PROYECTO FIN DE GRADO

TÍTULO: Deep learning para la clasificación de sarcomas de tejidos blandos en imágenes histopatológicas

AUTOR/A: María Sánchez Martínez

TITULACIÓN: Ingeniería Telemática

TUTOR/A: Juana Mª Gutiérrez Arriola

DEPARTAMENTO: Departamento de Ingeniería Telemática y Electrónica

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Nicolás Sáenz Lechón

TUTOR/A: Juana Mª Gutiérrez Arriola

SECRETARIO/A: Eduardo Juárez Martínez

Fecha de lectura: 22 de julio de 2025

Calificación:

El Secretario/La Secretaria,

Resumen

Los sarcomas de partes blandas son tumores malignos poco frecuentes que pueden originarse en tejidos como músculo, grasa o vasos sanguíneos, y se caracterizan por una elevada heterogeneidad histológica y un comportamiento agresivo. Debido a su baja prevalencia y complejidad histológica, su diagnóstico es especialmente complejo y requiere la colaboración de expertos en centros de referencia.

Este trabajo propone una implementación basada en aprendizaje profundo para la clasificación de los tres subtipos más frecuentes (leiomiosarcoma, liposarcoma desdiferenciado y sarcoma pleomórfico indiferenciado) a partir de imágenes histológicas digitalizadas, con el objetivo de servir como herramienta de apoyo al diagnóstico o para la investigación en sarcomas.

La implementación parte de un proyecto previo obsoleto, realizando distintos experimentos hasta llegar a una solución final. Las mejoras más importantes incluyen la migración a las versiones actuales de `FastAI` y `PyTorch` y la adopción de validación cruzada estratificada por paciente, probando diversos ensambles de modelos hasta llegar al modelo final. Para el entrenamiento se generó un conjunto de datos propio a partir del conjunto TCGA-SARC, mediante un proceso de selección y preprocesamiento de imágenes en formato `.svs` para extraer imágenes (*tiles*) de 1024×1024 píxeles.

A lo largo de los distintos experimentos realizados, se identificó la importancia de contar con un conjunto de datos representativo, de emplear validación cruzada en contextos con pocos datos y de colaborar con un patólogo con experiencia en sarcomas para la selección de muestras y análisis de resultados. Las estrategias empleadas incluyen ajuste fino (*fine-tuning*), fijación de semilla *seed* para asegurar reproducibilidad, técnicas de aumento de datos como *MixUp* y escalado de temperaturas para mejorar la calibración. Finalmente, se desarrolló un sistema de inferencia que combina las predicciones de los modelos del ensamble mediante ponderación adaptativa por paciente, teniendo en cuenta la confianza, la incertidumbre y el desacuerdo entre modelos.

Por último, se implementó una aplicación en `Google Colab` utilizando `Gradio`, que permite cargar imágenes en formato `.svs`, generar los *tiles*, ejecutar el modelo y visualizar las predicciones de forma accesible y transparente. El sistema exporta también las predicciones por modelo y por *tile*, así como indicadores de desacuerdo, lo que facilita un análisis detallado del comportamiento del sistema.

Abstract

Soft tissue sarcomas are rare malignant tumors that can originate in tissues such as muscle, fat or blood vessels, and are characterized by high histologic heterogeneity and aggressive behavior. Due to their low prevalence and inherent heterogeneity, their diagnosis is especially complex and requires expert collaboration at referral centers.

This work proposes an implementation based on deep learning for the classification of the three most frequent subtypes (leiomyosarcoma, dedifferentiated liposarcoma and undifferentiated pleomorphic sarcoma) from digitized histological images, with the aim of serving as a diagnostic support tool or for sarcoma research.

The deployment begins from a previous obsolete project, performing different experiments until reaching a final solution. The most important improvements include the migration to the current versions of `FastAI` and `PyTorch` and the adoption of cross-validation stratified by patient, testing different model assemblies until reaching the final model. For training, a custom-made dataset was generated from the TCGA-SARC set, through a process of image selection and preprocessing to extract 1024×1024 pixel tiles.

Throughout the various experiments to final assembly, the importance of having a representative dataset, employing cross-validation in data-poor settings, and collaborating with an experienced sarcoma pathologist for sample selection and analysis of results was identified. Strategies employed included fine-tuning, seed fixation to ensure reproducibility, data augmentation techniques such as *MixUp* and temperature scaling to improve calibration. Finally, an inference system was developed that combines the predictions of the ensemble models by adaptive weighting per patient, taking into account confidence, uncertainty, and inter-model disagreement.

Finally, an application was implemented in Google Colab using Gradio, which allows images to be loaded in .svs format, tiles to be generated, the model to be run, and predictions to be visualized in an accessible and transparent manner. The system also exports predictions by model and by tile, as well as disagreement indicators, facilitating a detailed analysis of the system's behavior.

A todas las personas que me acompañan en este «viaje».
En especial, a mis padres y a Roberto, por apoyarme de manera incansable en los momentos más difíciles y ayudarme a transformar el miedo en motor.

Índice de figuras

Índice general

Capítulo 1

Introducción

1.1. Marco y motivación del proyecto

El aprendizaje profundo o *Deep Learning* (DL) es un área de la inteligencia artificial que, gracias a su capacidad para detectar patrones complejos en grandes cantidades de datos, ha supuesto un avance significativo en numerosos campos científicos y tecnológicos, destacando especialmente en el ámbito médico. Su capacidad para extraer automáticamente características relevantes de distintos tipos de datos abre grandes posibilidades para diversas aplicaciones, como el diagnóstico clínico.

Los sarcomas de partes blandas (SPB) son tumores malignos poco frecuentes que presentan una gran heterogeneidad histológica y molecular. Esta variabilidad no solo dificulta su diagnóstico preciso, sino que también influye significativamente en el pronóstico y tratamiento de los pacientes. A pesar de los avances tecnológicos y clínicos, la tasa de diagnósticos erróneos en sarcomas sigue siendo alta, tanto a nivel de subtipo como en el reconocimiento general de la enfermedad. Esto se debe principalmente a su baja prevalencia, la marcada heterogeneidad histopatológica entre subtipos y la escasez de profesionales con experiencia en sarcomas.

Este proyecto aborda la necesidad de mejorar la clasificación histopatológica de los sarcomas mediante la aplicación del aprendizaje profundo a imágenes histológicas digitalizadas (*Whole Slide Images*, WSI). El objetivo principal es desarrollar una herramienta que sirva de apoyo en el diagnóstico histopatológico, especialmente en subtipos histológicos complejos. Además, puede ser útil como herramienta de investigación, ya que existen tipos que carecen de características histológicas claramente diferenciables, como el sarcoma pleomórfico indiferenciado, que constituye un grupo de exclusión. Por otro lado, la alta heterogeneidad de los sarcomas se manifiesta no solo entre subtipos distintos, sino también entre pacientes con el mismo subtipo histológico.

1.2. Objetivos técnicos y académicos

Los objetivos de este proyecto son, desde el punto de vista técnico:

- Implementar un sistema de aprendizaje profundo para la clasificación de sarcomas de partes blandas a partir de imágenes histológicas.
- Generar un conjunto de datos propio a partir de imágenes WSI del dataset TCGA-SARC, aplicando criterios de selección de subtipos, preprocesamiento de las WSI y generación de mosaicos (*tiles*).
- Realizar distintos experimentos para comparar distintas estrategias de entrenamiento del modelo.
- Prevenir la fuga de datos entre los conjuntos de datos de entrenamiento y validación mediante partición estratificada por paciente.
- Utilizar servicios de computación y almacenamiento en la nube para el desarrollo, entrenamiento y despliegue del sistema.
- Implementar una aplicación que permita a los patólogos usar el sistema propuesto de forma accesible.

Desde el punto de vista académico, se adquieren las siguientes competencias y habilidades:

- Capacidad de búsqueda y análisis de literatura científica especializada.
- Identificación de los principales desafíos asociados al diagnóstico de sarcomas y planteamiento de soluciones.
- Comprensión de las técnicas empleadas en proyectos actuales para el diagnóstico histopatológico de sarcomas.
- Desarrollo de competencias en el uso de herramientas como PyTorch, FastAI, Python y Gradio, así como en la gestión de datos biomédicos en formato WSI.
- Comprensión de la importancia de elaborar un conjunto de datos grande y representativo para garantizar la validez de los resultados.
- Identificación y prevención de problemas como la fuga de datos, la calibración deficiente del modelo o la necesidad de validación cruzada en contextos con pocos datos.
- Mejora de la capacidad de resolución de problemas, autonomía y toma de decisiones durante el desarrollo, entrenamiento y evaluación del modelo.

1.3. Estructura del resto de la memoria

En los próximos capítulos se desarrolla de forma progresiva el trabajo realizado. El Capítulo 2 presenta el estado del arte. En primer lugar, se introducen los sarcomas de partes blandas, abordando sus principales desafíos diagnósticos desde el punto de vista clínico e histopatológico, así como los retos clínicos y estructurales. A continuación, se revisa el estado del arte de inteligencia artificial aplicada a la medicina, describiendo los principales tipos de aprendizaje automático y profundizando en el contexto histórico y técnico del aprendizaje profundo. Posteriormente, se explican los fundamentos del entrenamiento de redes neuronales y algunas de las estrategias que se utilizarán posteriormente en el desarrollo.

El Capítulo 3 describe el trabajo realizado. En primer lugar, se explica la implementación del estado del arte que sirve como punto de partida. A continuación, se detalla el proceso de generación del dataset, incluyendo la selección y preprocesamiento de los datos y la generación de *tiles*. Por último, se explican cinco experimentos realizados, junto con el análisis de resultados que lleva a la solución final.

En el Capítulo 4 se incluye el presupuesto asumido para la realización de este proyecto, así como una estimación del coste en un entorno profesional.

Por último, el Capítulo 5 presenta el impacto del trabajo y el Capítulo 6 recoge las conclusiones del trabajo y las futuras líneas de desarrollo.

Capítulo 2

Estado del arte

2.1. Sarcomas: visión general

Los sarcomas son un grupo heterogéneo de tumores malignos que se desarrollan en los tejidos estructurales del cuerpo, como el muscular, el óseo, el cartilaginoso, el adiposo y el conectivo. Todos estos tejidos derivan del mesénquima embrionario, lo que define a los sarcomas como tumores de origen mesenquimal [1]. Este origen explica que puedan aparecer en casi cualquier parte del cuerpo, ya que los tejidos mesenquimales están distribuidos por todo el organismo. A diferencia de los carcinomas, que se originan en tejidos epiteliales como la piel, las mucosas o las glándulas y suelen formar estructuras organizadas reconocibles al microscopio, los sarcomas no presentan estructuras epiteliales definidas y su morfología puede ser muy variable.

Representan menos del 1 % de los casos de cáncer en adultos y cerca del 20 % de los tumores malignos pediátricos, considerándose una enfermedad rara al tener una incidencia inferior a 6 casos por cada 100,000 habitantes [1]. Por otra parte, hay más de 150 subtipos reconocidos por la Organización Mundial de la Salud (OMS) [2], muchos de ellos altamente agresivos y cada uno con características moleculares y clínicas diferentes. Esta baja prevalencia y amplia heterogeneidad histológica, entre otros factores, hacen que los sarcomas sean difíciles de estudiar y diagnosticar. Como consecuencia, hasta un 40 % de los pacientes reciben un diagnóstico inicial erróneo y en torno a un 30 % experimenta un diagnóstico tardío [3] [4].

2.2. Sarcomas de partes blandas

Los sarcomas de partes blandas (SPB) constituyen alrededor del 84 % de los casos de sarcoma [1] y afectan a tejidos como los músculos, la grasa, los nervios o los vasos sanguíneos. La mayor parte de los SPB primarios se localizan en las extremidades, ya que la mayor parte de los músculos, vasos sanguíneos y nervios se ubican a lo largo de los brazos y de las piernas [5]. Las siguientes localizaciones más frecuentes son el tronco y el retroperitoneo, y alrededor del 5–10 % en la cabeza y el cuello [1] [6].

2.2.1. Diagnóstico

La forma de presentación más común del sarcoma es la aparición de una masa o bulto. En muchas ocasiones, estas masas son indoloras y su crecimiento puede pasar desapercibido durante semanas o incluso meses. No obstante, cuando el tumor aumenta de tamaño, puede empezar a generar síntomas locales, como dolor por compresión de estructuras vecinas, como nervios, músculos u órganos cercanos. En regiones como el retroperitoneo o la pelvis, donde existe mayor espacio para la expansión tumoral, los SPB pueden alcanzar un tamaño considerable antes de producir signos clínicos evidentes. Esta evolución silenciosa, junto con el hecho de que la mayoría de las tumoraciones de partes blandas son benignas, hace que en muchos casos se diagnostiquen en fases avanzadas [7].

Con el objetivo de mejorar el diagnóstico precoz, se han establecido una serie de criterios de riesgo que, si están presentes, deben alertar al clínico sobre la posibilidad de estar ante un sarcoma. Estos criterios incluyen:

- Tumoración de tamaño superior a cinco centímetros
- Crecimiento rápido
- Localización profunda

En caso de cumplirse alguno de estos requisitos, debe realizarse una resonancia magnética nuclear (RMN) de la zona afectada [8]. Además de confirmar la existencia de la masa tumoral, proporciona información sobre sus características, como forma, densidad y signos de necrosis, y permite evaluar su afectación a las estructuras anatómicas cercanas. La RMN no solo es una prueba útil para el diagnóstico inicial, sino que es fundamental para la planificación tanto de la biopsia como de la cirugía. Si tras realizar la RMN la sospecha de sarcoma se mantiene, es imprescindible que el paciente sea tratado por un equipo multidisciplinar en un centro de referencia con experiencia en sarcomas, ya que está demostrado que mejora el pronóstico sustancialmente [5] [9]. Incluso una biopsia mal realizada puede acarrear consecuencias terapéuticas, quirúrgicas y reconstructivas posteriores [1].

El diagnóstico definitivo se establece mediante el estudio histopatológico de una muestra del tumor obtenida por biopsia para determinar tanto el tipo como el grado histológicos. Esta debe realizarse siempre antes de la resección quirúrgica, ya que el tipo de cirugía es muy diferente en función del grado de malignidad del tumor. Además, se recomienda que la biopsia sea llevada a cabo por el mismo equipo multidisciplinar que va a realizar el tratamiento definitivo, con el fin de evitar complicaciones que puedan comprometer la cirugía posterior, como trayectorias de punción inadecuadas o diseminación local de células tumorales [7] [1].

Finalmente, se completa el diagnóstico con una tomografía axial computarizada (TAC) torácica. Esta prueba permite valorar la presencia de metástasis pulmonares, que constituyen el lugar más frecuente de diseminación a distancia en este tipo de tumores [1].

2.2.2. Histopatología

Actualmente se reconocen más de cien subtipos histológicos distintos, cada uno con características clínicas y moleculares propias [6]. En la Figura 1 se muestra la distribución aproximada de los subtipos histológicos más frecuentes de SPB, destacando la gran proporción de casos agrupados en la categoría «otros», reflejo de la gran heterogeneidad de esta patología. Algunos de los subtipos más frecuentes son el leiomiosarcoma, el liposarcoma y el sarcoma pleomórfico indiferenciado [10].

