilitación progresiva de capas entrenables en la base para mejorar la adaptación al dominio.

10.7.1 Hardware de entrenamiento

El entrenamiento de modelos de *deep learning* para clasificación de imágenes presenta requerimientos computacionales significativos que deben ser considerados para garantizar la viabilidad del proyecto. Las redes neuronales convolucionales requieren cantidades sustanciales de memoria de video para el almacenamiento de activaciones, gradientes y parámetros durante el entrenamiento, con requerimientos típicos que superan los 2 GB de VRAM para arquitecturas modernas.

Inicialmente, el entrenamiento se probó en entornos locales. Bajo las condiciones de 800 imágenes de entrenamiento y 200 de validación, organizadas en batches de 32, con 25 épocas sobre MobileNet, el tiempo total de entrenamiento fue de aproximadamente 73 minutos. En contraste, al ejecutar la misma configuración en Google Colab Pro, utilizando GPUs NVIDIA Tesla K80/T4, el tiempo se redujo a alrededor de 15 minutos. Esta diferencia significativa de rendimiento motivó la migración definitiva del proceso de entrenamiento hacia entornos en la nube adquiriendo unidades computacionales.

Las GPUs proporcionan aceleración masiva para aplicaciones de inteligencia artificial, con capacidades de memoria que alcanzan hasta 188 GB en las configuraciones más avanzadas. Esta arquitectura de procesamiento paralelo resulta especialmente eficiente para las operaciones intensivas que forman parte del entrenamiento de redes neuronales convolucionales.

Durante este período se utilizó la versión de python admitida por el entorno, en este caso, la versión 3.11. Por otra parte, también se utilizaron como librerías fundamentales Keras v3.4.1 y Tensorflow v2.17.0

10.7.1 Estrategia de partición de datos y validación

Dado el volumen limitado y la heterogeneidad del dataset, se diseñaron diversas estrategias de división y reutilización de los datos con el objetivo de maximizar la robustez del modelo y evitar el *overfitting*.

En una primera instancia, se aplicó una división básica en dos subconjuntos: el 80% del total de imágenes se destinó a entrenamiento del modelo, mientras que el 20% restante se utilizó

exclusivamente para pruebas. Esta separación se mantuvo constante para asegurar que el conjunto de evaluación reflejara casos no vistos durante el aprendizaje. Sin embargo, los modelos obtenidos bajo esta estrategia no superaron el 80% de precisión sobre el conjunto de validación, lo que indicó que era necesario implementar enfoques más eficaces, considerando la naturaleza clínica del proyecto.

Como consecuencia, se adoptó una estrategia más dinámica para aprovechar mejor las imágenes destinadas a entrenamiento. En lugar de utilizarlas todas simultáneamente, se subdividió el 80% correspondiente a entrenamiento en dos lotes diferenciados, que fueron utilizados en forma intercalada a lo largo de distintas sesiones. Esta modalidad permitió mejorar la generalización del modelo y alcanzar un rendimiento cercano al 95% de certeza sobre las predicciones en el conjunto de validación. Se determinaron ciertos parámetros esenciales para esta etapa: tomar batches de 8 imágenes cada una, y realizar ciclos de 50 épocas, para poder realizar ajustes necesarios en caso de no obtener los resultados esperados.

La validación del rendimiento se realizó utilizando matrices de confusión, con el fin de evaluar la precisión por clase, la sensibilidad y la especificidad del modelo en cada una de las categorías de aireación. Esta métrica resultó ser más informativa que la simple precisión global, ya que reflejaba el verdadero impacto clínico de las predicciones erróneas. Así, fue posible detectar sesgos, por ejemplo, hacia clases mayoritarias como Consolidación, o dificultades particulares para diferenciar B1 y B2, que comparten características morfológicas similares.

Esta modalidad de trabajo incremental, basada en pruebas intercaladas por lotes, contribuyó a evaluar la capacidad de generalización del modelo en subconjuntos clínicamente variados.

10.7.2 Proceso de ajuste y experimentación

Durante el proceso de entrenamiento se llevaron a cabo múltiples pruebas con el objetivo de evaluar distintas configuraciones de la arquitectura de la red neuronal y determinar cuáles aportaban mejoras significativas en términos de precisión y generalización para poder continuar con la evolución de un único modelo. Las pruebas no se limitaron a una única variante del modelo, sino que incluyeron aproximadamente quince experimentos en los que se ajustaron parámetros estructurales y de entrenamiento. En particular, se exploraron las siguientes dimensiones:

- **Capas de clasificación**: se evaluó el impacto de agregar diferentes combinaciones de capas de clasificación luego de las capas convolucionales de cada arquitectura probada. Estas incluyeron:
    - **Feed Forward Fully Connected**: capas densas añadidas al final de la red para aumentar la capacidad de representación. Se variaron distintas capacidades: 512 neuronas, 1024, todas utilizando como función de activación ReLU.
    - **Dropout**: con distintos valores de probabilidad, como mecanismo de regularización para evitar sobreajuste.
    - **Pooling**: variantes de *GlobalAveragePooling* y *GlobalMaxPooling*, como paso previo a la fase de clasificación.
    - **Capa de salida de clasificación**: encargada de mapear las representaciones extraídas hacia las clases definidas en el problema. En este caso siempre se utilizaron 4 neuronas (una para cada clase) y la función softmax para definir las probabilidades de pertenencia.

  El objetivo de esta experimentación fue analizar si la incorporación y configuración de estas capas adicionales permitía incrementar la capacidad del modelo sin introducir sobreajuste ni un aumento innecesario de la complejidad computacional.

- **Función de pérdida y optimizadores**: se compararon configuraciones con distintos algoritmos de optimización (por ejemplo, *Adam* y *SGD*) y funciones de pérdida adecuadas al problema de clasificación multiclase. Se probaron distintos valores de *learning rate* basado en la tasa de aprendizaje de la red; si se notaba que los valores de precisión aumentaban exponencialmente, se disminuía este valor y se realizaban nuevos entrenamientos.

- **Cantidad de parámetros entrenables**: se estudió la relación entre el número de parámetros ajustados y la calidad de los resultados, considerando el compromiso entre complejidad del modelo y recursos de cómputo. En este caso, se comprobó que congelar los parámetros de cada arquitectura y hacer variar los de las capas añadidas, aportaba mejores resultados globales.

De estas pruebas se extrajeron varias conclusiones relevantes. Por un lado, se comprobó que incrementar de manera excesiva el número de capas densas no condujo a una mejora sustancial del rendimiento, mientras que la incorporación de *Dropout* resultó efectiva para mitigar el sobreajuste. Asimismo, se observó que los modelos con menor número de

parámetros entrenables, pero mejor estructurados, alcanzaron desempeños competitivos, justificando la elección de MobileNet como red principal.

Durante este proceso, se almacenaron los modelos de inteligencia artificial en el formato especificado por la librería Tensorflow (extensiones .h5 y .keras), para poder registrar la evolución, y también obtener un flujo iterativo sobre los ajustes a realizar. En el Anexo VI se encuentra una tabla con las distintas arquitecturas utilizadas, y sus resultados.

Si bien no se buscó una optimización orientada únicamente a métricas, el objetivo principal fue obtener un modelo funcional, con el fin de ser integrado al sistema y soportar la variabilidad del conjunto de datos clínicos sin comprometer la estabilidad operativa. Por otro lado, el valor del producto desarrollado está, precisamente, en poder cambiar a futuro el modelo para, conforme avancen las tecnologías y las técnicas, seguir mejorando la clasificación.

10.7.3 Limitaciones del modelo entrenado

Si bien el modelo de clasificación desarrollado logró desempeñarse de manera satisfactoria dentro del entorno experimental y alcanzó métricas aceptables de desempeño, es necesario reconocer ciertas limitaciones inherentes tanto al conjunto de datos como al enfoque metodológico adoptado.

Una de las principales restricciones fue el volumen del dataset, limitado a alrededor de 1000 imágenes. Aunque se aplicaron técnicas de *data augmentation* y se reutilizaron imágenes en lotes intercalados, sigue existiendo el riesgo de que el modelo no haya captado toda la variabilidad morfológica presente en imágenes clínicas reales. La distribución entre clases tampoco fue perfectamente balanceada, y si bien se aplicaron medidas para mitigar este desbalance, pudo haber influido en la sensibilidad específica hacia clases minoritarias. Sin embargo, los resultados obtenidos de clasificar imágenes que no eran parte del conjunto de datos utilizados para el entrenamiento estaban dentro de los esperados por los médicos consultados, lo que también fue tomado como validación experta del modelo de entrenamiento entregado.

Otra limitación significativa fue la naturaleza estática del enfoque adoptado, que se basó exclusivamente en la clasificación de imágenes individuales. No se incorporó información temporal, a pesar de que las secuencias de video podrían aportar una dimensión clínica valiosa. Esta elección fue producto de restricciones prácticas en la disponibilidad del dataset y en la capacidad computacional para entrenar modelos más complejos.

Desde el punto de vista arquitectónico, si bien se exploraron variantes de MobileNet y VGG16, no se realizó una evaluación sistemática de modelos más modernos como EfficientNet, ConvNeXt o Vision Transformers. Tampoco se implementaron estrategias de ensamblado de modelos ni mecanismos de atención, que podrían mejorar la capacidad de interpretación y precisión del sistema al enfocarse dinámicamente en regiones relevantes de la imagen.

En términos de validación, se trabajó con un único conjunto de testeo estático. No se implementaron estrategias como validación cruzada dada la limitación cuantitativa, ni se evaluó el modelo sobre conjuntos externos provenientes de otras fuentes clínicas, lo cual limita la evidencia sobre su capacidad de generalización.

10.8 Preprocesamiento de ecografías

El modelo de red neuronal convolucional desarrollado para el sistema requiere imágenes estáticas como entrada para realizar la clasificación de patrones de aireación pulmonar. Sin embargo, el protocolo clínico de adquisición de datos contempla secuencias de video ecográfico como formato nativo de captura. Esta discrepancia entre el formato de entrada del sistema y el tipo de dato generado durante la práctica clínica requirió el diseño e implementación de un pipeline de preprocesamiento específico.

10.8.1 Extracción y estandarización de frames

El preprocesamiento implementa una estrategia de extracción temporal que convierte las secuencias de video ecográfico en frames individuales, manteniendo la coherencia temporal y la calidad de imagen requerida para el análisis automatizado. Se estableció una frecuencia de muestreo estándar de 25 fps (frames por segundo), valor que garantiza una resolución temporal adecuada para capturar las variaciones dinámicas del patrón pulmonar sin generar redundancia excesiva en los datos procesados.