- **Leiomiosarcoma**. Se origina en células del músculo liso, por lo que puede aparecer en localizaciones muy diversas, como el retroperitoneo, el útero, los vasos sanguíneos y las extremidades. Es uno de los subtipos más agresivos y suele presentar una alta tasa de metástasis hematógena, especialmente al pulmón y al hígado [11].
- **Liposarcoma**. Se desarrolla a partir del tejido adiposo y presenta una gran heterogeneidad clínica, histológica y genética. Se reconocen distintos subtipos dentro de este, siendo los más frecuentes el bien diferenciado, el desdiferenciado y el mixoide [12].
- **Sarcoma pleomórfico indiferenciado**. Se trata de un diagnóstico de exclusión que se utiliza cuando no se puede identificar una línea de diferenciación clara, incluso tras aplicar estudios inmunohistoquímicos y moleculares. Es típicamente un tumor de alto grado, con celularidad alta, pleomorfismo marcado y mitosis abundantes [13].

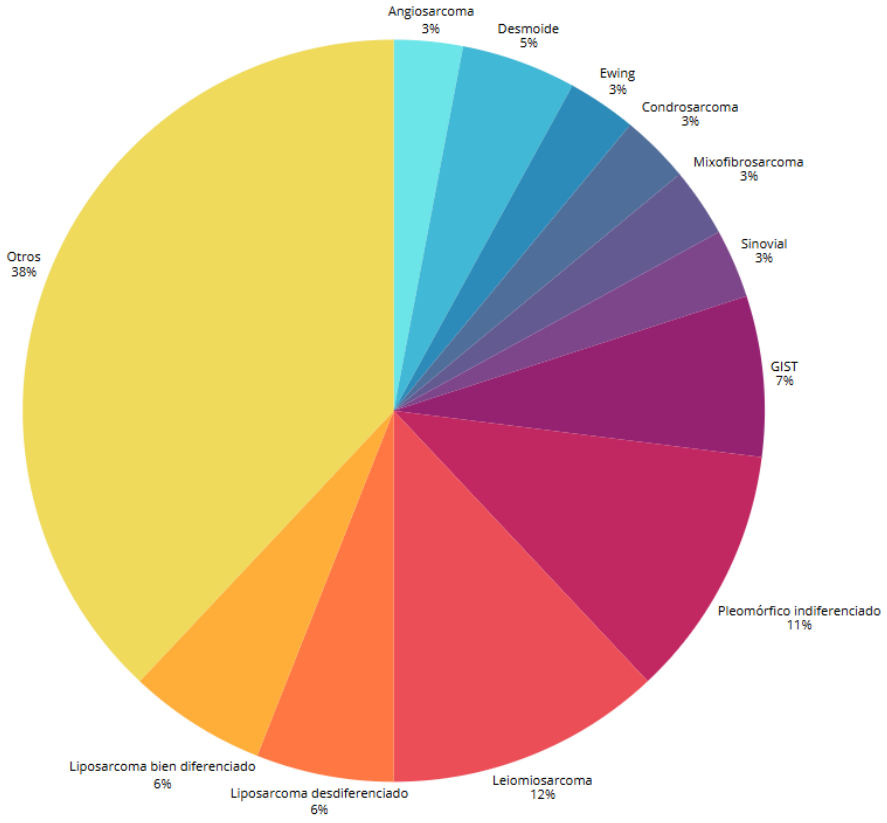


Figura 1: Distribución de histotipos en pacientes adultos con sarcoma de tejidos blandos. Figura adaptada de [10]

Además del subtipo, es fundamental determinar el grado histológico, ya que influye directamente en el pronóstico y en las decisiones terapéuticas. El sistema FNCLCC (*French Fédération Nationale des Centres de Lutte Contre le Cancer*), ampliamente utilizado en Europa, se basa en tres parámetros: diferenciación tumoral, índice mitótico y porcentaje de necrosis. Cada uno se puntúa de 1 a 3, y la suma total define el grado (1–3). Este grado es uno de los parámetros de pronóstico más sólidos de riesgo metastásico y supervivencia global, con tasas de mortalidad a 5 años del 8 % en sarcomas de bajo grado frente a más del 38 % en los de alto grado [14].

El estudio histopatológico convencional comienza con la tinción con hematoxilina-eosina, que permite valorar la arquitectura celular, el patrón de crecimiento, el tipo celular predominante y la presencia de necrosis o mitosis. No obstante, dada la heterogeneidad de los SPB se necesitan técnicas complementarias para establecer un diagnóstico definitivo [15] [10]:

- **Inmunohistoquímica**. Es esencial para diferenciar tumores con morfologías similares mediante la detección de proteínas específicas. Por ejemplo, MDM2 en el liposarcoma desdiferenciado, o desmina y actina en los tumores de estirpe muscular [8].
- **Técnicas moleculares** como FISH o RT-PCR. Existen dos tipos de mutaciones con valor clínico en los sarcomas:
  - **Translocaciones**. Varios tipos de sarcomas muestran translocaciones características, como la translocación *t(X;18)* del sarcoma sinovial. Estas translocaciones provocan la fusión de genes que normalmente se encuentran en lugares separados del genoma, originando moléculas híbridas con funciones nuevas. Dado que estas fusiones génicas son específicas de determinados subtipos y están presentes en prácticamente todos los casos de algunos sarcomas, su identificación no solo tiene implicaciones en la comprensión de la patogenia, sino que también abre grandes posibilidades diagnósticas y terapéuticas [16].
  - **Mutaciones puntuales**. Otro tipo de hallazgo específico de los sarcomas son las mutaciones, como por ejemplo las mutaciones activadoras de c-kit en el GIST (*Gastrointestinal Stromal Tumor*). Aunque algunas mutaciones no tienen interés diagnóstico, pueden determinar el pronóstico de algunos sarcomas [16].

La revisión por patólogos expertos en sarcomas ha demostrado ser fundamental para reducir la tasa de errores diagnósticos. Distintos estudios han documentado discordancias mayores (diagnóstico principal erróneo) en torno al 8–11 % de los casos, y discordancias menores (subtipo incorrecto o discrepancias en el grado histológico) en un 16–35 % [8]. A esto contribuyen su baja incidencia, la enorme heterogeneidad de subtipos y su frecuente indiferenciación.

2.2.3. Tratamiento

El tratamiento principal de los SPB es la cirugía, que debe ser realizada en centros de referencia y con márgenes amplios. Estos márgenes pueden conseguirse mediante una cirugía amplia (resección del tumor con margen de tejido sano) o, en determinados casos, mediante una cirugía radical (existen dos modalidades: amputación de la extremidad y cirugía compartimental, o resección de todo el compartimento muscular) [7].

La radioterapia puede utilizarse tanto de forma preoperatoria (neoadyuvante) con el objetivo de facilitar una cirugía más conservadora y mejorar el control local, como postoperatoria (adyuvante), especialmente en pacientes con alto riesgo de recidiva local [7]. Algunos subtipos histológicos, como el liposarcoma mixoide, el tumor fibroso solitario y el condrosarcoma mixoide extraesquelético, obtienen un mayor beneficio del tratamiento con radioterapia [8].

La quimioterapia igualmente puede aplicarse de forma neoadyuvante o adyuvante. Sin embargo, el beneficio en términos de supervivencia global es limitado,

por lo que su administración solo está justificada en SPB de alto grado y gran tamaño, sobre todo si están localizados en las extremidades [7]. En fases avanzadas, algunos subtipos pueden beneficiarse de quimioterapias dirigidas o ensayos clínicos con el objetivo de ralentizar la progresión tumoral o prolongar la supervivencia [8].

En todos los casos es importante un enfoque multidisciplinar que incluya patólogos, radiólogos, oncólogos médicos, quirúrgicos y ortopédicos, y especialistas en medicina nuclear y en órganos específicos según la localización del tumor [8].

2.2.4. Retos clínicos y estructurales

El diagnóstico y manejo de los SPB presenta diversos desafíos. En muchos casos, la ausencia de sospecha inicial por parte del clínico lleva a retrasos en el diagnóstico o incluso a tratamientos inadecuados, como resecciones incompletas realizadas fuera de centros especializados. Esto no solo dificulta el control local del tumor, sino que se asocia con un peor pronóstico y con un aumento de la tasa de amputaciones [6] [10].

Previamente se han expuesto las dificultades inherentes al diagnóstico histopatológico y la importancia del manejo multidisciplinar en un centro de referencia, pero también existen barreras organizativas que dificultan un primer abordaje adecuado. En España, solo existen ocho CSUR (Centros, Servicios y Unidades de Referencia) para sarcomas: tres en Barcelona, tres en Madrid, uno en Sevilla y uno en Valencia [17]. Esto implica que muchos pacientes tienen que desplazarse para acceder a la atención necesaria. Además, el sistema sanitario está organizado por comunidades autónomas, lo que genera una distribución desigual de recursos diagnósticos, terapéuticos, especialistas y tiempos de espera, siendo un factor importante en el retraso diagnóstico. Por otra parte, aunque existen plataformas de historia clínica compartida, solo algunas comunidades autónomas están integradas, dificultando el acceso a las pruebas previas y ralentizando el proceso de derivación a un CSUR.

La falta de formación específica también representa un obstáculo. Una encuesta a cirujanos realizada en distintos hospitales españoles reveló que solo un 24,3 % había recibido formación especializada en sarcomas [18].

Frente a estas dificultades, los avances tecnológicos ofrecen herramientas para afrontar los desafíos que implica el sarcoma. Por ejemplo, las plataformas de telecomunicación permiten a los pacientes acceder a segundas opiniones especializadas, además de facilitar la formación especializada de los profesionales sanitarios. Por otra parte, la nanotecnología ha demostrado mejorar la entrega selectiva de fármacos, reducir su toxicidad y aumentar la concentración en el tumor. Además, las tecnologías de secuenciación de nueva generación (NGS) facilitan la caracterización molecular, lo que es muy útil para orientar tanto el diagnóstico como el tratamiento personalizado. Por otra parte, la inteligencia artificial (IA) está revolucionando la patología digital, con modelos capaces de extraer información clínicamente relevante a partir de muestras histológicas, lo que abre nuevas vías diagnósticas y terapéuticas [6].

2.3. Inteligencia artificial aplicada a la medicina

El concepto de inteligencia artificial (IA) hace referencia al conjunto de técnicas o algoritmos que permiten a una máquina ejecutar tareas que típicamente se atribuyen a la inteligencia humana, como el aprendizaje o el razonamiento lógico.

Aunque algunas técnicas, como la IA generativa, han emergido recientemente, el concepto de IA se remonta a la década de 1950, cuando Alan Turing —considerado uno de los padres de la ciencia de la computación— publicó un artículo en el que proponía un método teórico para determinar si una máquina era inteligente. Este método, que llamó «juego de imitación» y conocido como *test de Turing*, determinaba que si una máquina era capaz de hacerse pasar por un ser humano en una conversación y engañar a su interlocutor, podía considerarse inteligente [19].

Los primeros enfoques eficaces en el desarrollo de sistemas inteligentes se basaban en inteligencia artificial simbólica o guiada por el conocimiento, como los sistemas expertos. Estos utilizaban reglas básicas del tipo *if-then* codificadas manualmente para imitar el razonamiento humano en tareas concretas. Este enfoque es útil en contextos donde las reglas son claras y bien definidas —como en el desarrollo de un sistema capaz de jugar una partida de ajedrez—, pero la mayoría de los problemas del mundo real presentan una complejidad mucho mayor y no pueden modelarse fácilmente usando reglas explícitas [20].

2.3.1. Aprendizaje automático

Una de las soluciones más efectivas para abordar la necesidad de codificar reglas fue dejar que el propio sistema de inteligencia artificial aprendiera esas reglas. Así surgió el aprendizaje automático o *machine learning* (ML). La idea fundamental es que, al exponer un modelo de aprendizaje automático —es decir, un algoritmo que ajusta sus parámetros a partir de los datos— a una gran cantidad de datos, este sea capaz de identificar patrones relevantes y generalizarlos, de forma que pueda realizar predicciones o tomar decisiones cuando se enfrente a nuevos datos.

Existen cuatro tipos de algoritmos de aprendizaje automático que se emplean en función del contexto para el que se vayan a utilizar: supervisados, no supervisados, semisupervisados y por refuerzo [21]. En el contexto de la salud, se utilizan principalmente el aprendizaje supervisado y el no supervisado [20].

Aprendizaje supervisado

En este enfoque, el modelo recibe como entrada los datos etiquetados, es decir, los datos con sus respectivas salidas esperadas o *ground truth*. El objetivo del modelo es aprender una función que relacione las entradas con las salidas, de forma que sea capaz de generalizar ese conocimiento y aplicarlo correctamente a nuevos datos no vistos durante el entrenamiento.

En el aprendizaje supervisado se siguen distintas fases. En primer lugar, se lleva a cabo la recolección y preparación de datos, que implica obtener datos etiquetados y realizar tareas de preprocesamiento, como normalización o limpieza de datos. A continuación, se divide el conjunto de datos o dataset en tres subconjuntos: entrenamiento, validación y prueba. El modelo se entrena utilizando el primer conjunto, ajustando sus parámetros internos para aprender patrones que relacionen las entradas con las salidas esperadas. De forma paralela, se utiliza el conjunto de validación para supervisar el rendimiento del modelo y ajustar los hiperparámetros con el objetivo de prevenir el sobreajuste (*overfitting*)—ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento y pierde capacidad de generalización—. Por último, el conjunto de prueba (*test set*) —que no ha sido visto por el modelo en ninguna fase anterior— permite evaluar la capacidad real del modelo de generalización ante datos completamente nuevos. Una vez entrenado y validado, el modelo puede utilizarse para inferencia, es decir, para hacer predicciones sobre nuevos casos [22].

Los modelos supervisados pueden entrenarse para resolver distintos tipos de tareas, principalmente clasificación y regresión:

- En un **problema de regresión**, la salida es una variable numérica continua. Por ejemplo, estimar la duración de la estancia hospitalaria de un paciente [20].
- En un **problema de clasificación**, la salida es una etiqueta discreta. Esto permite, por ejemplo, clasificar si una imagen médica corresponde a una lesión benigna o maligna (clasificación binaria), o identificar el subtipo histológico de un tumor a partir de una imagen digitalizada (clasificación multiclase).

Aprendizaje no supervisado

Un modelo de aprendizaje no supervisado recibe los datos sin etiquetas, es decir, sin conocer de antemano cuál es la salida esperada. A diferencia del aprendizaje supervisado, donde se entrena una función que asigna una salida a partir de una entrada, en este caso el modelo trata de descubrir patrones en los datos de entrada sin referencia externa.