Los frames extraídos se procesan y almacenan en formato JPEG, preservando la codificación RGB de 3 canales para mantener compatibilidad con la arquitectura de entrada del modelo pre-entrenado.

10.8.2 Gestión de memoria y optimización de recursos

Se implementó una estrategia de gestión de almacenamiento que prioriza la eficiencia operativa del sistema. Los frames procesados se almacenan temporalmente en memoria

RAM durante la fase de análisis, evitando operaciones de escritura a disco innecesarias que podrían impactar el rendimiento en tiempo real. Al finalizar la ejecución del programa, las imágenes temporales son eliminadas automáticamente del sistema de archivos, previniendo la acumulación de datos residuales que podrían comprometer el espacio de almacenamiento disponible en sesiones sucesivas de uso.

11. Desarrollos futuros

El proyecto fue entregado cumpliendo con los requisitos fundamentales y las exigencias planteadas por los demandantes. No obstante, se identificaron posibles mejoras y extensiones que permitirían enriquecer la solución en futuras iteraciones, considerando tanto los avances tecnológicos como la posibilidad de extender el campo de estudio abordado:

- **Incorporación de nuevos dashboards.** Actualmente, la aplicación está limitada gráficamente al análisis del estado de la pleura a partir de imágenes de ecografías. Podrían incorporarse nuevos paneles u opciones de visualización para abordar otras problemáticas relacionadas con los pulmones u otro tipo de ecografías. Sin embargo, esto requeriría la utilización de nuevos modelos de aprendizaje.

- **Entrenamiento de nuevos modelos de inteligencia artificial.** En concordancia con lo anterior, si se desea ampliar o modificar el campo de estudio, será necesario entrenar nuevos modelos de inteligencia artificial. Para obtener la mejor precisión posible, esta etapa deberá realizarse utilizando nuevas imágenes o videos correctamente clasificados de acuerdo al caso de uso.

- **Refinamiento del modelo de inteligencia artificial actual** Si bien el módulo de inteligencia artificial entrega resultados aceptables, se podrían implementar mejoras para incrementar su efectividad. Entre las principales alternativas se destacan

    - **Ampliación del dataset**. Incorporar nuevas imágenes anotadas por múltiples expertos permitirá mejorar la cobertura de situaciones clínicas, reducir el sesgo de etiquetado y robustecer la capacidad de generalización del modelo.

    - **Exploración de arquitecturas más modernas**. Aunque las arquitecturas actuales están bien optimizadas, los avances recientes en el área de visión por computadora sugieren explorar alternativas como EfficientNet y

ConvNeXt, que ofrecen mejoras tanto en eficiencia computacional como en generalización.

- **Incorporación de la dimensión temporal.** Mediante arquitecturas que procesen secuencias completas (como LSTM, GRU o Transformers), se podría evitar la necesidad de transformar videos en imágenes. Sin embargo, estos modelos secuenciales requieren mayores tiempos de entrenamiento y podrían impactar en el rendimiento general del software.
- **Evaluación con métricas clínicas más precisas**. Incluir métricas como la sensibilidad por clase, la correlación con los LUS *Scores*, y el análisis estratificado del desempeño por perfil de paciente o condiciones de adquisición permitirá validar clínicamente los resultados de forma más rigurosa.

Estas líneas de trabajo permitirán no solo fortalecer la precisión del sistema, sino también ampliar su aplicabilidad clínica, convirtiéndolo en una herramienta de mayor impacto en la práctica médica y sentando las bases para su eventual transferencia tecnológica y adopción en entornos hospitalarios reales.

12. Memorias del proyecto

12.1 Diseño de arquitectura

Desde el inicio del proyecto se optó por una arquitectura que combinara tecnologías web con acceso a recursos del sistema operativo, como el sistema de archivos, la ejecución de scripts en Python y la utilización de un modelo de inteligencia artificial. Esta elección permitió alcanzar un equilibrio entre la flexibilidad en el desarrollo y la capacidad de procesamiento necesaria para las tareas complejas involucradas.

A diferencia de una solución completamente web, la arquitectura híbrida adoptada facilitó el trabajo directo con archivos locales y la ejecución de procesos intensivos de forma controlada, sin renunciar a una interfaz moderna y accesible. Entre los beneficios de esta decisión se destaca la posibilidad de realizar pruebas locales en diferentes plataformas sin requerir modificaciones significativas, además de simplificar una futura migración a entornos productivos o contenedores en caso de ser necesario.

El uso de esta arquitectura también habilita la posibilidad de expandir la solución hacia un entorno web o incorporar nuevos módulos, reutilizando componentes existentes y manteniendo la misma estructura de desarrollo.

Por otra parte, el entrenamiento y la ejecución de modelos de inteligencia artificial demandan una elevada capacidad de procesamiento. Implementar una arquitectura distribuida basada en la web habría implicado altos costos de infraestructura y mayores dificultades para realizar pruebas de forma eficiente. Al utilizar los recursos locales de los sistemas donde se ejecutó la aplicación, fue posible reducir tanto los tiempos de ejecución como los costos asociados a los procesos de entrenamiento y predicción, a pesar de las limitaciones de hardware disponibles.

12.2 Proceso de toma de decisiones técnicas

Durante la primera etapa de diseño se barajaron distintas posibilidades para resolver la problemática: realizar una aplicación web ó bien de escritorio. Bajo una mirada técnica, teniendo en cuenta los casos de uso, la practicidad, y la disponibilidad, se optó por la segunda alternativa. El equipo comprendió que esto implicaría una mayor demanda de recursos, pero a la vez era un trade-off necesario en pos de satisfacer los requerimientos preestablecidos.

Por otra parte, se evaluaron distintas técnicas de *machine learning* para aplicar el modelo de inteligencia artificial: aprendizaje federado, que consiste en implementar un sistema de análisis retroalimentado en un servidor ó bien aprendizaje localizado, tomando como referencia un modelo previamente entrenado para la tarea de clasificación.

12.3 Desarrollo del Sistema

El desarrollo del sistema presentó múltiples desafíos técnicos que exigieron una fuerte inversión en aprendizaje e investigación, particularmente en lo que respecta al tratamiento de videos. Aunque los integrantes del equipo contaban con experiencia previa en el desarrollo de aplicaciones web, nunca habían abordado un proyecto con una arquitectura híbrida que combinara elementos del stack web tradicional con procesamiento local de archivos multimedia.

Uno de los aspectos más complejos fue el procesamiento adecuado de los videos para que pudieran ser utilizados por el modelo de inteligencia artificial. Como se mencionó anteriormente, el programa encargado del análisis debía trabajar con imágenes estáticas.

Para lograr esta transformación, se decidió utilizar la librería FFmpeg, una herramienta sumamente potente pero con una curva de aprendizaje elevada. Fue necesario leer su documentación oficial, explorar los comandos, parámetros, y realizar numerosas pruebas para poder extraer los frames de los videos con la frecuencia, calidad y formato adecuados. También se tuvieron que adaptar los resultados de procesar los videos mediante la librería a las condiciones requeridas por la red neuronal, como la resolución de imagen, la cantidad de canales de color, y la estructura de datos a intercambiar.

A lo largo del proceso, surgieron problemas inesperados relacionados con codificaciones, pérdidas de calidad por compresión, y diferencias en la interpretación del formato. Estos obstáculos obligaron al equipo a profundizar en conceptos como codecs de video, tasas de muestreo, normalización de imágenes y preprocesamiento de datos. Gran parte de este trabajo no estaba contemplado en la estimación inicial del proyecto, por lo que se dedicó un tiempo mayor al previsto para esta etapa, lo que impactó en el cronograma general. Esta fue una de las razones por las que el tiempo de desarrollo pasó de 250 horas estimadas, a 300 horas realizadas.

Además, la configuración inicial del entorno también representó un desafío importante. Al tratarse de una arquitectura híbrida que integraba componentes web, scripts de procesamiento en Python y ejecución de procesos externos como FFmpeg, fue necesario definir cuidadosamente la estructura del proyecto, las dependencias, y la interoperabilidad entre módulos. Esta tarea, que podría parecer trivial en otros contextos, requirió una dedicación considerable al ser la primera experiencia del equipo. La definición de los scripts de inicio, la configuración del entorno virtual, la integración de procesos asincrónicos y la validación del flujo completo demandaron tiempo y pruebas iterativas.

Para asegurar la robustez del sistema, se implementaron pruebas de integración diseñadas para validar la correcta creación y el formato de los archivos intermedios. Estas pruebas verifican que las imágenes temporales se generen en la cantidad correcta según la tasa de fotogramas del video original. De igual forma, se valida que el archivo JSON con los resultados de la predicción de la red neuronal convolucional se genere correctamente, asegurando la integridad y el formato esperado de los datos. La validación rigurosa de estos datos intermedios es un punto crucial para el funcionamiento fiable del sistema, mientras que el resto de los requerimientos funcionales fueron validados por los responsables del proyecto a lo largo de su desarrollo.

En conjunto, el desarrollo del sistema no solo implicó escribir código, sino también adquirir nuevos conocimientos técnicos, redefinir estimaciones sobre la marcha y adaptar continuamente la planificación a medida que se profundizaba en cada subsistema.

12.4 Entrenamiento del Modelo

El entrenamiento del modelo de inteligencia artificial representó una etapa clave dentro del proyecto y resultó acertado haber modificado su lugar dentro de la planificación inicial. Si bien originalmente se había contemplado entrenar el modelo en fases más tempranas, postergar esta tarea permitió encarar el proceso con mayor conocimiento del dominio y un sistema funcional sobre el cual realizar pruebas integradas. Al momento de entrenar los pesos de la red neuronal, los integrantes del equipo ya estaban completamente interiorizados tanto en el flujo de datos como en las particularidades del procesamiento de video, lo que facilitó la validación práctica de los resultados y el ajuste fino de las entradas y salidas del modelo.

En un principio, se esperaba contar con un modelo pre-entrenado específicamente con imágenes de ecografías pulmonares, lo que ahorraría tiempos en investigación y desarrollo. Sin embargo, esto no fue así, y por ende surgió una nueva limitación que implicó reformular la estrategia: analizar el conjunto de datos (imágenes) que serían utilizadas, investigar distintos modelos generales de clasificación de imágenes y comenzar un proceso de prueba y error para evaluar su rendimiento. Se probaron distintas arquitecturas, criterios de congelamiento de capas, estrategias de fine-tuning y funciones de pérdida, hasta encontrar una configuración que permitiera resultados razonables. De aquí surge una de las diferencias respecto al tiempo planificado, se habían estimado 100 horas de trabajo para esta etapa, pero resultaron ser 140.