Una de las técnicas más útiles de aprendizaje no supervisado es el *clustering*, que consiste en agrupar un conjunto de datos en función de su similitud según una determinada métrica. Este enfoque no solo resulta útil cuando no se dispone de datos etiquetados, sino también como herramienta para descubrir patrones ocultos. Por ejemplo, en el caso de la identificación del subtipo histológico de un tumor, el modelo podría separar los datos en más categorías tumorales de las conocidas, lo que podría permitir descubrir matices diagnósticos relevantes que podrían haber pasado desapercibidos ante el ojo humano [20].

No obstante, los algoritmos de aprendizaje no supervisado son menos precisos y fiables que los métodos supervisados, precisamente por la complejidad de resolver un problema sin ninguna indicación previa [20].

Algoritmos de aprendizaje supervisado

El aprendizaje supervisado es el enfoque más empleado en medicina. A continuación se describen los algoritmos más representativos.

- **Naïve Bayes**: algoritmo de clasificación basado en el teorema de Bayes, que asume independencia entre las características de entrada. A partir de un conjunto de datos etiquetado, el modelo estima la probabilidad de cada clase (por ejemplo, paciente con o sin diabetes) y la probabilidad de observar cada característica (como el nivel de glucosa o la edad) en cada clase. Durante la fase de inferencia, calcula la probabilidad de pertenecer a cada clase combinando las probabilidades individuales de las características, y asigna la clase con la probabilidad más alta.
  Es uno de los algoritmos de clasificación más utilizados por su simplicidad y eficiencia en conjuntos de datos pequeños, aunque suponer independencia entre variables puede limitar su rendimiento en problemas complejos [22].

  En el ámbito médico se ha utilizado con éxito en la detección de enfermedades cutáneas a partir de imágenes, alcanzando precisiones superiores al 91 %, y en la detección de enfermedades cardíacas, con una precisión del 88,16 % [23].
- **Support Vector Machines (SVM)**: algoritmo utilizado tanto en tareas de clasificación como de regresión, aunque es más empleado en clasificación. Representa los datos en un espacio multidimensional, donde cada dimensión es una característica, y busca el hiperplano óptimo que separa las clases.
  SVM destaca por su eficacia en entornos de alta dimensionalidad [22].

  Ha sido utilizado para la predicción temprana de insuficiencia cardíaca, alcanzando precisiones entre el 57,85 % y el 91,83 % [23].
- **Árboles de decisión**: representa el proceso de toma de decisiones en una estructura jerárquica en forma de árbol mediante reglas o condiciones basadas en las características de entrada. En cada etapa del entrenamiento se dividen los datos de entrada en subconjuntos más homogéneos según el valor de una determinada característica a evaluar. Cada nodo interno del árbol representa una condición sobre una de estas características —por ejemplo, «¿la glucosa es mayor a 150?»—, cada rama refleja el resultado de dicha condición y cada nodo hoja indica una clase o valor numérico de salida, en función de si se trata un problema de regresión o de clasificación. Una de sus principales ventajas es su simplicidad e interpretabilidad, pero tiende al sobreajuste [22].

  Los árboles de decisión se han utilizado ampliamente para el diagnóstico y la predicción de enfermedades cardiovasculares. En un estudio se predijo la presencia de enfermedad cardíaca con un 88 % de precisión utilizando solo ocho variables clínicas [23].

- **K-Nearest Neighbors (K-NN)**: este algoritmo, más utilizado en tareas de clasificación que en regresión, no construye un modelo explícito, sino que clasifica las nuevas muestras comparándolas directamente con los datos de entrenamiento. Para ello, cuando se presenta un dato nuevo en inferencia, el algoritmo calcula la distancia entre este y todos los ejemplos del conjunto de entrenamiento, y selecciona los K vecinos más próximos. La clase asignada será aquella que aparezca con mayor frecuencia entre esos K vecinos. Entre sus ventajas destacan su facilidad de implementación, su capacidad para manejar múltiples clases y su buen rendimiento en ciertos problemas donde las clases están bien separadas. No obstante, K-NN es sensible a atributos irrelevantes y es costoso computacionalmente debido a los cálculos de distancias realizados [22].

  En medicina, K-NN ha sido utilizado con éxito en tareas de detección y diagnóstico de enfermedades, como en la predicción de enfermedades cardíacas. Algunos estudios han combinado K-NN con algoritmos genéticos o redes neuronales para mejorar su precisión diagnóstica, logrando en ciertos casos un rendimiento superior al de modelos más complejos [23].
- **Redes neuronales y aprendizaje profundo**: el aprendizaje profundo utiliza redes neuronales artificiales compuestas por unidades interconectadas llamadas neuronas, inspiradas en el sistema nervioso humano —aunque de una forma muy simplificada—. Estas redes están organizadas en múltiples capas ocultas para extraer características progresivamente más profundas de los datos.

  Este enfoque es especialmente útil para capturar patrones no lineales y relaciones complejas entre variables, incluso en datos no estructurados. Por esta razón, se han convertido en la técnica de referencia en tareas como la clasificación de imágenes médicas o la segmentación de tejidos tumorales. No obstante, su entrenamiento requiere un alto coste computacional [22].

2.3.2. Aprendizaje profundo

El aprendizaje profundo o *deep learning* (DL) es una subárea dentro del aprendizaje automático, como se observa en la Figura 2, que se basa en el uso de redes neuronales artificiales (RNA), formadas por neuronas artificiales organizadas en múltiples capas consecutivas. Estas redes pueden alcanzar una gran profundidad al incorporar múltiples capas ocultas, lo que da lugar al término de aprendizaje profundo.

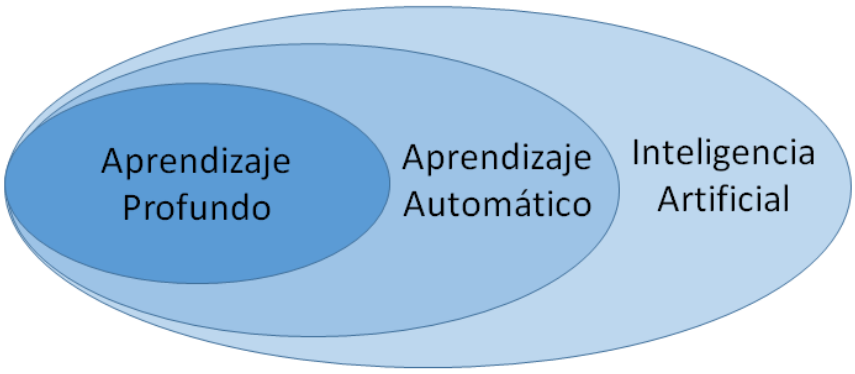


Figura 2: Áreas de la inteligencia artificial [24]

Sistema nervioso

Antes de abordar los modelos matemáticos que se han desarrollado basándose en la estructura del cerebro, resulta útil repasar brevemente el funcionamiento básico de una neurona biológica.

Las neuronas son las unidades funcionales fundamentales del sistema nervioso, especializadas en el procesamiento y transmisión de señales entre células. Desde un punto de vista estructural, se distinguen cuatro partes principales [25], como se muestra en la Figura 3.

- **Dendritas**: prolongaciones ramificadas que reciben la mayoría de las señales entrantes desde otras neuronas.
- **Soma**: el cuerpo celular, donde se integran las señales recibidas.
- **Axón**: una extensión larga que transmite los impulsos eléctricos desde el soma hasta las terminaciones del axón.
- **Sinapsis**: puntos de conexión que permiten la transmisión de señales a otras neuronas, a través de procesos eléctricos y químicos.

La comunicación entre neuronas se produce mediante un mecanismo electroquímico: cuando la señal recibida por una neurona supera un determinado umbral de activación, se genera un potencial de acción que viaja a lo largo del axón y provoca la liberación de neurotransmisores en la sinapsis, excitando así a otras neuronas y propagando la señal a través de la red neuronal biológica.

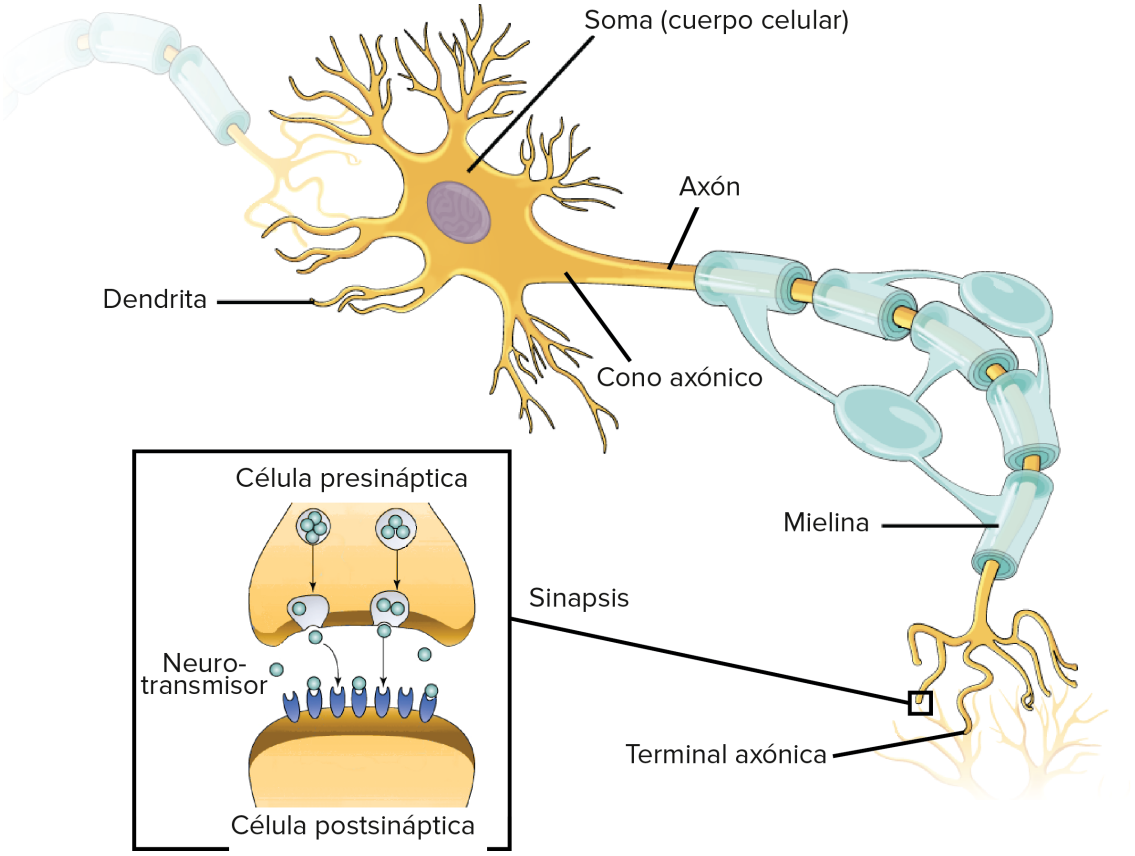


Figura 3: Estructura de una neurona [26]

Hasta finales del siglo XIX, se concebía el cerebro como un tejido continuo, formado por una única red interconectada de células nerviosas. Esta idea, conocida como doctrina reticular, fue cuestionada en 1888 por Santiago Ramón y Cajal, quien, utilizando técnicas de tinción desarrolladas por Camillo Golgi, demostró que las células nerviosas no formaban una red continua, sino que estaban separadas entre sí. Este hallazgo sentó las bases de la llamada doctrina de la neurona, según la cual la neurona es la unidad funcional y estructural del sistema nervioso [25].

El descubrimiento de Ramón y Cajal no solo transformó la neurociencia, sino que también abrió la puerta al desarrollo de modelos computacionales inspirados en el comportamiento de las neuronas biológicas. Uno de los primeros fue el modelo de McCulloch y Pitts (1943).

Modelo neuronal de McCulloch-Pitts

Warren McCulloch —neurólogo y cibernético estadounidense—, motivado por la demostración de Alan Turing de que su máquina teórica —conocida hoy día como máquina de Turing— podía resolver cualquier problema matemático formulable como un algoritmo, se propuso demostrar que dicho modelo computacional podía, a su vez, implementarse mediante un conjunto finito de estructuras inspiradas en el funcionamiento de las neuronas biológicas.

En 1942, McCulloch conoció a Walter Pitts, un brillante lógico estadounidense. Ambos colaboraron en la redacción del artículo que sentaría las bases del modelo neuronal más influyente de la época: «*A Logical Calculus of Ideas Immanent in Nervous Activity*», publicado en 1943 [25]. En este artículo, McCulloch y Pitts formularon un modelo lógico que aproximaba el comportamiento de una neurona desde una perspectiva computacional, partiendo de cinco hipótesis fundamentales sobre el sistema nervioso [25]:

1. **La actividad de la neurona sigue un principio todo o nada**. Supone que tanto las señales que le llegan a la neurona como la señal generada van a ser de tipo binario: o 1 o 0. Por lo tanto, la neurona es una función binaria que transforma un conjunto de valores booleanos en otro valor booleano.
2. **Para que una neurona se active, un número fijo de sinapsis debe ser excitado dentro de un intervalo de tiempo; este número es constante e independiente de la actividad previa o la posición de la neurona**. Cada neurona tiene asociado un umbral fijo de activación. Si el número de señales excitadoras que recibe la neurona es igual o superior a ese umbral, la neurona se activa y produce una salida de valor 1. En caso contrario, la salida será 0. Una consecuencia directa de esta formulación es que, si el umbral supera al número total de señales de entrada posibles, la neurona nunca podrá activarse, permaneciendo inactiva permanentemente.
3. **El único retardo significativo en el sistema nervioso es el retardo sináptico**.
4. **La activación de una sinapsis inhibidora bloquea completamente la activación de la neurona**. Si una de las entradas a la neurona corresponde a una señal inhibidora, entonces la salida será forzosamente 0, independientemente del número de señales excitadoras presentes.
5. **La estructura de la red neuronal permanece constante en el tiempo (no hay aprendizaje ni plasticidad en el modelo)**. El modelo de McCulloch-Pitts no contempla mecanismos de ajuste o aprendizaje. Las conexiones entre neuronas, así como los pesos (que en este caso son todos iguales) y los umbrales, son fijos. Por tanto, una vez definida la red, su comportamiento es completamente determinista y no varía con la experiencia o con los datos de entrada.

Este modelo neuronal puede representarse como se muestra en la Figura 4, donde una neurona recibe cuatro señales de entrada: X_1, X_2, X_3 y X_4. Cada una de ellas puede tomar el valor 0 o 1, y puede ser de tipo excitador o inhibidor. La neurona está asociada a un umbral de excitabilidad θ, y su salida se denota por la variable y.