Otro de los desafíos que acompañó este desvío fue la cantidad de imágenes clasificadas disponibles para el entrenamiento. Si bien estaban correctamente etiquetadas, su volumen era muy limitado en comparación con lo que usualmente se necesita para entrenar redes neuronales profundas con alto nivel de precisión como las que requiere el proyecto. Esta restricción amplificó la complejidad del proceso, ya que no solo fue necesario ajustar los hiper parámetros del modelo para evitar sobreajuste, sino también implementar técnicas de validación cuidadosas y repetidas iteraciones para poder obtener resultados consistentes.

La falta de datos también obligó a extender los ciclos de entrenamiento y pruebas, superando lo estimado originalmente. En muchas ocasiones, los resultados del modelo no

eran representativos o suficientemente confiables, lo que llevó a descartar entrenamientos completos y replantear configuraciones desde cero. A pesar de estas dificultades, se logró alcanzar un rendimiento aceptable para los objetivos del proyecto, y se sentaron bases sólidas para futuras mejoras a partir de la incorporación de más datos y de nuevas técnicas que puedan surgir.

En retrospectiva, esta etapa resultó fundamental no solo desde el punto de vista técnico, sino también como instancia de aprendizaje. Permitió al equipo experimentar con el entrenamiento de modelos en condiciones no ideales, tomar decisiones respaldadas por evidencia y comprender con mayor profundidad tanto las limitaciones como las posibilidades del enfoque adoptado.

Asimismo, surgió una reflexión sobre la dificultad de estimar con precisión los tiempos necesarios para el desarrollo de un clasificador de imágenes, independientemente de la naturaleza del problema. Aunque es posible aproximar valores en función de experiencias previas, la realidad demuestra que influyen múltiples factores que impiden establecer cifras exactas, entre los que se destacan:

- Limitaciones en el conjunto de datos.
- Capacidad de cómputo. En general, el uso de unidades de procesamiento especializadas reduce significativamente el tiempo de entrenamiento; sin embargo, las de mayor potencia suelen estar disponibles únicamente en la nube bajo esquemas de IaaS (Infrastructure as a Service).
- Arquitectura de la red**.** Su complejidad (cantidad de capas, parámetros entrenables y tipo de operaciones) influye directamente en los recursos necesarios y en el tiempo de convergencia.

12.5 Documentación y desarrollo de Informe

En la planificación inicial se indicó que el desarrollo del documento escrito del proyecto se iba a realizar durante las distintas etapas del proyecto pero esta decisión no fue realizada de tal forma. La principal causa de esto fue que gran parte de las etapas previamente planificadas tomaron más tiempo de lo indicado y surgieron problemáticas que no estaban previstas, por lo que realizar el documento final en paralelo no iba a colaborar para mejorar esta situación.

Durante el desarrollo de todas las etapas lo que se realizó fue un tablero Kanban dentro de la plataforma *Trello*. El cual contaba con 4 columnas: *blocked*, *to do*, *doing*, *done*. En este

tablero se creaban todas las tareas definidas y subtareas más específicas y se asignaban entre los integrantes del equipo. De esta forma se podía visualizar el estado actual del proyecto y tener un seguimiento claro de las tareas que se habían realizado y las que aún quedaban por realizar.

Además, en otros documentos sin ningún formato en específico se registraban tareas y problemas que se presentaron durante el desarrollo con el fin de tener un registro de las situaciones que se presentaron para luego poder plasmarlas en el documento final. Todas las herramientas utilizadas sirvieron para realizar el documento de entrega y para retomar fácilmente las tareas luego de los recesos, sin tener una curva muy compleja para recordar el contexto del proyecto.

Finalmente, luego de terminar la etapa de desarrollo del proyecto se utilizaron las tareas documentadas, las bitácoras con los eventos relevantes del desarrollo y múltiples gráficos de resultados del entrenamiento para realizar la documentación del proceso del proyecto a entregar en conjunto con el *software* empaquetado. En conjunto, se utilizaron modelos extensos de lenguaje (LLM) para complementar la escritura y desarrollo del informe. Esta decisión permitió que el tiempo estimado para la documentación final del proyecto tome un tiempo menor al estimado.

12.6 Planificación versus realización

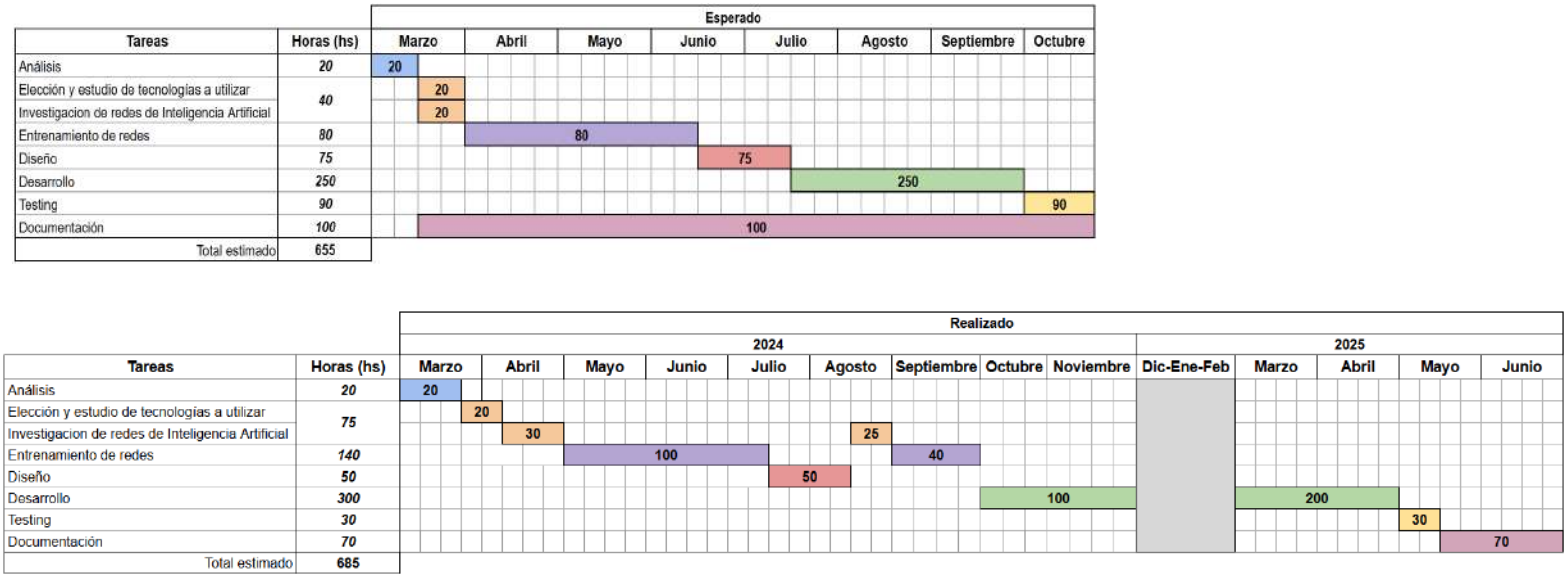

Figura 20. Comparativa entre los tiempos estimados y los realizados

Frente a lo mencionado anteriormente, y sumado a la aparición de nuevas responsabilidades personales y profesionales, el equipo experimentó períodos de pausa en el desarrollo activo del sistema, por ejemplo en el período de Diciembre de 2024 a Febrero de 2025. Este proceso comenzó en julio de 2024, fecha a partir de la que la disponibilidad diaria de ambos integrantes para cumplir con las tareas enmarcadas disminuyó significativamente. Otras causas como el estudio de librerías desconocidas por los miembros o las desviaciones generadas por el complejo proceso de ajuste de la red neuronal impactaron fuertemente en la desviación del tiempo planificado. La Tabla 4 muestra la comparación entre las estimaciones originales y los tiempos reales de ejecución.

Tabla 4. Comparación cuantitativa entre tiempo planificado y ejecutado

Tarea	Horas planificadas	Horas reales	Desviación (hs)	Desviación (%)
Análisis	20	20	0	0%
Investigación de IA	40	75	+35	+87.5%
Entrenamiento de redes	80	140	+60	+75%
Diseño	75	50	-25	-33.3%
Desarrollo	250	300	+50	+20%
Testing	90	30	-60	-66.7%
Documentacion	100	70	-30	-30%
Total	**655**	**685**	**+30**	**+4.6%**

En relación al análisis comparativo entre las horas ejecutadas y estimadas, se identifica que la desviación más significativa se presenta en la fase de entrenamiento de la red neuronal convolucional. Esta variación se atribuye fundamentalmente a dos factores: la curva de aprendizaje asociada a la implementación de técnicas de deep learning y la no disponibilidad de modelos pre-entrenados compatibles con el dataset específico del proyecto.

Es importante destacar que la postergación de la fecha de entrega no implicaba consecuencias negativas para usuarios finales, dado que el sistema aún no se encontraba en etapa de uso externo. Esta condición permitió priorizar temporalmente otros compromisos sin afectar la calidad del producto final.

A pesar de las interrupciones, el trabajo previo de análisis, documentación y definición de arquitectura funcionó como una guía sólida. Gracias a este enfoque, fue posible retomar el desarrollo sin una curva de entrada significativa, lo que evidencia la importancia de una etapa inicial bien estructurada. En definitiva, si bien se modificaron los plazos inicialmente previstos, se logró completar el proyecto de manera exitosa, con un aprendizaje enriquecedor tanto en lo técnico como en la gestión del tiempo y prioridades.

12.7 Análisis FODA a posteriori

Otro aspecto a destacar de la diferencia entre las expectativas iniciales y los resultados efectivamente alcanzados es el análisis FODA..

El presente análisis comparativo tiene como objetivo evaluar la evolución de los factores internos y externos que condicionaron el desarrollo del proyecto, contrastando la matriz FODA elaborada durante la fase de planificación inicial con una evaluación actualizada basada en la experiencia acumulada y los resultados obtenidos. Esta comparación permite no solo documentar el proceso de aprendizaje organizacional, sino también extraer lecciones para futuros desarrollos.

La metodología de comparación se estructura en torno a cuatro ejes fundamentales: la confirmación o refutación de las fortalezas identificadas inicialmente, la materialización efectiva de las oportunidades proyectadas, la evolución de las debilidades reconocidas y su eventual mitigación, y la manifestación real de las amenazas anticipadas junto con la emergencia de nuevos riesgos no previstos originalmente. A continuación se detallan los casos mencionados:

Fortalezas

La experiencia previa en algoritmos de aprendizaje automático (F1) se confirmó como un activo crítico, facilitando decisiones técnicas fundamentadas como la selección de MobileNet y la implementación de transfer learning.