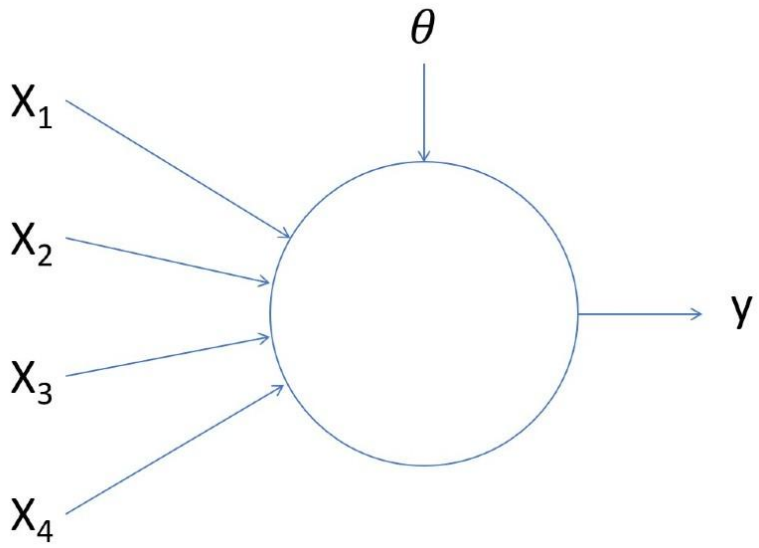


Figura 4: Representación de la neurona de McCulloch-Pitts [25]

En base a las hipótesis anteriores, el funcionamiento de la neurona sigue las siguientes reglas:

1. Si alguna de las entradas X_i es inhibidora y tiene valor 1, la salida será forzosamente $y = 0$, con independencia del resto de señales.
2. En ausencia de señales inhibidoras activas, la neurona suma las señales excitadoras presentes.
3. El valor resultante de la suma se compara con el umbral θ:
   - Si la suma es mayor o igual que θ, entonces la neurona se activa y $y = 1$.
   - Si la suma es menor que θ, entonces la neurona no se activa y $y = 0$.

Esta última conversión —la que convierte el resultado de la suma de las entradas en un 1 o en un 0 en función de que se alcance o no el umbral de excitación de la neurona— es lo que se conoce como función de activación, ya que determina si la neurona se va a activar (generando una señal de 1) o no (en cuyo caso generará una señal de 0) [25]. Podemos representar esta función como se muestra en la Figura 5.

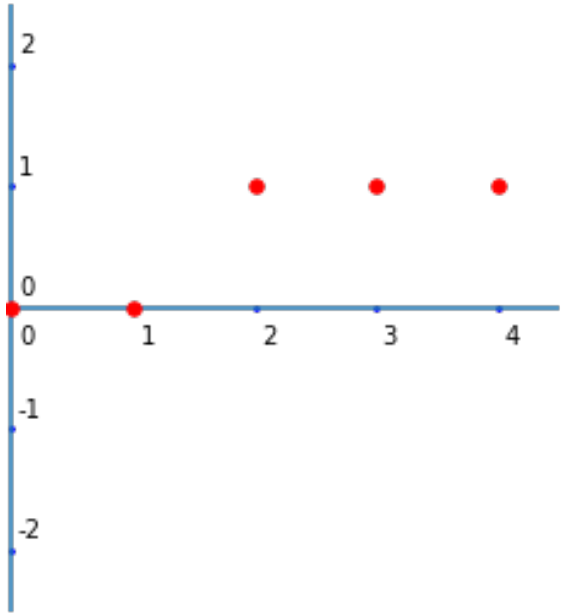


Figura 5: Representación de la función de activación [25]

En el eje horizontal se representa la suma de las señales de entrada, y en el eje vertical se representa la salida generada por la neurona. En este ejemplo, la neurona tiene un umbral $\theta = 2$. De esta forma, si la suma de las señales de entrada es menor que 2 —es decir, 0 o 1—, la salida de la neurona será $y = 0$. Por otro lado, si esta suma es igual o superior a 2, la salida será $y = 1$.

La función que describe este comportamiento se conoce como «función escalón», ya que presenta un valor constante de 0 hasta alcanzar el umbral, a partir del cual la salida pasa abruptamente a 1, como se muestra en la Ecuación 2.1.

$$F(x) = \begin{cases} 0, & \text{si } x < 0 \\ 1, & \text{si } x \geq 0 \end{cases} \tag{2.1}$$

Entonces, podemos representar la neurona de McCulloch-Pitts como en la Figura 6. En este nuevo diagrama se ha separado una de las entradas, $\overline{X}$, representando una posible señal inhibidora activa. El número de señales inhibidoras es irrelevante, ya que el comportamiento de la neurona será el mismo: si al menos una está activa, la salida será 0, independientemente del resto de entradas. Se asume, por tanto, que el resto de señales X_i son excitadoras.

En el interior del esquema de la neurona se ilustran las dos operaciones que rigen su funcionamiento. Primero se realiza la suma ($\sum$) de las señales de entrada excitadoras, siempre que no haya señales inhibidoras activas. Luego, se aplica la función de activación que compara este valor con el umbral de activación θ. Si la suma es mayor o igual que θ, la salida será 1; en caso contrario, será 0.

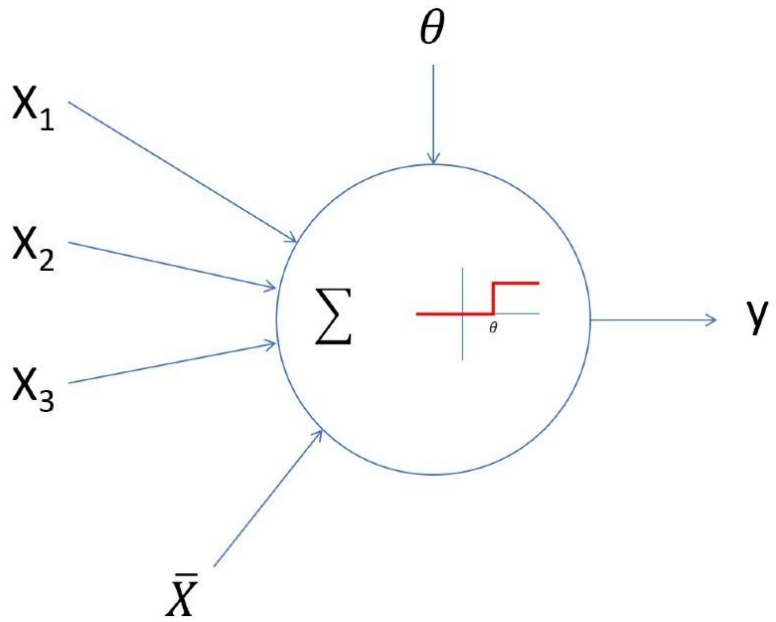


Figura 6: Representación de la neurona de McCulloch-Pitts con función de activación [25]

Cabe destacar que la primera operación aplicada a las entradas —la suma de las señales excitadoras— es una función lineal, ya que consiste simplemente en sumar los valores de entrada (todos con el mismo peso). Por el contrario, la segunda operación ——la función escalón— es no lineal, ya que transforma abruptamente el resultado anterior en una salida binaria (0 o 1) según si se alcanza o no el umbral de activación θ. Es en esta combinación de funciones lineales y no lineales donde reside gran parte de la potencia de cálculo de las neuronas artificiales [25].

El modelo de neurona de McCulloch-Pitts despertó rápidamente el interés de la comunidad científica, ya que permitía simular funciones lógicas booleanas mediante configuraciones simples de entradas excitadoras e inhibidoras. A continuación, se presentan algunos ejemplos representativos [25].

- **Función AND**. La puerta lógica AND devuelve 1 únicamente cuando todas las entradas tienen valor 1. Para implementarla con una neurona de McCulloch-Pitts, basta utilizar dos entradas excitadoras y fijar el umbral de activación en 2. De este modo, la salida será 1 solo si ambas entradas están activas simultáneamente, como se observa en la Figura 7.

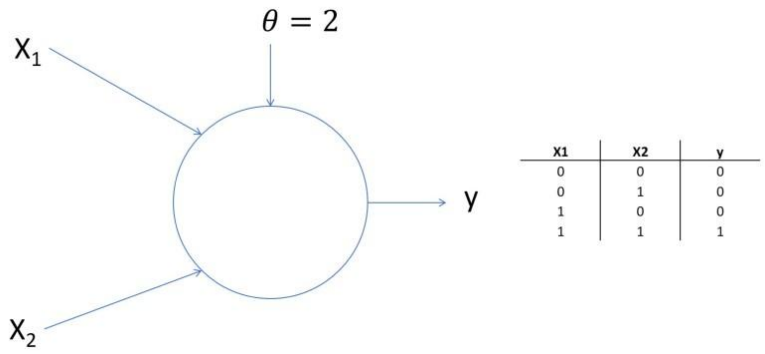


Figura 7: Función AND [25]

- **Función OR**. La función OR produce una salida de 1 si al menos una de las entradas tiene valor 1, como se muestra en la Figura 8. Para simularla, se pueden utilizar dos entradas excitadoras y establecer el umbral en 1. Es decir, basta con que una única entrada esté activa para que la neurona se active.

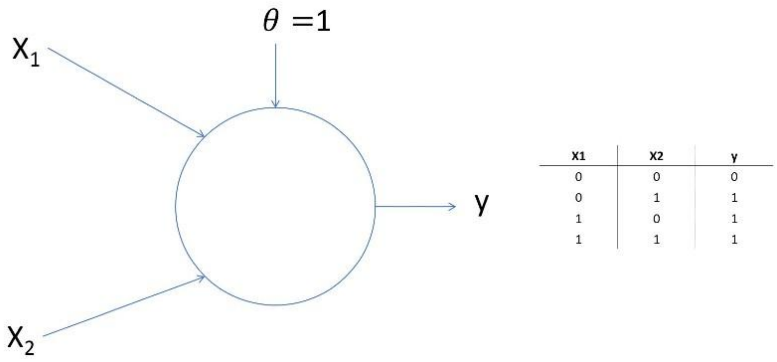


Figura 8: Función OR [25]

- **Función NOT**. Para modelar una función NOT, que invierte el valor de la entrada (produce 1 si la entrada es 0, y 0 si es 1), se emplea una entrada inhibidora como señal de entrada y una segunda entrada excitadora de valor constante 1. El umbral se fija en 1. De esta forma, si la entrada inhibidora tiene valor 0, la entrada constante activa la neurona, como se observa en la Figura 9.

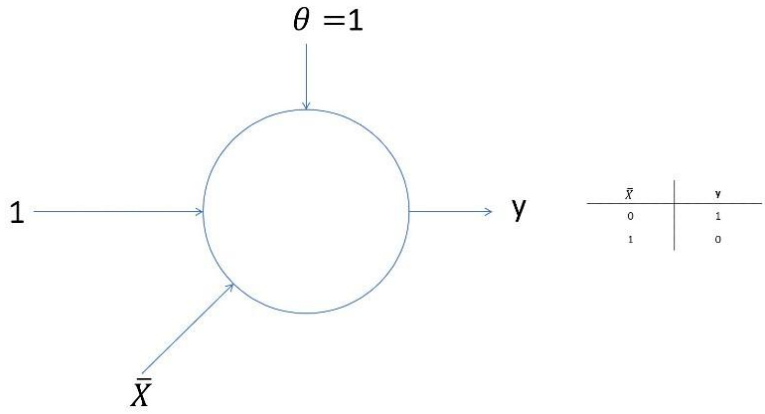


Figura 9: Función NOT [25]

- **Función NOR**. La función NOR produce una salida de 1 únicamente cuando todas las entradas son 0, como se muestra en la Figura 10. Para simularla, se puede emplear una pequeña red de dos neuronas. La primera implementa una función OR, detectando si hay alguna entrada activa, y la segunda recibe la salida de la primera como señal inhibidora junto con una entrada constante de valor 1, reproduciendo el comportamiento de una NOT.

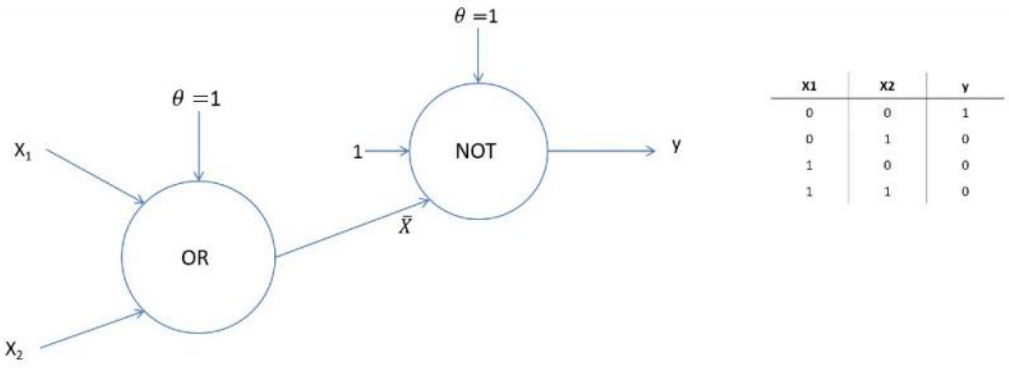


Figura 10: Función NOR [25]

En los anteriores ejemplos, la neurona recibe dos entradas excitadoras, X_1 y X_2, que pueden tomar los valores 0 o 1. Esto da lugar a cuatro combinaciones posibles: $[0, 0]$, $[0, 1]$, $[1, 0]$ y $[1, 1]$, que pueden representarse como puntos en el plano. Al representar gráficamente estos puntos y calcular la suma $X_1 + X_2$, se observa que es posible dividir el plano en dos regiones separadas por la recta definida por el umbral de activación. Esta recta incluye todos los puntos cuya suma de entradas es igual al umbral, y actúa como una frontera que distingue los casos en los que la neurona se activa (salida = 1) de aquellos en los que no lo hace (salida = 0) [25].

A partir de las distintas configuraciones del umbral de activación θ, se pueden simular diferentes funciones lógicas.

Por ejemplo, si el valor del umbral es 0, se estaría definiendo una línea formada por aquellos puntos en los que la suma de sus coordenadas X e Y sea 0, como se observa en la Figura 11 [25]. En este caso, las cuatro posibles combinaciones de X_1 y X_2 quedarían sobre la recta o por encima de esta. En consecuencia, todas las combinaciones posibles activan la neurona, que produce una salida de valor 1.

Este comportamiento equivale a la función lógica TRUE, donde la salida es siempre 1, independientemente de los valores de entrada.

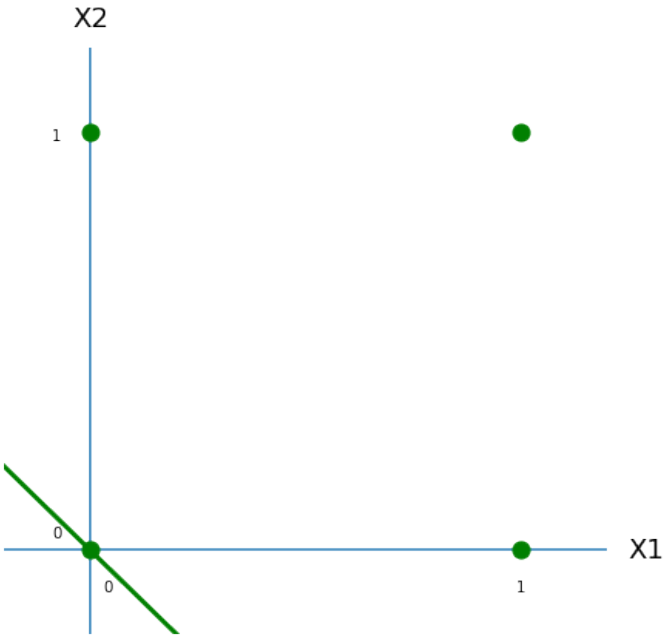


Figura 11: $\theta = 0$: función TRUE [25]

Si el umbral se establece en $\theta = 1$, la división de los puntos del plano es la representada en la Figura 12. Los puntos representados en color verde corresponden a combinaciones que se encuentran en la línea divisoria o por encima de esta, y generan una salida igual a 1. Es decir, se activan todas aquellas configuraciones en las que al menos una de las entradas toma el valor 1. El único punto representado en color rojo es $[0, 0]$, que se encuentra por debajo de la recta y no activa la neurona, generando por tanto una salida de valor 0.

Esta configuración reproduce el comportamiento de la función OR, en la que la salida es 1 cuando al menos una de las entradas sea igual a 1.

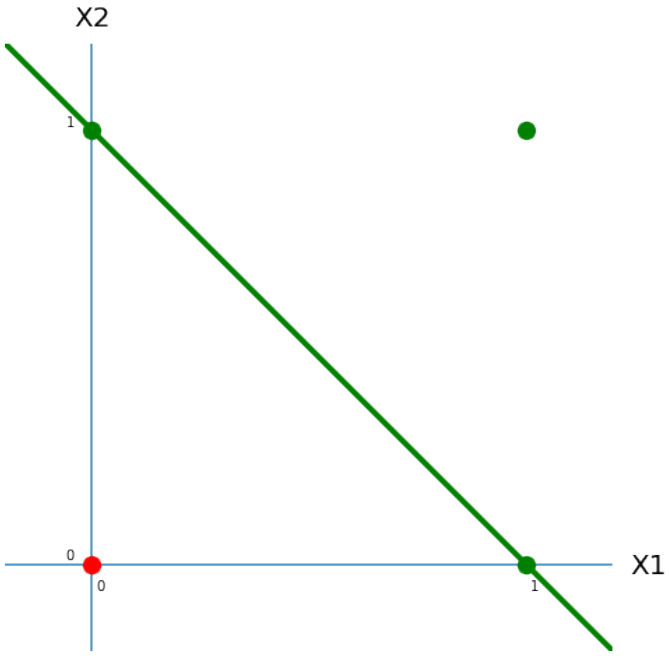


Figura 12: $\theta = 1$: función OR [25]

Si el umbral es $\theta = 2$, la única combinación que queda sobre la recta o por encima de esta es la correspondiente a los valores de entrada [1, 1], como se muestra en la Figura 13.

En este caso, la neurona se estaría comportando como una función AND, generando una señal de salida de valor 1 solo cuando ambas entradas tomen el valor 1 [25].

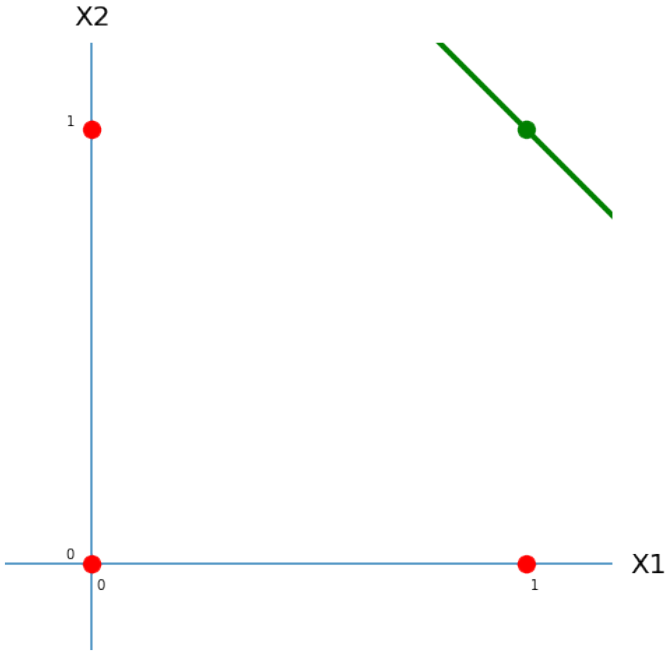


Figura 13: $\theta = 2$: función AND [25]

Limitaciones de la neurona de McCulloch-Pitts

Como se ha visto anteriormente, la neurona de McCulloch-Pitts actúa como un clasificador lineal, ya que separa los puntos del plano en dos regiones según si la suma de las entradas supera o no el umbral de activación. En todos los casos, la frontera de decisión es una recta, lo que impide aplicar este método de clasificación a escenarios no separables linealmente [25].

Un ejemplo que muestra esta limitación es la función XOR (OR exclusivo), que devuelve 1 cuando solo una de las entradas es igual a 1, y 0 cuando ambas son iguales (ya sea 0 o 1). Si representamos en el plano las combinaciones posibles de entrada, no existe ninguna recta que permita separar correctamente los puntos con salida 1 de los que tienen salida 0, como se observa en la Figura 14.

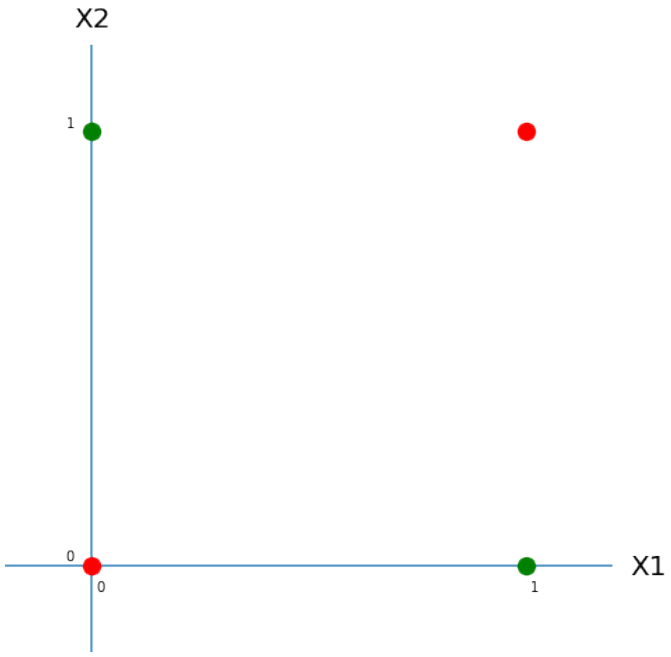


Figura 14: Ejemplo de función no separable linealmente: función XOR [25]

En conclusión, la neurona de McCulloch-Pitts no puede simular cualquier función lógica, y en particular, no es capaz de representar funciones que no sean linealmente separables, como ocurre con la función XOR.

A continuación, se resumen algunas de las limitaciones más relevantes del modelo de McCulloch-Pitts [25]:

- Solo admitía entradas booleanas (valores 0 o 1).
- Todas las entradas excitadoras tenían el mismo peso.
- La salida de la neurona era también booleana.
- La función de activación era siempre una función escalón.
- Solo podía representar funciones linealmente separables, por lo que no era posible simular, por ejemplo, una función XOR.
- El modelo no incluía un mecanismo para la configuración de las neuronas. Es decir, dado un conjunto de entradas y salidas deseadas y un conjunto de neuronas conectadas, no se especificaba cómo ajustar los parámetros (umbrales, conexiones, estructura) para que la red de neuronas se comportara según esos datos. Este proceso, conocido actualmente como entrenamiento, no fue definido por estos autores.

A pesar de estas limitaciones, la neurona de McCulloch-Pitts representó un hito fundamental en la historia de la inteligencia artificial. Introdujo por primera vez una abstracción computacional inspirada en el sistema nervioso capaz de realizar operaciones lógicas, y sentó las bases teóricas sobre las que se construirían las redes neuronales artificiales modernas.

Aprendizaje de Hebb

En 1949, el psicólogo canadiense Donald Hebb propuso en su obra «*The Organization of Behavior*» un principio fundamental que revolucionó la forma en la que se entendían las neuronas artificiales: la Regla de Hebb. Este principio sostenía que, cuando una neurona participa de forma repetida en la activación de otra, la importancia de la conexión entre ambas —lo que en redes artificiales se denomina peso sináptico— aumenta. Esta idea suele resumirse con la frase: «neuronas que se activan juntas, permanecen juntas», aunque Hebb subrayó que esta relación debía ser causal, es decir, que la primera neurona debía activarse antes que la segunda para que se produjera el refuerzo de la conexión.

Para que un modelo como el de McCulloch-Pitts pudiera incorporar este tipo de aprendizaje, sería necesario que las conexiones entre neuronas tuvieran pesos variables capaces de modificarse con la experiencia. Este enfoque se conoce como aprendizaje de Hebb, y anticipó uno de los pilares del entrenamiento de redes neuronales artificiales [25].

El perceptrón de Frank Rosenblatt

Otro hito destacado —y a menudo más conocido que el modelo de McCulloch-Pitts— fue el desarrollo del perceptrón, propuesto por el psicólogo estadounidense Frank Rosenblatt entre 1957 y 1962. El modelo del perceptrón fue publicado en 1958 en el artículo «*The Perceptron: A probabilistic model for information storage and organization*» y se presentaba como un dispositivo capaz de aprender a partir de datos, lo que lo convirtió en uno de los primeros modelos de redes neuronales entrenables [25].

El objetivo inicial de Rosenblatt era desarrollar una máquina capaz de clasificar imágenes siguiendo un enfoque de aprendizaje inspirado en la regla de Hebb. La primera implementación del perceptrón no fue un dispositivo físico, sino un programa ejecutado en un IBM 704 en la Universidad de Cornell, desarrollado en 1957, un año antes de la publicación formal de su propuesta.

No fue hasta 1962 cuando Rosenblatt obtuvo los recursos necesarios para construir una versión física del modelo, conocida como «*Mark I Perceptron*». Este dispositivo fue diseñado específicamente como un clasificador de imágenes. Para ello, contaba con una matriz de 400 fotosensores dispuestos en una cuadrícula de 20×20, lo que equivalía a procesar imágenes de 20×20 píxeles. Durante el entrenamiento, el operador mostraba diferentes imágenes a estos sensores —por ejemplo, fotografías de hombres y mujeres—, y el perceptrón generaba una predicción. Una vez emitida la predicción, el operador indicaba manualmente si había sido correcta o incorrecta mediante un conjunto de interruptores. En función de esta señal de error, los pesos sinápticos del sistema eran ajustados mediante potenciómetros controlados por motores, que modificaban los valores según el resultado obtenido [27].

El Mark I Perceptron estaba estructurado en tres componentes funcionales principales: las unidades sensoras (*S-units*), las unidades asociativas (*A-units*) y las unidades de respuesta (*R-units*), como se observa en la Figura 15.

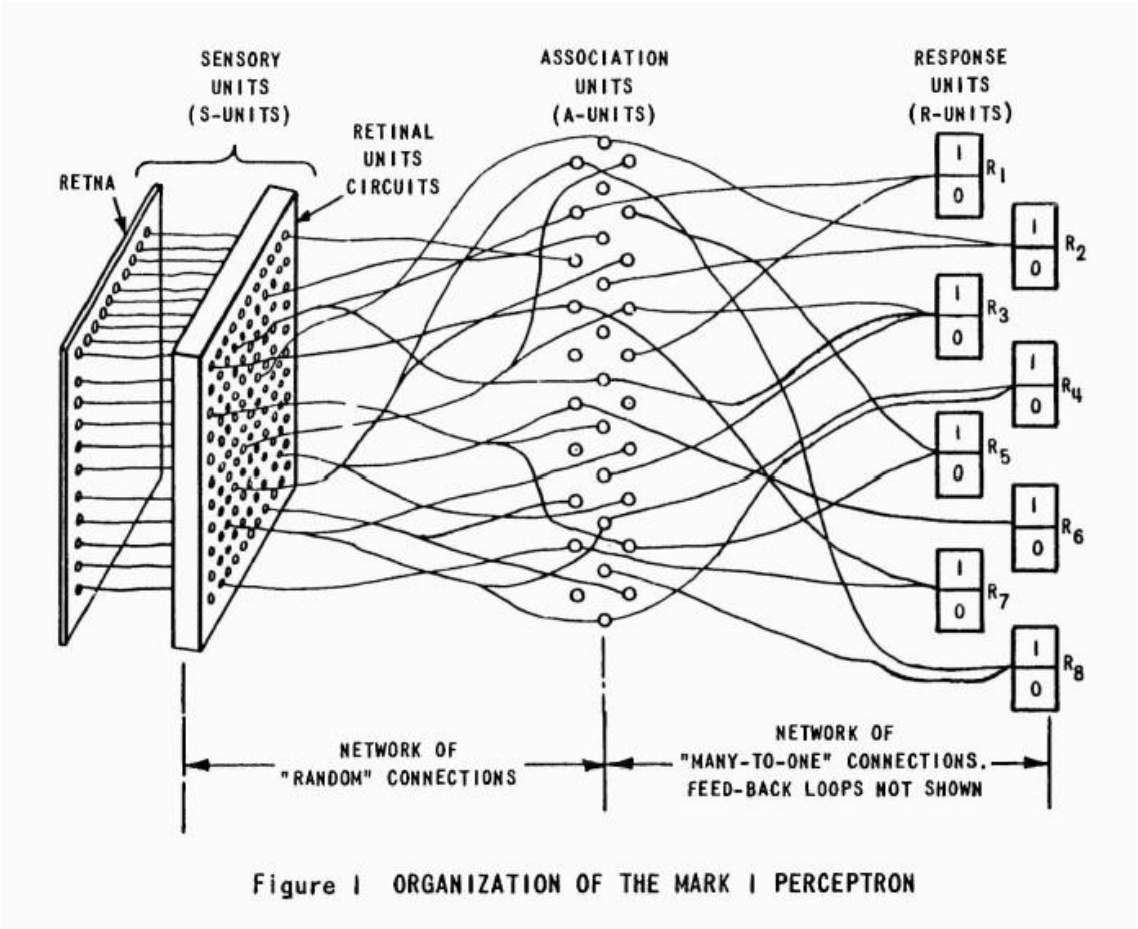


Figura 15: Organización básica del *Mark I Perceptron* [25]

El enfoque propuesto por Frank Rosenblatt introdujo un nuevo tipo de neurona artificial, que puede representarse mediante el esquema de la Figura 16.