Respecto al acceso a datos de entrenamiento (F3), inicialmente se esperaba una cantidad de elementos mayor a la obtenida. Si bien 1000 imágenes representan un nùmero bajo para lograr un entrenamiento de una red convolucional lo suficientemente potente para el uso mèdico, se debe destacar el trabajo realizado por los profesionales, ya que para lograr esa tarea debìan obtener las ecografìas, dividirlas en fotogramas y posteriormente evaluar cada imagen para determinar su clasificaciòn. Por otra parte, entre el material aportado se

encontraban más de 50 videos, con los que se puede aumentar el conjunto de datos de entrenamiento sustancialmente para futuros desarrollos.

Las capacidades de generación de conclusiones gráficas y numéricas (F4) no solo se confirmaron sino que se expandieron significativamente, desarrollando sistemas de visualización y métricas más sofisticados que los originalmente planificados. La capacidad de análisis detallado (F5) se implementó exitosamente con adaptaciones metodológicas que permitieron análisis integral con métricas de confianza.

Debilidades

Las debilidades identificadas mostraron una evolución heterogénea. La dependencia del dataset específico (D1) se confirmó como limitación crítica, mitigada parcialmente mediante técnicas de transfer learning, aunque la generalización del modelo permanece restringida al dominio de entrenamiento. La definición y ajuste del modelo (D2) también se hizo presente durante el transcurso del proyecto, pues el sobreentrenamiento ocurrió más a menudo de lo esperado.

Las limitaciones técnicas por restricciones temporales y de recursos (D3) se manifestaron en los entrenamientos locales de las distintas arquitecturas planteadas para el clasificador. La dependencia de recursos especializados (D4) se mantuvo constante. Contrariamente a lo previsto, no fue necesario recurrir a consultoría externa (D5), siendo suficientes las capacidades del equipo de trabajo.

Oportunidades

La demanda creciente de soluciones de IA médica (O1) se mantuvo acorde a lo proyectado, acelerada por el creciente uso de herramientas de inteligencia artificial que validó la relevancia del proyecto.

El aprovechamiento de avances tecnológicos (O3) se concretó parcialmente con la incorporación de arquitecturas eficientes como MobileNet V3, así como la evolución en los chips de procesamiento gráfico, que permitió obtener unidades de cómputo más robustas en la plataforma Google Colab. El potencial de expansión internacional (O4) fue identificado pero no explorado durante el período del proyecto debido a limitaciones de recursos.

Amenazas

La competencia creciente (A1) no se intensificó según lo previsto, debido a la especificidad del área de trabajo. Los cambios regulatorios (A2) no se materializaron significativamente, manteniéndose como consideración latente de diseño.

Las limitaciones presupuestarias (A3) impactaron parcialmente el alcance del proyecto, particularmente en recursos computacionales. Como se mencionó anteriormente, si se desea continuar con el entrenamiento de la red neuronal para obtener un modelo 100% óptimo para el ámbito, se debe hacer efectiva la incorporación de unidades gráficas altamente especializadas. Contrario a lo temido, la aceptación profesional médica (A4) resultó positiva mediante el enfoque de sistema de apoyo diagnóstico. La escasez de recursos tecnológicos para despliegue (A5) fue mitigada significativamente por los requerimientos computacionales reducidos de MobileNet, junto a la decisión de utilizar frameworks poco demandantes durante el desarrollo.

Síntesis

La comparación FODA revela que las fortalezas internas se materializaron en mayor medida que las proyecciones iniciales, mientras que las debilidades mostraron capacidad de mitigación superior a la esperada. Las oportunidades externas se confirmaron en gran medida, y las amenazas, aunque presentes, fueron menos limitantes de lo anticipado, particularmente debido a decisiones arquitecturales apropiadas y colaboraciones efectivas establecidas durante el desarrollo.

13. Conclusiones

En base a los objetivos planteados, todos fueron resueltos y desarrollados. El producto permite ser instalado localmente en cualquier PC sin depender de su sistema operativo. En su funcionamiento se habilita tanto la carga individual como múltiple de distintas ecografías, permitiendo un análisis completo del estado pulmonar al poder importar imágenes de distintos sectores de un mismo estudio.

Una vez cargados los videos, son procesados por un modelo de clasificación previamente desarrollado y entrenado. Además, se permite utilizar un clasificador de imágenes distinto al entregado, dando la posibilidad al usuario de hacer uso de nuevas herramientas para obtener mejores resultados. Finalmente, se muestra un panel general donde se visualizan los gráficos interactivos explicados durante el desarrollo del producto, y se da la posibilidad de exportarlos a un informe para su posterior estudio en profundidad. Cumple también con el objetivo de poder cargar videos de manera masiva y generar un archivo con los resultados del análisis por sobre todos los videos considerados. Todos los elementos visuales y parámetros a mostrar y/o modificar fueron validados continuamente durante el desarrollo para evitar retrocesos innecesarios durante el proyecto.

En cuanto a la gestión del proyecto, el desarrollo tomó un tiempo mayor a lo esperado. Esto se debió, en parte, por requerir de mayor experiencia en algunos aspectos para poder finalizar el desarrollo y, por otro lado, por pausas e inactividad que se dieron a causa de contratiempos y nuevas responsabilidades de los integrantes. Estos cambios de prioridad se dieron luego de analizar el impacto del proyecto en caso de postergar la fecha de entrega. Al no tener una dependencia económica ni urgencia sobre la entrega del producto por parte del demandante, la decisión de postergar la entrega no supuso una situación crítica o que pudiera poner en peligro la finalización del proyecto.

En cuanto a la estimación de tiempos, el equipo subestimó el tiempo que puede tomar el entrenamiento de este tipo de redes y la demanda computacional que esta exige. En esta versión preliminar se presentó un modelo que puede servir como predecesor para un próximo trabajo que se especifique únicamente en la obtención de mayor cantidad de datos, en la prueba y configuración de otros hiper parámetros para lograr resultados más precisos. Igual fue de gran aprendizaje este proceso para aplicar los conocimientos adquiridos en cuanto a este tipo de redes a un dataset de aplicación real.

En conclusión, se logró cumplir con los objetivos establecidos, a pesar de que en distintas etapas del proyecto surgieron situaciones en las que hubo que tomar variaciones respecto al plan de trabajo para poder mantener el ritmo de desarrollo y la voluntad del equipo. También hubo que indagar sobre herramientas nuevas y patrones que los integrantes no conocían previamente. Todas estas decisiones y etapas fueron un gran aprendizaje tanto para este proyecto como para los futuros. Además de formar profesionalmente a los integrantes en cuanto al aspecto técnico, destacando el impacto social y científico que puede aportar en un área que, si bien no es el campo principal del equipo, permite generar soluciones concretas, avances y herramientas, aspectos fundamentales que debe aportar un ingeniero.

14. Bibliografía

Bouhemad, B., Zhang, M., Lu, Q., & Rouby, J. J. (2007). Clinical review: bedside lung ultrasound in critical care practice. *Critical Care*, 11(1), 205.

Howard, A., Sandler, M., Chu, G., et al. (2019). Searching for mobilenetv3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314-1324.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1), 60.

Soldati, G., Demi, M., Smargiassi, A., et al. (2019). The role of ultrasound lung artifacts in the diagnosis of respiratory diseases. *Expert Review of Respiratory Medicine*, 13(2), 163-172.

Tusman, G., Acosta, C. M., Böhm, S. H., et al. (2012). Lung recruitment and positive end-expiratory pressure have different effects on CO2 elimination in healthy and sick lungs. *Anesthesia & Analgesia*, 115(2), 318-324.

Xirouchaki, N., Magkanas, E., Vaporidi, K., et al. (2011). Lung ultrasound in critically ill patients: comparison with bedside chest radiography. *Intensive Care Medicine*, 37(9), 1488-1493.

Tusman, G., Acosta, C. M., & Costantini, M. (2016). Ultrasonography for the assessment of lung recruitment maneuvers. *Critical Ultrasound Journal, 8*(1), 8. https://doi.org/10.1186/s13089-016-0045-9

Rouby, J. J., Arbelot, C., Gao, Y., Zhang, M., Lv, J., An, Y., Chunyao, W., Bin, D., Valente Barbas, C. S., Dexheimer Neto, F. L., Prior Caltabeloti, F., Lima, E., Cebey, A., Perbet, S., Constantin, J. M., & APECHO Study Group. (2018). Training for lung ultrasound score measurement in critically ill patients. *American Journal of Respiratory and Critical Care Medicine, 198*(3), 398–401. https://doi.org/10.1164/rccm.201802-0227LE

Mongodi, S., Chiumello, D., & Mojoli, F. (2024). Lung ultrasound score for the assessment of lung aeration in ARDS patients: Comparison of two approaches. *Ultrasound International Open, 10*, a24218709. https://doi.org/10.1055/a-2421-8709

Raina, R., Madhavan, A., & Ng, A. Y. (2009). Large-scale deep unsupervised learning using graphics processors. Proceedings of the 26th Annual International Conference on Machine Learning, 873-880.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Google Research. (2024). Colaboratory FAQ - Hardware specifications. Google LLC. https://research.google.com/colaboratory/faq.html

NVIDIA Corporation. (2024). GPU Computing Performance Benchmarks for AI and Machine Learning. NVIDIA Developer Documentation. https://developer.nvidia.com/deep-learning-performance-training-inference

Electron. (s.f.). *Electron documentation*. https://www.electronjs.org/docs/latest

FFmpeg. (s.f.). *FFmpeg documentation*. https://ffmpeg.org/documentation.html

Vite. (s.f.). *Getting started with Vite*. https://vite.dev/guide/

TensorFlow. (s.f.). *Guía de TensorFlow*. https://www.tensorflow.org/guide?hl=es-419

Python. (s.f.). *The Python Standard Library documentation*. https://docs.python.org/3/

NumPy. (s.f.). *NumPy documentation*. https://numpy.org/doc/

Conventional Commits. (s.f.). *Conventional Commits v1.0.0*. https://www.conventionalcommits.org/en/v1.0.0/

15. Glosario

Término	Definición
Atelectasia	Colapso parcial o total de una región del pulmón debido a la falta de aire en los alvéolos, lo que reduce el intercambio gaseoso y puede comprometer la oxigenación sanguínea.
Backbone convolucional	Arquitectura de red neuronal convolucional pre entrenada que se utiliza como base o esqueleto principal para extraer características de imágenes, sobre la cual se pueden agregar capas adicionales específicas para la tarea objetivo.
Backend	Parte del sistema encargada de procesar la lógica de negocio, acceder a los datos y gestionar el funcionamiento interno de la aplicación.
Batch	Modalidad de procesamiento en la cual se manejan múltiples datos o archivos simultáneamente, sin necesidad de intervención del usuario para cada uno.
Cluster	Conjunto de datos similares agrupados según patrones comunes. Utilizado en tareas de análisis o visualización.
Data augmentation	Técnica de procesamiento de imágenes que consiste en manipular su estado
Dropout	Técnica de regularización en redes neuronales que consiste en desactivar aleatoriamente un porcentaje de neuronas durante el entrenamiento para prevenir el sobreajuste y mejorar la capacidad de generalización del modelo.
Electron	Framework que permite desarrollar aplicaciones de escritorio utilizando tecnologías web (HTML, CSS, JavaScript). Combina Node.js y Chromium.
Features	Atributos extraídos de los datos que son usados por el modelo de inteligencia artificial para aprender y clasificar.

Término	Definición
FFMPEG	Librería para procesar archivos de audio y video. Se utilizó para recortar videos y extraer frames.
Fine-tuning	Proceso de reentrenamiento de una red neuronal pre entrenada utilizando datos específicos del dominio objetivo, ajustando los pesos para optimizar el rendimiento en la nueva tarea.
FLOPs	Medida que cuantifica el número de operaciones de punto flotante requeridas por un algoritmo, utilizada como métrica de complejidad computacional.
Frame	Cada una de las imágenes individuales que componen un video. En este proyecto, los análisis de ecografía se realizan frame a frame.
Framework	Estructura de software que proporciona una base común de código y herramientas para desarrollar aplicaciones de manera más eficiente, ofreciendo funcionalidades reutilizables y definiendo la arquitectura general del sistema.
Frontend	Parte visual de la aplicación con la que interactúa el usuario directamente, usualmente desarrollada en tecnologías como React o Angular
ipcMain	Módulo de Electron que permite la comunicación entre la interfaz gráfica y el backend de la aplicación.
JSON	Formato de texto ligero usado para el intercambio de datos entre aplicaciones. Se emplea en este proyecto para transferir resultados entre Node.js y Python.
Kanban	Método de gestión visual de trabajo que organiza tareas en columnas según su estado

Término	Definición
Matriz de confusión	Tabla utilizada para evaluar el rendimiento de un modelo de clasificación. Resume las predicciones realizadas por el modelo frente a los valores reales conocidos. Cada fila representa las instancias reales de una clase y cada columna representa las predicciones hechas por el modelo (o viceversa).
MobX	Biblioteca para gestión de estados en aplicaciones React. Implementa el patrón observador para actualizar automáticamente la interfaz gráfica ante cambios de datos.
Mocks	Simulaciones de funciones o datos utilizados durante el desarrollo cuando una funcionalidad aún no está implementada.
Node.js	Entorno de ejecución para JavaScript del lado del servidor. Permite construir aplicaciones backend modernas y eficientes.
Overfitting	Situación donde un modelo aprende demasiado bien los datos de entrenamiento y falla al generalizar sobre nuevos datos.
Purity	Métrica utilizada para evaluar la calidad de una agrupación (clustering). Mide qué tan homogéneos son los grupos generados por un algoritmo de clustering en relación con las clases reales de los datos.
Python	Lenguaje de programación ampliamente utilizado en inteligencia artificial y ciencia de datos. En este proyecto se usa para implementar y ejecutar el modelo de clasificación de imágenes.
React	Biblioteca de JavaScript para construir interfaces de usuario. Permite crear componentes reutilizables y administrar el estado de forma eficiente.
Score	Métrica numérica que resume el resultado del análisis por parte del modelo de inteligencia artificial. Puede calcularse mediante diferentes métodos.
Softmax	Función matemática que convierte un vector de valores en una

Término	Definición
	distribución de probabilidades, asignando valores entre 0 y 1 que suman 1
Stack	Conjunto de tecnologías, herramientas y frameworks utilizados para desarrollar aplicaciones web
Store	Componente dedicado a almacenar información compartida entre componentes que se mantienen y acceden durante tiempo de ejecución.
Transfer learning	Técnica de entrenamiento donde se aprovecha el conocimiento de modelos ya entrenados para acelerar el desarrollo de uno nuevo, con mejores resultados y menor cantidad de datos.
Transformers	Modelo de arquitectura neuronal que procesa datos secuenciales considerando todas las posiciones de entrada simultáneamente, lo que permite capturar relaciones contextuales a largo plazo de forma eficiente.
Trello	Aplicación de gestión de proyectos basada en tableros visuales, que permite organizar tareas mediante listas y tarjetas, facilitando la colaboración en equipo y el seguimiento de flujos de trabajo.
VRAM	Tipo de memoria de alta velocidad integrada en las tarjetas gráficas, diseñada específicamente para almacenar datos gráficos como texturas, buffers de imagen y, en el contexto de machine learning, activaciones de redes neuronales, gradientes y parámetros del modelo durante el entrenamiento y la inferencia

16. Anexos

Anexo I: Documentación de Endpoints del Backend

A continuación, se documentan los principales endpoints implementados en el backend del sistema, utilizados para la comunicación entre el frontend y la capa lógica a través del módulo *ipcMain* de Electron.

Cada endpoint gestiona funcionalidades específicas necesarias para el procesamiento de videos, la ejecución del modelo de inteligencia artificial, el cálculo de métricas y la generación de informes.

"ffmpeg-crop"

Funcionalidad:
Realiza el recorte de un video ecográfico y genera los frames correspondientes.

Parámetros de entrada:

- filePath: Ruta del archivo de video a procesar.
- args: Filtro de recorte aplicado.

Salida:

- Objeto con isSuccess, data (buffer del video recortado) y msg (ruta del archivo generado).

"run-model"

Funcionalidad:
Ejecuta el script Python para el análisis de frames mediante el modelo de inteligencia artificial.

Parámetros de entrada:

- fileNames: Lista de nombres de videos a analizar.

Salida:

- true si la ejecución finaliza correctamente.
- false en caso de error.

"calc-score"

Funcionalidad:
Calcula el score pulmonar basado en la matriz de resultados de frames, según el método seleccionado.

Parámetros de entrada:

- method: Método de cálculo (1: Promedio, 2: Probabilístico).
- data: Matriz de resultados por frame.

Salida:

- Objeto con scoreFrames (scores por frame) y scoreFinal (score global).

"calc-score-global"

Funcionalidad:
Calcula scores normalizados por clase a partir de los resultados globales de un video.

Parámetros de entrada:

- data: Matriz de resultados de frames.

Salida:

- Array de scores normalizados para cada clase (Normal, B1, B2, Consolidación).

"save-csv"

Funcionalidad:
Guarda un archivo CSV generado a partir del contenido proporcionado.

Parámetros de entrada:

- csvContent: Contenido del archivo CSV en formato texto.

Salida:

- Objeto indicando success y la filePath del archivo guardado.

"select-file-path"

Funcionalidad:
Abre un cuadro de diálogo para seleccionar la ubicación de guardado de un informe PDF.

Parámetros de entrada:

- title: Título de la ventana de diálogo.

Salida:

- Objeto con la ruta seleccionada o indicación de cancelación.

"save-pdf-file"

Funcionalidad:
Guarda un archivo PDF generado a partir de un buffer.

Parámetros de entrada:

- filePath: Ruta donde se guardará el archivo PDF.
- buffer: Contenido binario del archivo.

Salida:

- Objeto indicando el éxito de la operación.

"upload-new-model"

Funcionalidad:
Permite seleccionar e incorporar un nuevo modelo de inteligencia artificial a la aplicación.

Parámetros de entrada:

- No requiere parámetros.

Salida:

- Objeto de registro del modelo cargado.

"get-model-registry"

Funcionalidad:
Recupera la lista de modelos de IA actualmente registrados en el sistema.

Parámetros de entrada:

- No requiere parámetros.

Salida:

- Lista de modelos disponibles.

"set-default-model"

Funcionalidad:
Establece un modelo de IA seleccionado como el modelo predeterminado.

Parámetros de entrada:

- modelId: Identificador del modelo a definir como predeterminado.

Salida:

- Confirmación del cambio de modelo por defecto.

"finish-batch-analysis"

Funcionalidad:
Finaliza el análisis por lotes de múltiples videos y genera un informe consolidado en formato CSV.

Parámetros de entrada:

- No requiere parámetros explícitos (usa resultados previamente almacenados).

Salida:

- Objeto indicando success y la ruta del archivo CSV generado.

Anexo II: Formato del archivo de intercambio JSON

El intercambio de datos entre el backend de Node.js y el script de Python se realiza mediante un archivo en formato JSON.

El archivo contiene un objeto donde la clave corresponde al nombre del video analizado y el valor asociado es una matriz que representa los resultados de cada frame procesado.

El formato definido es el siguiente:

```
{
  "nombre_video": [
    [resultado_frame_1],
    [resultado_frame_2],
    [resultado_frame_3],
    ...
  ]
}
```

El campo `"nombre_video"` identifica el video procesado, mientras que cada entrada en el array contiene la salida del análisis para un frame individual. Esta estructura permite recorrer los resultados de forma ordenada para su posterior procesamiento en el sistema.

Anexo III: Resultados entre Mediana y Softmax

Tabla 5: Probabilidades de clases obtenidas en los primeros 20 frames de una ecografía analizada.

Frame	Normal (0)	B1 (1)	B2 (2)	Cons (3)
1	0.87	0.09	0.04	0.00
2	0.84	0.12	0.04	0.00
3	0.86	0.09	0.05	0.00
4	0.88	0.08	0.04	0.00
5	0.89	0.07	0.04	0.00
6	0.85	0.10	0.05	0.00
7	0.90	0.07	0.03	0.00
8	0.91	0.06	0.03	0.00
9	0.87	0.09	0.04	0.00
10	0.88	0.08	0.04	0.00
11	0.86	0.10	0.04	0.00
12	0.84	0.12	0.04	0.00
13	0.89	0.06	0.05	0.00
14	0.85	0.09	0.06	0.00
15	0.88	0.07	0.05	0.00
16	0.90	0.06	0.04	0.00
17	0.87	0.08	0.05	0.00
18	0.89	0.07	0.04	0.00
19	0.86	0.09	0.05	0.00
20	0.88	0.08	0.04	0.00

Tabla 5: Cálculos de mediana, mediana normalizada y softmax en base a la Tabla 3.