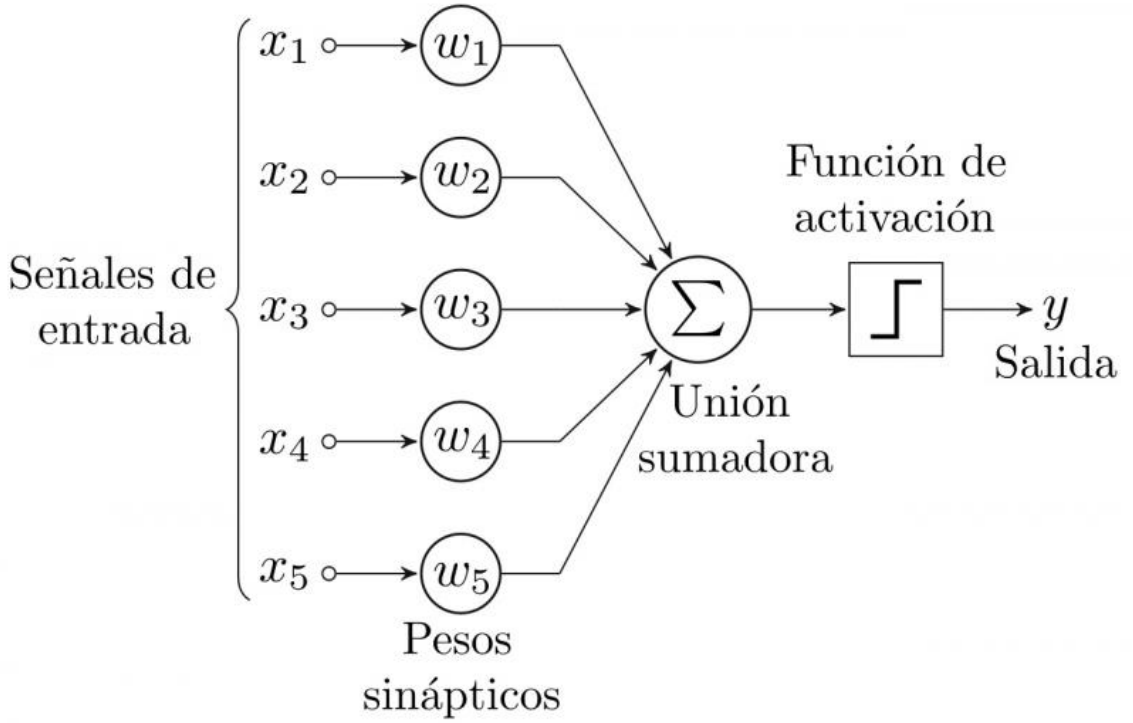


Figura 16: Representación de la neurona de Frank Rosenblatt [25]

A diferencia del modelo de McCulloch-Pitts, en el que todas las entradas excitadoras tenían el mismo peso, cada conexión de entrada en esta neurona tiene asociado un peso específico. Cada señal de entrada es multiplicada por su peso correspondiente, y a continuación se realiza la suma ponderada de todas esas señales. Si el resultado de esta suma iguala o supera el umbral de activación θ, la neurona se activa y genera una salida de valor 1; en caso contrario, la salida es 0. Este comportamiento se modela mediante una función escalón aplicada sobre la suma ponderada.

En esencia, el funcionamiento es similar al de la neurona de McCulloch-Pitts, pero con la diferencia clave de que los pesos son ajustables, lo que permite entrenar el modelo a partir de datos. Desde el punto de vista matemático, el modelo puede expresarse como se indica en la Ecuación 2.2, donde x_i son las entradas y w_i los pesos asociados a cada entrada.

$$\text{Salida} = \begin{cases} 1, & \text{si } \sum_i w_i \cdot x_i \geq \theta \\ 0, & \text{si } \sum_i w_i \cdot x_i < \theta \end{cases} \tag{2.2}$$

Esta ecuación puede simplificarse al pasar el umbral θ al lado izquierdo de la desigualdad y reescribiendo $-\theta$ como un término independiente b, conocido como sesgo o *bias*, tal y como se muestra en la Ecuación 2.3. En esencia, el umbral θ y el sesgo b son dos formas equivalentes de expresar lo mismo: el punto a partir del cual la neurona se activa. Sin embargo, a diferencia del modelo de McCulloch-Pitts, en este caso b es entrenable.

$$\text{Salida} = \begin{cases} 1, & \text{si } \sum_i w_i \cdot x_i + b \geq 0 \\ 0, & \text{si } \sum_i w_i \cdot x_i + b < 0 \end{cases} \tag{2.3}$$

Este tipo de unidad computacional es lo que hoy se conoce como perceptrón, nombre que puede referirse tanto a esta neurona individual como al dispositivo físico desarrollado por Rosenblatt.

Limitaciones de la neurona de Frank Rosenblatt

El modelo propuesto por Rosenblatt dio lugar a una arquitectura que hoy se conoce como red neuronal de tipo *feedforward* o prealimentada, es decir, una red en la que las señales se propagan en una única dirección, desde las unidades de entrada hacia las de salida, sin retroalimentación ni ciclos.

El perceptrón de Rosenblatt se correspondía con una red de una sola capa, en la que las neuronas estaban dispuestas en paralelo y conectadas directamente a las entradas. Este tipo de arquitectura, conocido como perceptrón de capa única, es capaz únicamente de clasificar datos linealmente separables. En 1969, Marvin Minsky y Seymour Papert publicaron el libro «*Perceptrons*», donde analizaron en profundidad estas limitaciones. Entre otras conclusiones, demostraron que los perceptrones de una sola capa no podían resolver la función lógica XOR, lo que evidenciaba que este modelo era insuficiente para abordar problemas más complejos.

Estas críticas frenaron durante años el interés y la financiación en el campo de las redes neuronales, período conocido como «el invierno de la inteligencia artificial» [25].

Redes neuronales

Las limitaciones del perceptrón llevaron a los investigadores a plantear que conectar varias neuronas artificiales organizadas en capas podía aumentar considerablemte la capacidad de cómputo del sistema [25].

Esta idea dio lugar al desarrollo de las redes neuronales artificiales, en las que las neuronas se organizan en capas consecutivas: una capa de entrada, una o varias capas ocultas y una capa de salida. La disposición de estas capas y cómo estén conectadas las neuronas entre sí determinan la arquitectura de la red neuronal.

La arquitectura más sencilla dentro de las redes neuronales es la conocida como Perceptrón Multicapa o *Multilayer Perceptron* (MLP), un tipo de red neuronal prealimentada (*feedforward*). Sin embargo, es importante destacar que el término perceptrón en esta denominación se debe únicamente a motivos históricos [25]. En realidad, un MLP no está compuesto por perceptrones, sino por neuronas sigmoideas o por neuronas con otro tipo de función de activación, como se verá más adelante.

Una de las razones por las que el modelo original de perceptrón resultó insuficiente fue precisamente la limitación de la función escalón, aplicada sobre la suma ponderada de las entradas —es decir, el producto escalar entre los valores de entrada y los pesos, junto con el sesgo—. Al no ser derivable, esta función impedía el uso de técnicas basadas en cálculo diferencial, como el descenso del gradiente, lo que dificultaba el entrenamiento eficiente de redes con múltiples capas.

Para superar esta barrera, se introdujeron funciones de activación continuas y derivables, siendo una de las primeras y más utilizadas la función sigmoide, definida por la Ecuación 2.4.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{2.4}$$

Aunque existen muchas arquitecturas más complejas, como las redes convolucionales, la mayoría mantiene la estructura general del MLP. Por ello, puede considerarse una base conceptual sobre la que se construyen muchas de las redes neuronales modernas. Una neurona de este tipo, con una función de activación sigmoidal, puede representarse como en la Figura 17.

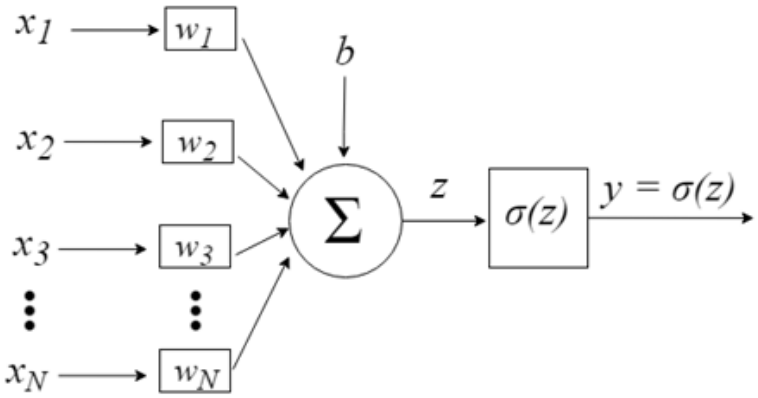


Figura 17: Representación del perceptrón [28]

La Figura 18 muestra la estructura general de una red neuronal de tipo *feedforward*, como el MLP. Se distinguen una capa de entrada, varias capas ocultas y una capa de salida. Este esquema básico sigue presente, con variaciones, en muchas arquitecturas modernas más complejas, como las redes neuronales convolucionales.

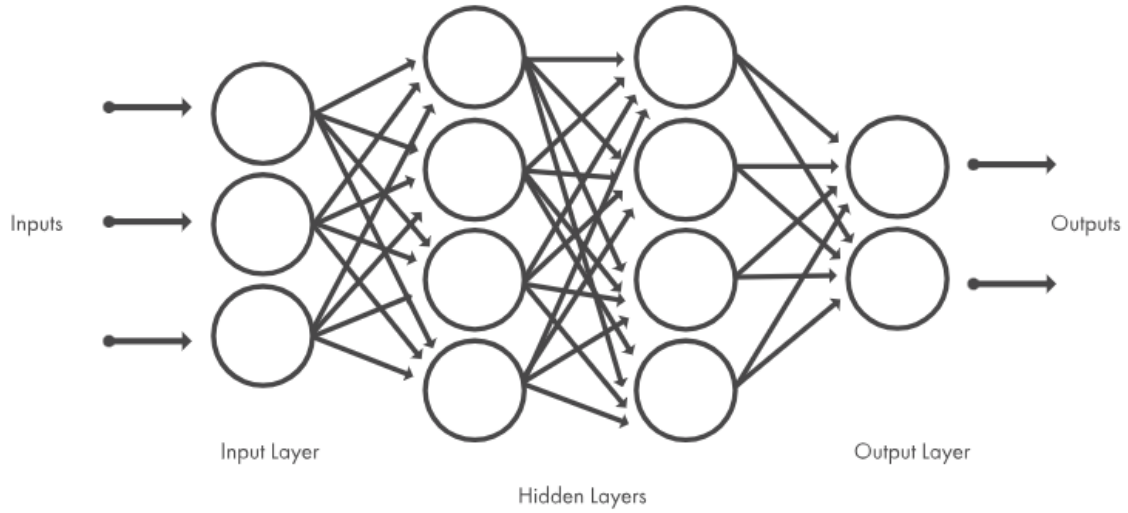


Figura 18: Arquitectura de red neuronal típica [29]

La capa de entrada, situada a la izquierda, simplemente sirve como interfaz para introducir los datos en la red. Aunque se represente con neuronas, esta capa no realiza ningún cálculo, simplemente transmite los datos de entrada al resto de la red. El número de neuronas de esta capa deberá coincidir con el número de valores de cada muestra a analizar.

A continuación, se sitúan una o varias capas ocultas. En el caso del MLP, estas están formadas por neuronas conectadas densamente (es decir, cada neurona recibe como entrada todos los valores de la capa anterior). El número de capas ocultas o neuronas a utilizar depende en gran medida del tipo de datos con los que se entrene la red neuronal y del problema específico que se desea resolver. En la práctica, estas decisiones suelen abordarse mediante métodos heurísticos, que consisten en entrenar y validar el modelo utilizando distintas combinaciones de valores para los hiperparámetros que se quieren ajustar [25].

Finalmente, la capa de salida contiene una o más neuronas, dependiendo del tipo de tarea: una para regresión o clasificación binaria, o varias en el caso de clasificación multiclase.

Entrenamiento de redes neuronales

Para que una red neuronal sea capaz de realizar predicciones útiles, deben cumplirse tres condiciones fundamentales: contar con datos adecuados, definir una arquitectura apropiada y ajustar correctamente los parámetros de la red —es decir, los pesos y los sesgos—. Esta última etapa, conocida como entrenamiento, consiste en encontrar los valores de esos parámetros que minimizan el error cometido por la red al hacer predicciones sobre los datos de entrenamiento.

Función de error

Para cuantificar qué tan buena es una determinada configuración de estos parámetros, se define una función de error —también conocida como función de pérdida (*loss function*)—, que mide la diferencia entre las predicciones de la red y los valores reales esperados.

Una forma común de definir esta función en problemas de regresión es mediante la suma de los errores cuadráticos entre las predicciones $\hat{y}_i$ y los valores objetivo y_i, como se indica en la Ecuación 2.5.

$$C = \sum_i (\hat{y}_i - y_i)^2 \tag{2.5}$$

Dado que cada salida $\hat{y}_i$ depende de los pesos y sesgos de todas las neuronas involucradas —como se observa en la Figura 19—, la función de error se convierte en una expresión no lineal y compleja en función de estos parámetros, como se muestra en la Ecuación 2.6.

$$C = (\sigma\,(w_{31} \cdot \sigma(w_{11} \cdot x_{1i} + w_{12} \cdot x_{2i} + b_1) + w_{32} \cdot \sigma(w_{21} \cdot x_{1i} + w_{22} \cdot x_{2i} + b_2) + b_3) - y_i)^2 \tag{2.6}$$

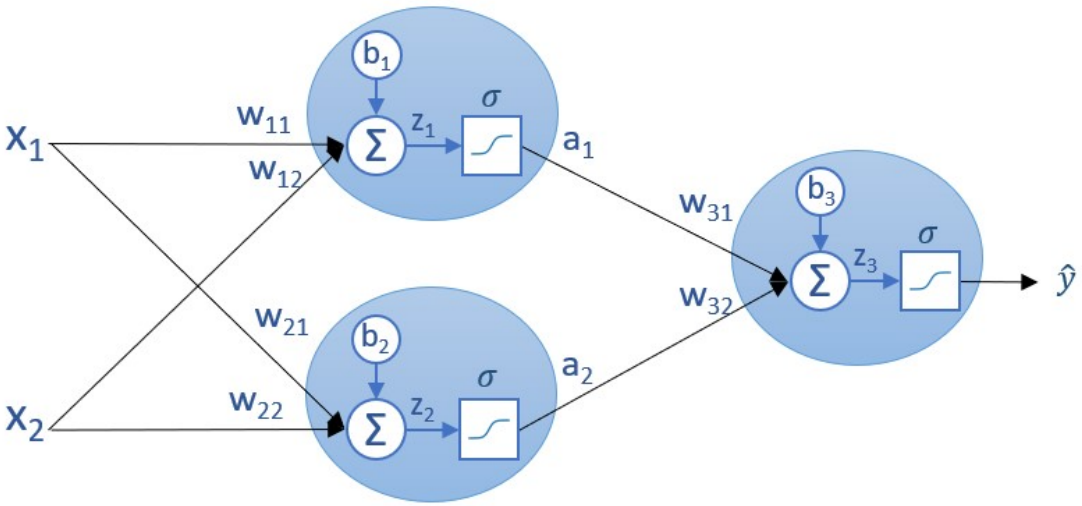


Figura 19: Esquema red neuronal [25]

Búsqueda del mínimo de la función de error
Una vez definida la función de error en función de los pesos y sesgos de la red, el objetivo del entrenamiento consiste en encontrar los valores de estos parámetros que minimicen dicha función sobre el conjunto de entrenamiento.