Clase	Mediana	Mediana Normalizada	Softmax
Normal	0.865	0.870	0.472
B1	0.087	0.087	0.216
B2	0.042	0.042	0.206
Cons	0.000	0.000	0.106

Anexo IV: Visualizaciones con Análisis de Componentes Principales y matrices de confusión considerando features extraídas por distintos modelos pre entrenados.

Como primera etapa de la investigación, se buscó evaluar la pureza del conjunto de datos a utilizar para el entrenamiento del modelo de inteligencia artificial. Este análisis tenía como objetivo diagnosticar la calidad y coherencia interna de las imágenes, así como determinar cuán diferenciables eran entre sí desde el punto de vista de un modelo no entrenado.

Para ello, se seleccionaron arquitecturas de redes neuronales convolucionales reconocidas, sobre las que se aplicaron los pesos preentrenados en ImageNet, un conjunto de datos estándar utilizado para tareas de clasificación de imágenes.

Una vez cargados estos modelos, se procesaron las imágenes del conjunto propio, y se extrajeron las features (representaciones intermedias) de cada una de ellas. Con estas representaciones vectoriales se llevó a cabo un análisis de clustering, con el fin de observar cómo se agrupan las imágenes en función de sus características internas. Esta metodología permitió obtener una aproximación objetiva al grado de diferenciabilidad entre clases (estados pulmonares), utilizando un estándar robusto y validado a nivel mundial.

Los gráficos cuentan con dos formas de visualización: en colores, los agrupamientos realizados mediante el algoritmo *kMeans*, agrupando aquellas imágenes cuyas representaciones sean más semejantes. Por otro lado, las formas representan la clase original a la que pertenecía esa imagen.

A continuación, se detallan los gráficos mencionados anteriormente, y los resultados de su posterior análisis:

Modelo pre-entrenado: MobileNet

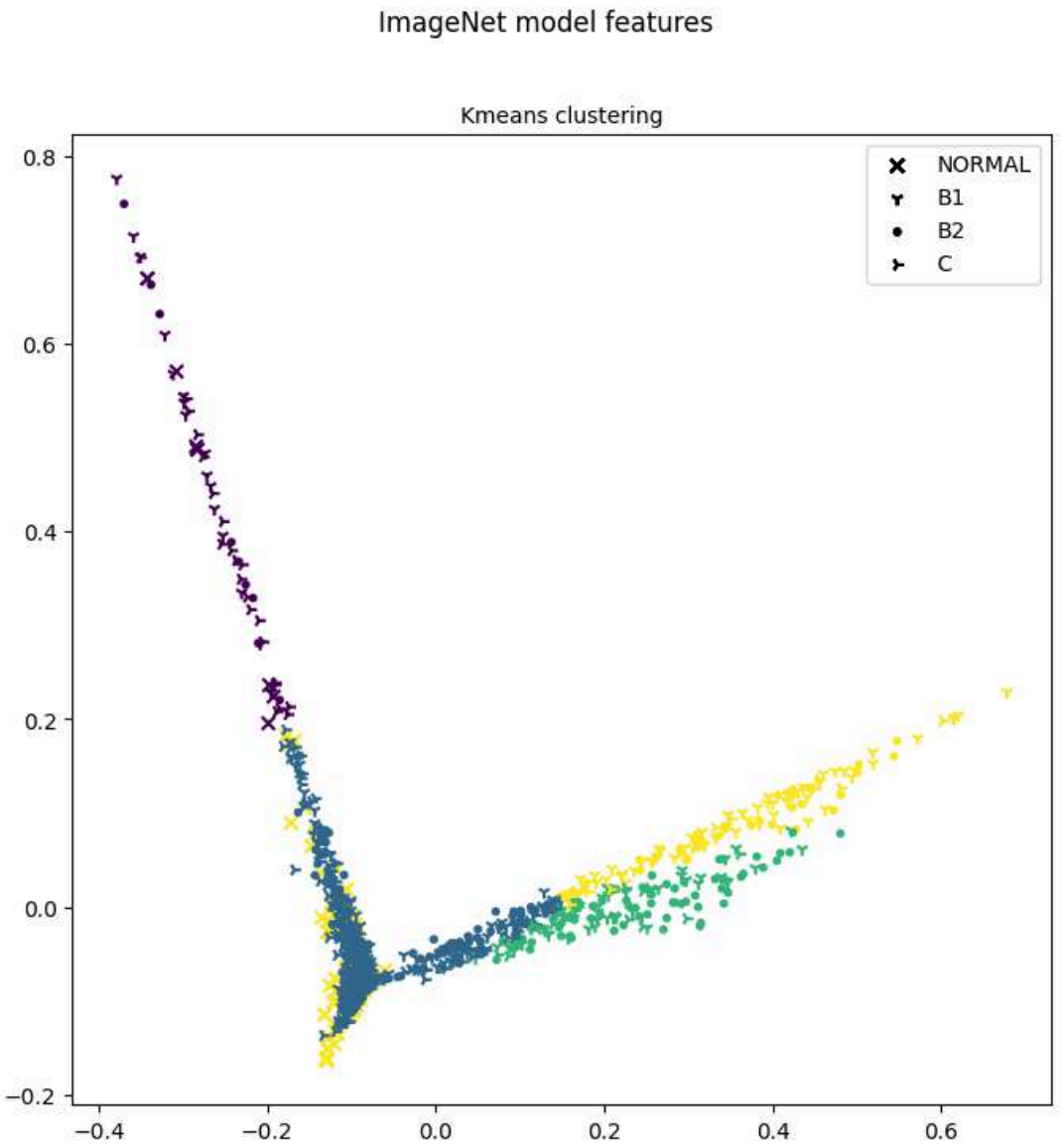

Figura 21. Clustering para MobileNet

Tabla 6. Matriz de confusión. Clases perteneciente vs Clase agrupada por clustering.(MobileNet)

Estado pulmonar	Normal	B1	B2	Consolidación
Normal	9	16	9	22
B1	109	106	113	170
B2	0	49	69	20
Consolidación	0	63	31	15

```
Pureza obtenida: 52%
```

Modelo pre-entrenado: MobileNet V2

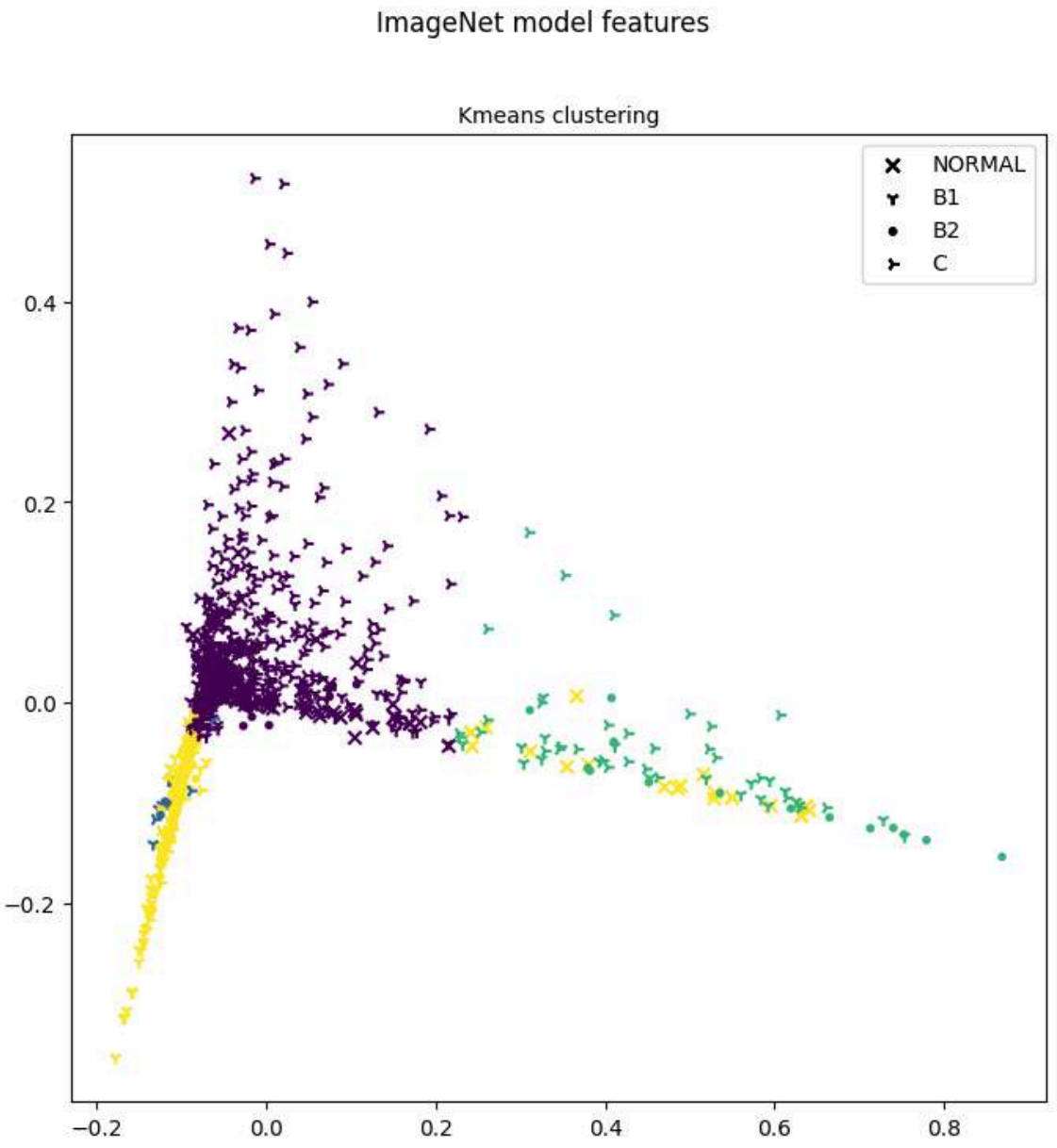

Figura 22. Clustering para MobileNet V2

Tabla 7. Matriz de confusión. Clases perteneciente vs Clase agrupada por clustering. (MobileNet V2)

Estado pulmonar	Normal	B1	B2	Consolidación
Normal	96	107	114	355
B1	0	7	3	3
B2	22	20	14	20
Consolidación	0	100	91	39

```
Pureza obtenida: 49%
```