Una posible solución es recurrir al cálculo diferencial, calculando los puntos donde la pendiente es cero, es decir, donde la derivada se anula. Sin embargo, para funciones de muchas variables —como ocurre en las redes neuronales, donde la función de error puede depender de miles o millones de parámetros—, el cálculo de la derivada puede ser extremadamente complejo, pues implica el cálculo de las derivadas parciales de cada una de las variables de la función [25]. En su lugar, se utilizan algoritmos iterativos que permiten aproximar el mínimo de la función de error paso a paso. Estos algoritmos se conocen como optimizadores.

Optimizadores
Para ilustrar este enfoque, supongamos una red neuronal con solo dos parámetros (por ejemplo, un peso y un sesgo) y el error cometido para distintas combinaciones de valores. El resultado puede visualizarse como una superficie tridimensional, como se muestra en la Figura 20, donde los dos ejes horizontales corresponden a los valores de los parámetros a y b, y el eje vertical al error asociado a cada combinación.

La tarea del optimizador consiste en encontrar, a través de una secuencia de pasos, el punto de esa superficie donde el error alcanza su valor mínimo. Para ello, calcula el gradiente —vector que contiene las derivadas parciales de la función de error respecto a cada parámetro— y ajusta sus valores en la dirección que reduce el error. Este proceso se repite de forma iterativa hasta acercarse a un mínimo.

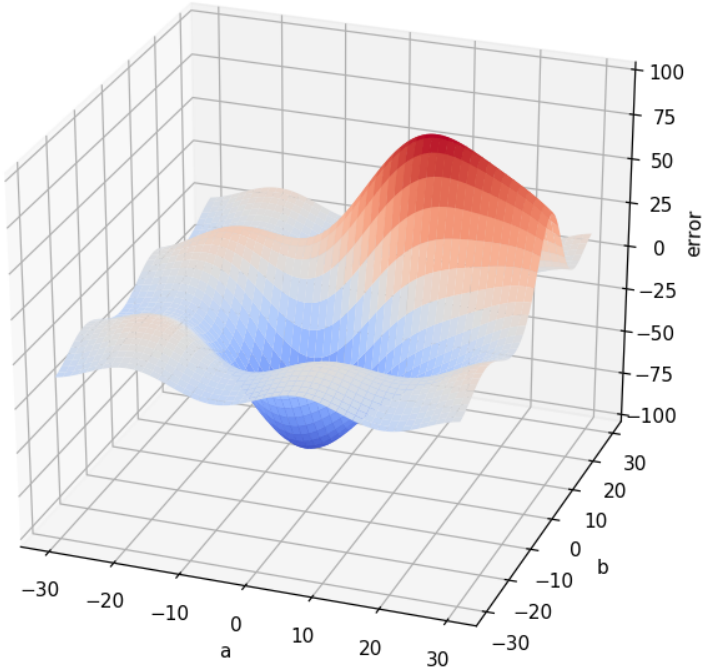


Figura 20: Representación tridimensional del error según a y b [25]

Descenso del gradiente
El descenso del gradiente (*Gradient Descent* o *Batch Gradient Descent*) es un algoritmo iterativo —optimizador— que permite encontrar los valores de los parámetros de una red neuronal que minimizan la función de error. Dado un punto cualquiera del espacio de parámetros (por ejemplo, una combinación de pesos y sesgos), el algoritmo calcula la dirección y magnitud del gradiente de la función de error: la dirección indica por dónde el error aumenta más rápidamente, y la magnitud indica cuánto aumenta. El algoritmo da un pequeño paso en la dirección opuesta al gradiente, con el objetivo de reducir el error. Este proceso se repite hasta alcanzar un mínimo.

En redes neuronales profundas, el cálculo del gradiente no se realiza directamente para cada parámetro, sino que se utiliza un procedimiento llamado retropropagación del error (*backpropagation*). Este algoritmo aplica la regla de la cadena para propagar el error desde la salida hacia las capas anteriores, y calcular así las derivadas parciales necesarias para actualizar todos los pesos y sesgos de la red.

Tasa de aprendizaje
Para controlar el tamaño de ese paso se utiliza un factor de escala conocido como tasa de aprendizaje (*learning rate*), representado como μ o η. Si este valor es demasiado alto, el algoritmo puede saltarse el mínimo o volverse inestable; si es demasiado bajo, el proceso será muy lento. En la práctica, se utilizan valores pequeños como 0,01 o 0,001, aunque existen métodos para ajustarlo automáticamente durante el entrenamiento.

Por ejemplo, si el valor de un parámetro a es 10 y su derivada parcial es 7, restarla directamente daría un salto hasta $a = 3$, lo que puede ser excesivo. Al aplicar una tasa de aprendizaje, se modula el ajuste, como se muestra en la Ecuación 2.7 [25].

$$a \leftarrow a - \mu \frac{\partial f}{\partial a} \tag{2.7}$$

Como resultado de este proceso, los parámetros de la red se ajustan en una dirección que reduce ligeramente el error cometido. Sin embargo, una sola iteración por los datos de entrenamiento —conocida como época (*epoch*)— no suele ser suficiente para que la red aprenda correctamente. El proceso suele repetirse durante múltiples épocas, hasta que la red converge a una configuración que minimiza la función de error y generaliza bien sobre nuevos datos.

Validación del modelo
Durante el entrenamiento se utiliza el conjunto de validación para evaluar el rendimiento del modelo y ajustar los hiperparámetros. Una práctica común es dividir los datos en dos subconjuntos: uno para el entrenamiento y otro para la validación. Sin embargo, cuando el tamaño del dataset es limitado, este enfoque puede no ser el más adecuado, ya que tener pocos datos de validación reduce la capacidad para estimar correctamente el rendimiento del modelo y empeora su capacidad de generalización.

La validación cruzada (*cross-validation*) es una estrategia común para mitigar este problema. Consiste en dividir el dataset en varias particiones o *folds* y entrenar el modelo k veces, de forma que en cada iteración se utiliza una partición diferente como conjunto de validación y las restantes como entrenamiento. Así, cada *fold* se emplea una vez como validación y $k - 1$ veces como entrenamiento, lo que permite validar el rendimiento del modelo con una mayor diversidad de datos. Sin embargo, esta estrategia implica un aumento considerable del tiempo de entrenamiento y del coste computacional, ya que es necesario entrenar el modelo desde cero en cada uno de los *k folds*.

Existen diferentes variantes, como *K-fold*, en la que el dataset se divide en K particiones (*folds*) del mismo tamaño, o la validación cruzada estratificada (*Stratified K-fold*), que garantiza que cada *fold* tenga una proporción de clases similar a la del dataset original. Sin embargo, estas técnicas no tienen en cuenta la estructura jerárquica del dataset cuando las muestras, por ejemplo, están agrupadas por paciente. En estos casos, es posible que imágenes de un mismo paciente aparezcan tanto en entrenamiento como en validación, lo que corrompería la validación por introducir fuga de datos (*data leakage*). Para evitar este problema, se puede utilizar una variante conocida como *Stratified Group K-fold*, que garantiza que todos los datos correspondientes a un mismo grupo (por ejemplo, un paciente) estén en una única partición en cada iteración.

Para evaluar correctamente la capacidad de generalización de un modelo, es fundamental realizar una evaluación externa tras el entrenamiento, utilizando datos no vistos durante el proceso de entrenamiento (incluyendo la validación). Esto es importante porque, aunque un modelo pueda mostrar un buen rendimiento sobre los conjuntos de entrenamiento y validación, podría estar sobreajustado y no generalizar correctamente a datos nuevos.

Métricas de evaluación
En problemas de clasificación multiclase existen varias métricas que permiten evaluar distintos aspectos del comportamiento del modelo. A continuación se describen las más habituales:

- **Exactitud (*Accuracy*)**. Es la proporción de predicciones correctas sobre el total de muestras.
- **Matriz de confusión**. Es una tabla que resume las predicciones del modelo frente a las etiquetas reales. Permite identificar qué clases se confunden con más frecuencia. En problemas multiclase, se representa como una matriz cuadrada donde las filas indican las clases reales y las columnas las predichas. Los valores en la diagonal corresponden a los aciertos.
- **Precisión, sensibilidad y F1-*score***. Estas métricas se calculan para cada clase, y son especialmente útiles cuando hay clases minoritarias:
  - **Precisión (*Precision*)**: proporción de verdaderos positivos entre todas las predicciones positivas para una clase. Indica cuántas de las predicciones fueron correctas.
  - **Sensibilidad (*Recall*)**: proporción de verdaderos positivos entre todos los ejemplos reales positivos. Indica cuántos de los positivos reales fueron detectados.

    - **F1-*score***: media armónica entre precisión y sensibilidad. Es una medida equilibrada del rendimiento cuando ambas métricas son igualmente importantes.
- **Curva ROC y AUROC**. La curva ROC (*Receiver Operating Characteristic*) representa la relación entre la tasa de verdaderos positivos (sensibilidad) y la tasa de falsos positivos a medida que se modifica el umbral de decisión aplicado sobre la probabilidad predicha por el modelo para una determinada clase. El área bajo esta curva (AUROC) cuantifica la capacidad del modelo para asignar puntuaciones más altas a las muestras de la clase correcta que a las del resto. Cuanto más se aproxima a 1, mayor es el poder discriminativo del modelo; un valor cercano a 0.5 indica un rendimiento similar al azar. En problemas multiclase, se genera una curva ROC para cada clase, y a partir de los AUROC individuales se pueden calcular métricas resumen:
    - **AUROC por clase**. Mide el rendimiento del modelo al distinguir correctamente una clase concreta del resto (estrategia *one-vs-rest*).
    - **AUROC macro-promediado**. Calcula la media de los AUROC por clase, tratando todas las clases por igual.
    - **AUROC micro-promediado**. Considera conjuntamente todas las predicciones y otorga más peso a las clases más frecuentes.

Otros optimizadores

El algoritmo descrito hasta ahora corresponde al conocido como descenso de gradiente, en el que se utiliza todo el conjunto de entrenamiento para calcular el gradiente antes de actualizar los parámetros de la red. Aunque este enfoque proporciona una estimación precisa de la dirección óptima en la que deben ajustarse los parámetros para minimizar el error global, también presenta una desventaja importante: cada época es computacionalmente costosa, ya que implica procesar todas las muestras del dataset antes de poder actualizar los parámetros.

Para hacer frente a esto, se han desarrollado variantes del descenso por gradiente. A continuación, se describen algunos de los más representativos [25]:

- **Descenso de gradiente estocástico (*Stochastic Gradient Descent*)**. Actualiza los parámetros de la red tras procesar una única muestra seleccionada aleatoriamente del conjunto de entrenamiento.
  Este enfoque reduce drásticamente el coste computacional por iteración, aunque introduce más variabilidad en el recorrido hacia el mínimo. Sin embargo, esto puede ser útil para evitar mínimos locales.
- **Descenso de gradiente por mini lotes (*Mini-batch Gradient Descent*)**. Calcula el gradiente a partir de pequeños subconjuntos de datos (*mini-batches*), lo que permite un compromiso entre eficiencia y estabilidad. Es el enfoque más usado en la práctica. El tamaño del lote (*batch size*) es un hiperparámetro clave.
- ***Momentum* (*Stochastic Gradient Descent with Momentum*)**. Este optimizador añade «inercia» a las actualizaciones, de forma que, en lugar de depender únicamente del gradiente actual, mantiene una media móvil exponencial de los gradientes pasados. Así, las actualizaciones en los parámetros tienden a seguir la dirección general del descenso acumulado, lo que permite suavizar

las oscilaciones y acelerar la convergencia hacia el mínimo.

- **Gradiente acelerado de Nesterov (*Nesterov Accelerated Gradient*)**. Es una variante del algoritmo *Momentum* que anticipa la dirección del siguiente paso, mejorando la estabilidad y velocidad de convergencia.
- **AdaGrad. Algoritmo de gradiente adaptativo (*Adaptative Gradient Algorithm*)**. Se trata de una modificación del descenso de gradiente estocástico que adapta individualmente la tasa de aprendizaje para cada parámetro en función de cómo ha evolucionado su gradiente a lo largo del entrenamiento. Para ello, acumula el cuadrado de los gradientes anteriores de cada parámetro. Cuanto mayor ha sido esta acumulación, menor será la tasa de aprendizaje que se aplique sobre ese parámetro.
  Esta estrategia permite que los parámetros con grandes variaciones iniciales se actualicen de forma más prudente en el futuro, lo que puede mejorar la eficiencia al comienzo del entrenamiento. Sin embargo, este mecanismo tiene un inconveniente: si el gradiente acumulado crece demasiado rápido, la tasa de aprendizaje puede llegar a ser tan pequeña que el parámetro apenas se modifique, lo que limita la capacidad de seguir ajustándolo en fases posteriores del entrenamiento.
- **RMSProp (*Root Mean Square Propagation*)**. Es una mejora de AdaGrad, evitando que la tasa de aprendizaje disminuya en exceso. En lugar de acumular indefinidamente el cuadrado de todos los gradientes pasados, RMSProp utiliza una media móvil exponencial que da mayor peso a los gradientes más recientes. Esto permite ajustar dinámicamente la tasa de aprendizaje de cada parámetro sin que llegue a hacerse demasiado pequeña, facilitando una convergencia más estable incluso en entrenamientos largos.
- **Adam (*Adaptive Moment Optimization*)**. Combina *Momentum* y RMSProp. Calcula dos medias móviles exponenciales: una de los gradientes (como *Momentum*) y otra de los cuadrados de los gradientes (como RMSProp). Estas dos estimaciones permiten ajustar dinámicamente la dirección y la magnitud de las actualizaciones, manteniendo tasas de aprendizaje adaptativas por parámetro y una trayectoria de optimización más estable.

La elección del optimizador tiene un papel crucial en el entrenamiento de redes neuronales, ya que determina cómo se actualizan los parámetros y con qué rapidez se aproxima la red a una solución óptima. Un buen optimizador puede reducir significativamente el número de iteraciones necesarias, acelerar la convergencia y mejorar la estabilidad del proceso de entrenamiento, mientras que una elección inadecuada puede provocar oscilaciones, estancamiento o sobreajuste [25].

Redes neuronales convolucionales

Aunque el perceptrón multicapa puede utilizarse para problemas generales de clasificación o regresión, presenta limitaciones al trabajar con datos estructurados espacialmente, como las imágenes.