Modelo pre-entrenado: EfficientNet

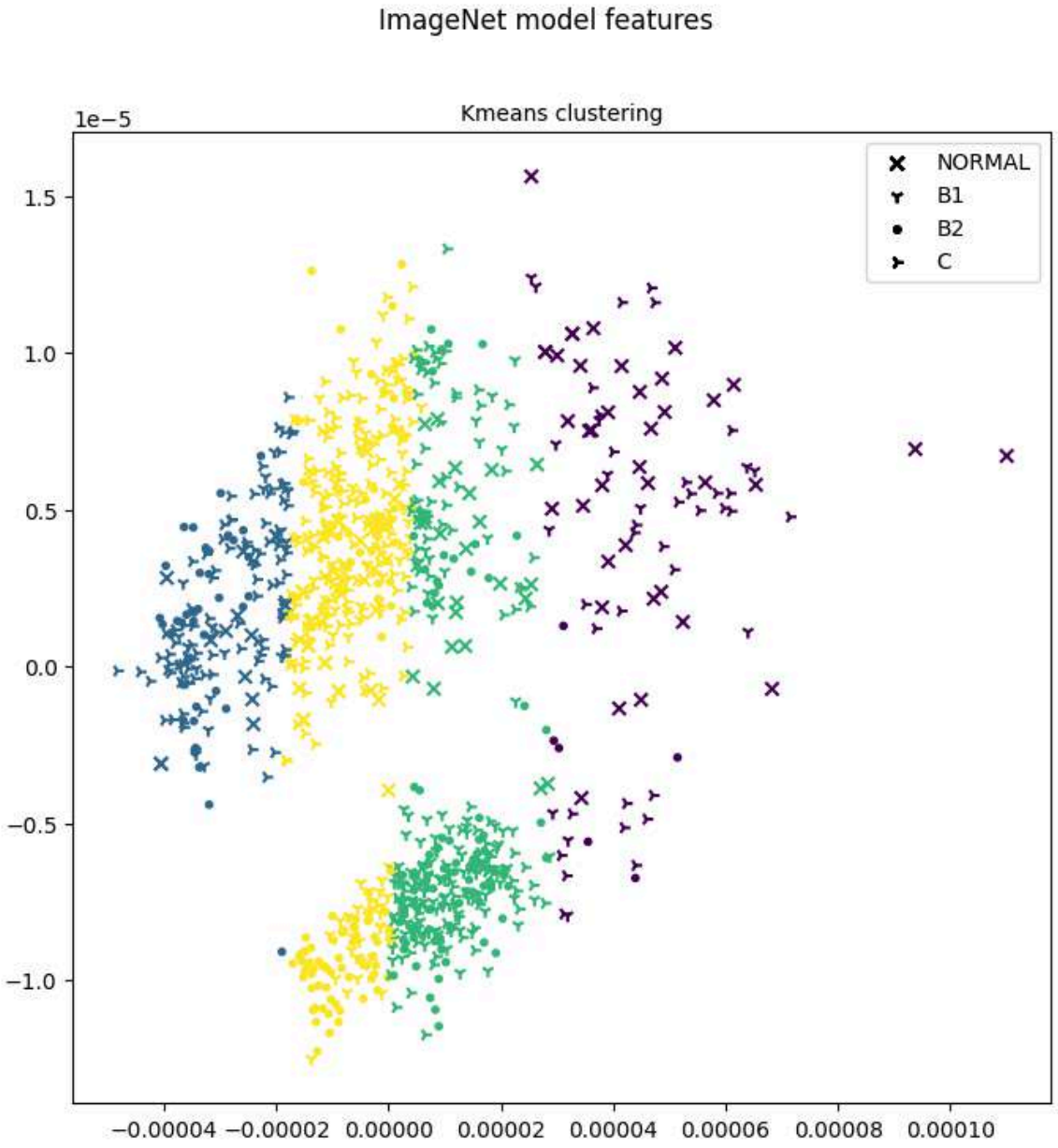

Figura 23. Clustering para EfficientNet

Tabla 8. Matriz de confusión. Clases perteneciente vs Clase agrupada por clustering. (EfficientNet)

Estado pulmonar	Normal	B1	B2	Consolidación
Normal	39	13	6	32
B1	15	21	39	99
B2	26	120	86	107
Consolidación	38	80	91	189

Pureza obtenida: 45%

Modelo pre-entrenado: ConvNeXT

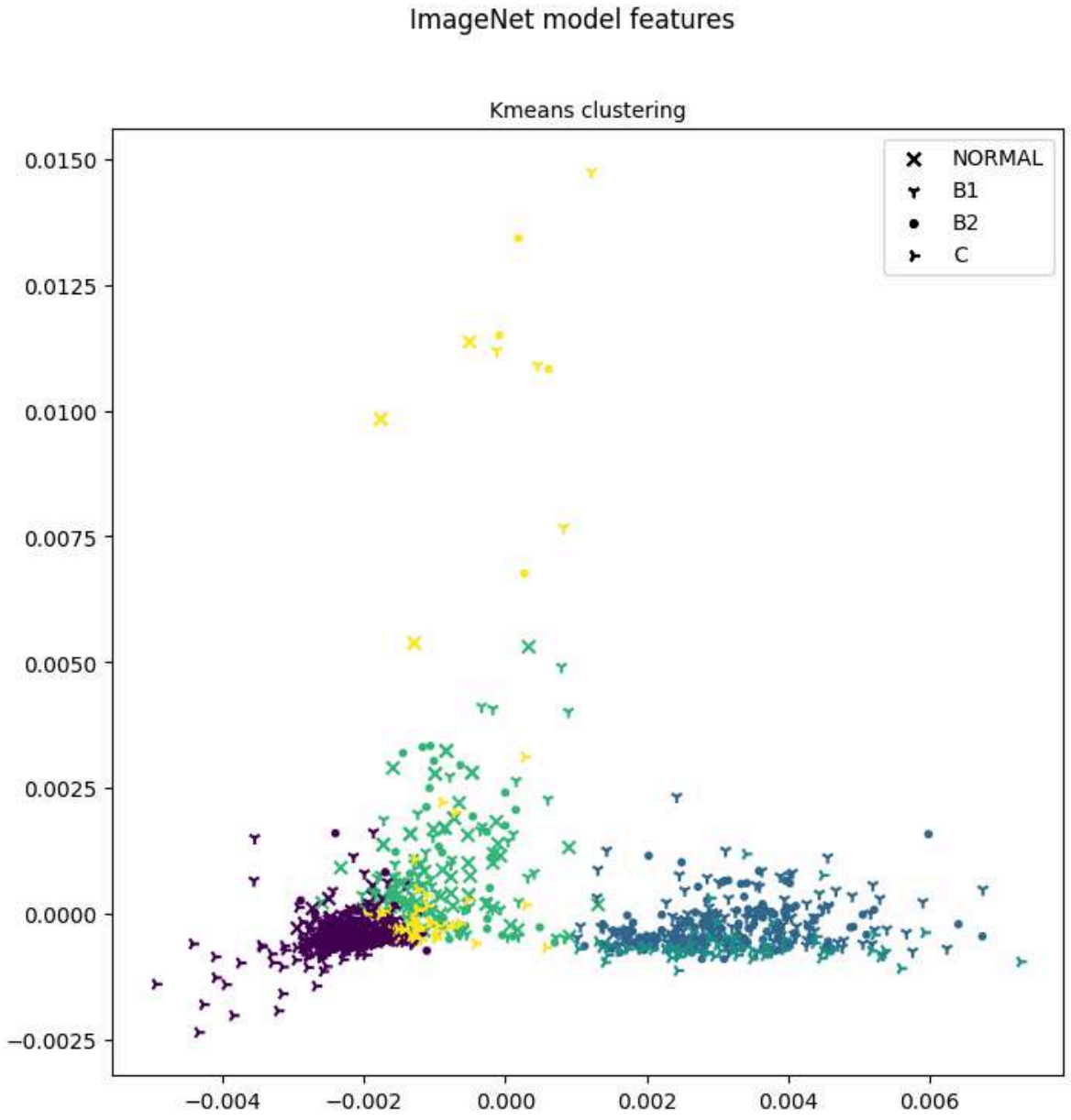

Figura 24. Clustering para ConvNeXT

Tabla 9. Matriz de confusión. Clases perteneciente vs Clase agrupada por clustering. (ConvNeXT)

Estado pulmonar	Normal	B1	B2	Consolidación
Normal	34	27	32	310
B1	1	139	139	69
B2	78	64	47	48
Consolidación	5	4	4	0

Pureza obtenida: 53%

Modelo pre-entrenado: NasNet

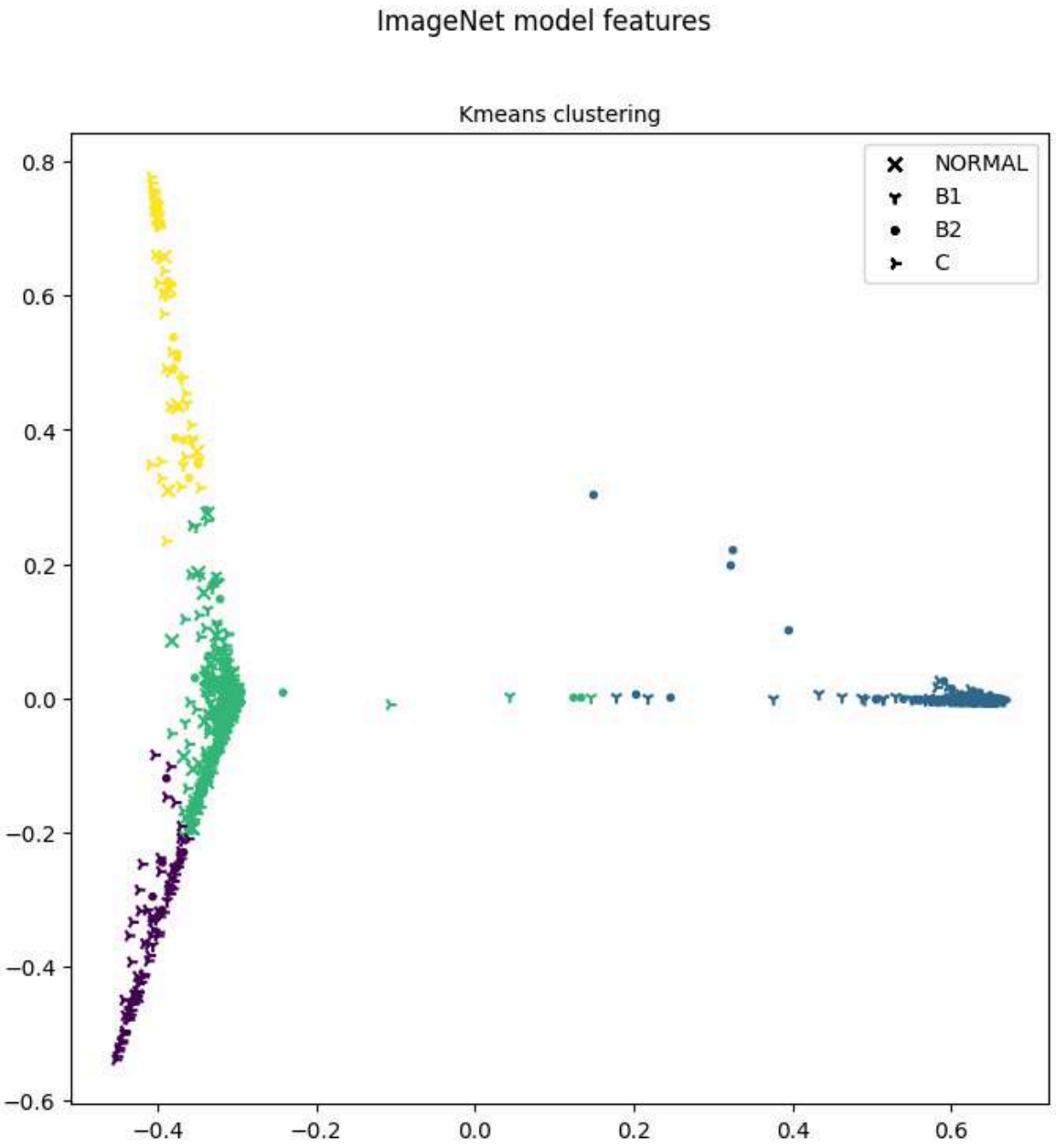

Figura 25. Clustering para NasNet

Tabla 10. Matriz de confusión. Clases perteneciente vs Clase agrupada por clustering. (NasNet)

Estado pulmonar	Normal	B1	B2	Consolidación
Normal	1	3	9	75
B1	0	141	113	170
B2	111	78	62	238
Consolidación	6	12	10	44

Pureza obtenida: 49%

Modelo pre-entrenado: InceptionResNet

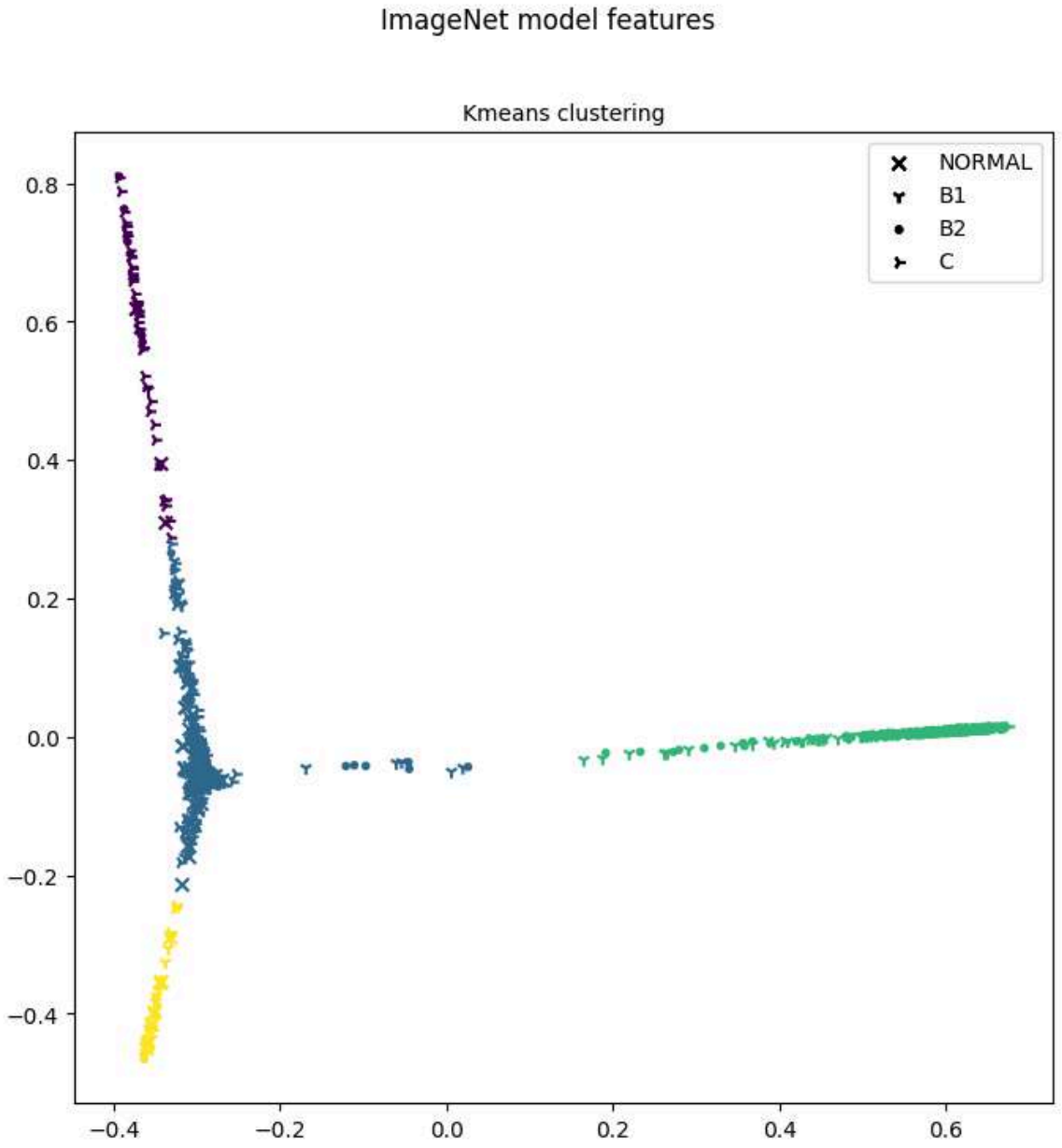

Figura 26. Clustering para InceptionResNet

Tabla 11. Matriz de confusión. Clases perteneciente vs Clase agrupada por clustering. (InceptionResNet)

Estado pulmonar	Normal	B1	B2	Consolidación
Normal	4	2	11	38
B1	104	81	66	313
B2	0	138	138	70
Consolidación	10	13	7	6

Pureza obtenida: 50%

A partir de los resultados, se puede observar que en general la pureza ronda el 50%. Este valor indica una limitada capacidad de agrupamiento basado en las características extraídas, lo que sugiere que las clases presentes en el conjunto de datos no se encuentran claramente diferenciadas. Esto puede deberse a la similitud visual entre clases, a una posible contaminación del dataset, o a que los modelos utilizados no son los más apropiados para el dominio específico del problema.

Anexo V: Modelo de Inteligencia Artificial

Para poder encontrar un modelo de Inteligencia Artificial acorde a nuestro caso de estudio, se decidió realizar un proceso de transfer-learning, agregando luego de la extracción de features un clasificador apropiado al problema a resolver, con diferentes arquitecturas (combinaciones de capas y neuronas por capa). La arquitectura general, en código, siguió la siguiente forma:

```
from keras.applications import MobileNet
from keras.layers import Dense, Flatten, Dropout, MaxPooling2D,
GlobalAveragePooling2D
from keras.models import Model
IMG_SHAPE = (224, 224, 3)
base_model = MobileNet(weights='imagenet', include_top=False,
input_shape=IMG_SHAPE)

# Añadimos capas de clasificación
last_layer = base_model.output
x = MaxPooling2D()(last_layer)
x = Flatten()(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
x = Dense(512, activation='relu')(x)
predictions = Dense(4, activation='sigmoid')(x)


model = Model(base_model.input, outputs=predictions)
```

Anexo VI: Comparación entre modelos de Inteligencia Artificial

Durante la etapa de entrenamiento, se probaron distintas combinaciones, basándonos en experiencias previas del Laboratorio de Bioingeniería para este caso de estudio. En primer lugar, se tomó como referencia el uso de MobileNet como modelo principal, al que luego se le agregaron las capas de clasificación acorde al problema planteado. En este punto se probaron las siguientes alternativas:

- Entrenamiento de la red neuronal completa. (15 millones de parámetros). Esta idea fue descartada por la complejidad del proceso, y porque el *dataset* de entrenamiento no era lo suficientemente amplio para poder asegurar que los parámetros se ajusten correctamente.
- Cambios en la capa de clasificación. Se probaron distintas configuraciones, aunque con estructuras similares, para poder tener varios puntos de comparación.
- Ajuste de hiper parámetros, con distintos optimizadores y valores de configuración.
- División del conjunto de datos. En base a experiencias previas y a las limitaciones dadas, se decidió que dividir el conjunto de datos en dos partes sería la mejor decisión para evitar sobreentrenamientos precipitados. Esto sucedió durante los primeros intentos, en los que el avance de la red se estancaba, y no mejoraba las métricas de precisión.

En la tabla 12 se pueden observar las variantes más exitosas probadas durante este proceso.

En términos de desempeño, la arquitectura MobileNet resultó ser la más estable y versátil, permitiendo introducir múltiples variaciones en las capas de clasificación, el tipo de pooling, la regularización mediante *dropout* y el optimizador utilizado.

Tabla 12. Arquitecturas de red utilizadas durante el entrenamiento.

Arquitectura principal	Capas de clasificación	Optimizador	Épocas de entrenamiento	Accuracy entrenamiento	Accuracy validación	Loss validación	Accuracy Test	Observaciones
MobileNet	MaxPool Flatten Dropout(0.5) Dense(1024, relu) Dense(4, softmax)	RMSProp (lr=0.001)	125	100%	88.9%	0.27	65%	Se establecieron como entrenables las últimas 10 capas de la red, y las de clasificación.
MobileNet	Idem anterior.	RMSprop (lr=0.001)	150	100%	90.5%	0.35	70%	Se establecieron como entrenables las últimas 4 capas de la red.
MobileNet	GAP Flatten Dropout(0.5) Dense(1024, relu) Dense(1024, relu) Dense(512, relu) Dense(4, sigmoid)	Adam(lr=0,0001, beta_1=0.9, beta_2=0.999)	50	100%	83.5%	0.35	28%	Clasificación por función sigmoidea.
MobileNet	GAP Dropout(0.7) Dense(1024, relu) Dense(1024, relu) Dense(4, softmax)	Adam (lr=0.001)	150	100%	87%	0.53	30%	Alternancia entre optimizador RMSProp y Adam. Clasificación por SOFTMAX.
MobileNet	GAP Flatten Dropout(0.7) Dense(1024, relu) x2 Dense(512, relu) Dense(4, softmax)	RMSprop (lr=0.001)	100	100%	83.5%	0.54	50%	Últimas 7 capas de la red descongeladas.