Las CNN están diseñadas para trabajar con datos que tienen estructura espacial, como ocurre con las imágenes. Su arquitectura se basa en una serie de capas que, de forma progresiva, extraen patrones visuales cada vez más complejos a partir de los píxeles. Actualmente son la herramienta de referencia para clasificación de imágenes, detección de objetos o segmentación de regiones anatómicas en medicina. Sus componentes principales son [24]:

- **Capas de convolución**: aplican filtros (también llamados *kernels*) que recorren la imagen extrayendo patrones locales como bordes, texturas o formas. Cada filtro se ajusta automáticamente durante el entrenamiento y actúa como detector de una característica específica.
- **Capas de activación no lineal (ReLU)**: se insertan tras cada convolución para introducir no linealidad en el modelo, lo que permite aprender representaciones complejas.
- **Capas de agrupamiento (*pooling*)**: reducen la resolución espacial de los mapas de activación, conservando la información más relevante. Esta operación disminuye el número de parámetros y mejora la invariancia ante pequeñas variaciones en la posición de los patrones detectados.
- **Capas completamente conectadas (*fully connected*)**: situadas en las etapas finales, integran la información extraída y permiten realizar la predicción final, por ejemplo, asignando una clase a una imagen.
- **Capa de salida y función de activación *softmax***: en tareas de clasificación multiclase, como la planteada en este proyecto, la última capa de la red suele ser una capa completamente conectada con tantas neuronas como clases. Esta capa genera una salida conocida como *logits*, que son valores reales sin restricción de rango. Para convertir estos valores en probabilidades interpretables, se aplica la función *softmax*, definida por:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{2.8}$$

  Siendo z_i el *logit* correspondiente a la clase i y K el número total de clases, la transformación garantiza que todas las salidas estén en el rango [0, 1] y que su suma sea exactamente 1. Esto permite interpretar el resultado como la probabilidad asignada a cada clase.
  Por otra parte, a diferencia de la función escalón, que se utilizaba en modelos neuronales antiguos, *softmax* es continua, derivable y adecuada para el entrenamiento mediante descenso del gradiente.

La operación básica que da nombre a estas redes neuronales es la convolución. Se trata de una operación matemática que aplica un filtro sobre pequeñas regiones de la imagen, multiplicando cada valor por el correspondiente en el kernel y sumando el resultado, como se observa en la Figura 21. Este proceso se repite desplazando el filtro por toda la imagen, lo que permite generar mapas de características que resumen la presencia de distintos patrones visuales. Durante el entrenamiento, la red aprende automáticamente los valores óptimos de estos filtros para maximizar su capacidad de detección.

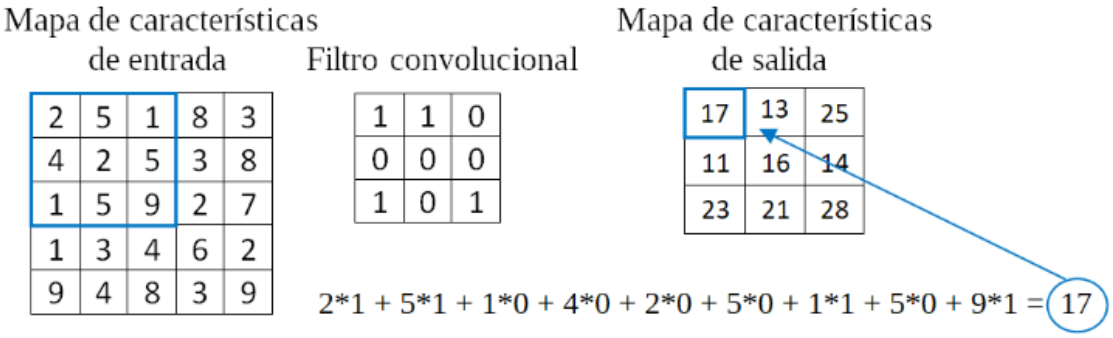


Figura 21: Convolución de matrices para el filtrado de imágenes [24]

El número de filtros, así como su tamaño y profundidad, afectan directamente a la capacidad del modelo y al tiempo necesario para entrenarlo. A mayor profundidad, mayor número de patrones complejos puede aprender, aunque también crece el riesgo de sobreajuste y la necesidad de datos.

2.3.3. Aprendizaje profundo para el diagnóstico y pronóstico de sarcomas de partes blandas

Aunque el aprendizaje profundo se ha aplicado con éxito en distintos ámbitos de la oncología, la mayoría de las implementaciones se han centrado en tumores de alta prevalencia, como cáncer de pulmón, mama, próstata o colorrectal. En contraste, su uso en tumores raros, como los sarcomas, ha sido más limitado, en parte por la escasez de datos, la alta heterogeneidad tumoral y la dificultad para obtener cohortes suficientemente representativas.

Uno de los principales ámbitos de aplicación es el análisis de imágenes médicas. Modelos basados en radiómica y aprendizaje profundo han logrado distinguir tumores benignos y malignos, como lipomas frente a liposarcomas bien diferenciados, con rendimientos comparables a los de radiólogos expertos [30]. También se han desarrollado modelos predictivos y nomogramas que integran características morfológicas, como el tamaño o los márgenes, para evaluar el riesgo de malignidad en lesiones de partes blandas [30].

En histopatología digital, se han empleado arquitecturas como AlexNet, ResNet, DenseNet o YOLOv4 para clasificar subtipos como el rabdomiosarcoma o detectar automáticamente figuras mitóticas. Además, modelos del tipo U-Net se han usado para segmentar tumores en imagen médica, facilitando la planificación de tratamientos como la radioterapia [31]. Algunos trabajos han explorado también el uso del aprendizaje profundo para evaluar la respuesta al tratamiento a través de cambios internos en la composición del tumor, incluso cuando el tamaño no varía [30].

En el ámbito pronóstico, un modelo basado en imágenes histológicas fue capaz de predecir tanto la supervivencia global, como la supervivencia libre de metástasis con una precisión en torno al 90 % [32].

Implementación de Foersch et al.

Foersch *et al.* en el artículo «*Deep learning for diagnosis and survival prediction in soft tissue sarcoma*» [33] proponen un modelo para el diagnóstico histológico y la predicción de supervivencia en sarcomas de partes blandas (STS), utilizando imágenes digitales de portaobjetos escaneados (*Whole Slide Images*, WSI) teñidas con hematoxilina-eosina (H&E). Su implementación se basa en redes neuronales convolucionales entrenadas con mosaicos (*tiles*) de alta resolución extraídos de estas WSI.

Dataset y preprocesamiento

El modelo fue diseñado para clasificar los cinco subtipos más frecuentes de STS: leiomiosarcoma (LMS), liposarcoma desdiferenciado (DDLPS), sarcoma sinovial (SS), sarcoma pleomórfico indiferenciado (UPS) y mixofibrosarcoma (MFS). Para ello se utilizaron dos cohortes independientes: una de entrenamiento y validación (240 pacientes, 456 WSI) procedente del consorcio TCGA (*The Cancer Genome Atlas Program*), y una cohorte externa de test (51 pacientes) proveniente de centros de referencia alemanes.

Arquitectura y entrenamiento
La arquitectura utilizada es *DenseNet121*, entrenada con la librería `FastAI` sobre `PyTorch`. Se utilizaron 75 *tiles* por paciente, seleccionados aleatoriamente. Cada *tile* tenía un tamaño de 1024×1024 píxeles, lo que permitía conservar tanto contexto tisular como detalles celulares relevantes. Las regiones tumorales fueron anotadas manualmente por expertos, y se aplicó normalización de color según Vahadane *et al.* para reducir la variabilidad de tinción entre centros. El modelo se entrenó con función de pérdida de entropía cruzada, optimizador Adam con *momentum* de 0,95, *batch size* 32 y un *learning rate* seleccionado automáticamente mediante `lr_find()`, con un valor final de 1×10^{-4}.

El diagnóstico a nivel de WSI se obtuvo mediante una votación ponderada entre *tiles*, teniendo en cuenta la clase predicha y la probabilidad asociada a cada uno. Para la interpretación visual, se generaron mapas de clasificación sobre la imagen completa mediante una ventana deslizante, además de aplicar *Class Activation Maps* (CAMs) para resaltar regiones relevantes.

Evaluación y resultados
La evaluación se realizó mediante validación cruzada estratificada (90/10), repetida seis veces con diferentes particiones. El modelo alcanzó una exactitud media del 79,9 % ± 6,1 % y un AUROC macro-promediado de 0,97 ± 0,01, con valores por clase entre 0,91 (DDLPS) y 1,00 (SS). En el test externo, el rendimiento se mantuvo (exactitud del 78,4 %, AUROC de 0,94).

Además, se llevó a cabo un estudio con nueve patólogos, en el que se demostró que el uso del modelo como herramienta de apoyo mejoraba significativamente la precisión diagnóstica (de 46,3 % a 87,1 %) y reducía el tiempo de análisis por caso (de 89,3 s a 43,2 s).

Predicción de supervivencia
Por otro lado, también se entrenó un segundo modelo para predecir la supervivencia específica a dos años en pacientes con leiomiosarcoma, a partir de 139 WSI de 85 casos. En este caso se utilizaron hasta 300 *tiles* por paciente. El modelo alcanzó una exactitud del 88,9 % ± 9,9 % y un AUROC de 0,91 ± 0,098, superando a predictores clínicos tradicionales como el grado FNCLCC o el estado de resección quirúrgica.

Por último, se utilizaron técnicas de reducción de dimensionalidad (*t*-SNE) y visualización por activación para identificar patrones histológicos asociados al pronóstico. Entre los hallazgos más relevantes se encontraron núcleos pequeños y monomorfos en casos de buen pronóstico, y núcleos grandes, pleomórficos y con hemorragia intratumoral en pacientes con peor evolución.

Capítulo 3

Desarrollo

En este capítulo se presenta el desarrollo de la implementación propuesta. En primer lugar, se describe de forma general el enfoque empleado, explicando la implementación de referencia utilizada como punto de partida. A continuación, se detalla el proceso de generación del dataset, incluyendo la selección y preprocesamiento de datos y la generación de *tiles*. Posteriormente, se desarrollan cinco experimentos realizados, explicando las distintas estrategias de entrenamiento y calibración aplicadas, así como el análisis de los resultados obtenidos. Finalmente, se describe la aplicación implementada para la inferencia.

3.1. Descripción general de la implementación

La implementación propuesta consiste en un modelo de aprendizaje profundo para la clasificación histopatológica de sarcomas de partes blandas a partir de imágenes histológicas digitalizadas. Esta implementación está inspirada en el trabajo de Foersch *et. al*, «*Deep learning for diagnosis and survival prediction in soft tissue sarcoma*» [33], comentado previamente. Sin embargo, no se parte de la implementación propuesta en el artículo, sino de una versión simplificada en formato `Jupyter Notebook`, publicada por los autores en el repositorio de código `GitHub` bajo el nombre «*DeepLearningSarcoma*» [34].

Esta versión simplificada —de ahora en adelante, implementación de referencia— no incluye la predicción de pronóstico, sino que solo realiza la predicción de los cinco subtipos histológicos más frecuentes a nivel de *tile*. Por otra parte, tampoco implementa muchas de las técnicas mencionadas en el artículo, como validación cruzada o CAMs.

3.1.1. Implementación de referencia

Además del `Jupyter Notebook`, se proporcionan las dependencias necesarias (`requirements`), muchas de las cuales se encuentran actualmente obsoletas —como se detalla más adelante—:

- python >= 3.6
- fastai 1.0.57 (v1)
- matplotlib 3.1.3
- pandas 1.1.5
- torch 1.4.0
- torchvision 0.5.0
- Pillow 7.0.0
- sklearn 0.23.1

El modelo está implementado en `PyTorch` y `FastAI`. `PyTorch`, una biblioteca de aprendizaje profundo de bajo nivel, es el *framework* base que permite construir y entrenar redes neuronales. Por otra parte, `FastAI` es una biblioteca de alto nivel construida sobre `PyTorch` que simplifica muchas de las tareas realizadas en aprendizaje profundo, como la carga de datos o el entrenamiento de modelos.

Además de `PyTorch` y `FastAI`, se utilizan otras bibliotecas fundamentales, como `scikit-learn` (`sklearn`) para tareas como el cálculo de métricas de evaluación o el cálculo de pesos de clase. También se utiliza `pandas` para el manejo de estructuras de datos, `matplotlib` para generar representaciones visuales y `Pillow` para cargar y procesar imágenes.

Dataset

Como entrada, el modelo recibe *tiles* en formato JPG, extraídos previamente de imágenes histológicas WSI del dataset TCGA-SARC. Cada *tile* (1024×1024 px) está etiquetado con el subtipo histológico —clase— correspondiente (LMS, DDLPS, UPS), y se asocia al paciente del que fue extraído. El dataset usado en esta implementación es limitado, ya que incluye solo 50 imágenes para el entrenamiento y 25 para validación.

La asignación de etiquetas se realiza mediante un archivo en formato *.csv*, que incluye las siguientes columnas: `patient_id` (identificador del paciente), `Path` (ruta de la imagen), `Label` (subtipo histológico), `training_set` (valor *TRAIN* o *VALID* según su uso) e `is_valid` (valor *True* o *False* según pertenezca al conjunto de validación o no).

Durante el entrenamiento, se aplica la técnica de aumento de datos (*data augmentation*), que consiste en aplicar transformaciones aleatorias a los *tiles* con el objetivo de aumentar la diversidad del conjunto de entrenamiento y mejorar la capacidad de generalización del modelo. Se emplean volteos verticales, rotaciones de hasta 90° y modificaciones en iluminación, contraste y brillo para simular la variabilidad introducida por distintos escáneres histológicos.

Además, se utiliza la técnica *cutout*, que consiste en eliminar pequeños parches cuadrados de la imagen para evitar que el modelo dependa excesivamente de zonas muy concretas y favorecer así un aprendizaje más robusto y generalizable.

Entrenamiento

El modelo se entrena utilizando la arquitectura `DenseNet121`, una red convolucional profunda preentrenada en ImageNet, una base de datos de referencia con millones de imágenes de categorías muy diversas, de forma que el modelo ha aprendido a detectar características visuales generales como bordes, texturas y formas que pueden transferirse a tareas de predicción más específicas. Este enfoque, conocido como aprendizaje por transferencia (*transfer learning*), permite empezar el entrenamiento a partir de unas bases generales ya aprendidas, lo que facilita que el modelo aprenda más rápido y con menos datos, algo especialmente útil cuando el conjunto de entrenamiento es limitado.

En la arquitectura `DenseNet121` cada capa recibe como entrada no solo la salida de la capa anterior, sino también las salidas de todas las capas previas. Esto es lo que se conoce como conectividad «densa», y permite que cada capa tenga acceso directo a los patrones aprendidos en capas anteriores. Esto favorece la reutilización de características y la reducción del número de parámetros.

Para la carga de datos se utiliza un `batch_size=5` y 16 *workers* (procesos paralelos encargados de la carga de datos) para acelerar la lectura de imágenes durante el entrenamiento. Además, en el repositorio se menciona que se usó una GPU Titan RTX durante el entrenamiento.

Se entrena utilizando como función de error la función de pérdida de entropía cruzada (`CrossEntropyLoss`) y se monitorizan dos métricas principales: exactitud (*accuracy*) y tasa de error (*error_rate*). En el Capítulo 2 se habló del error cuadrático como función de pérdida habitual en problemas de regresión, pero en tareas de clasificación no es apropiado porque no tiene en cuenta la confianza —es decir, la certeza en términos de probabilidad— con la que el modelo hace las predicciones. Por el contrario, `CrossEntropyLoss` penaliza especialmente los errores cometidos cuando el modelo está muy seguro de la decisión, lo que la hace especialmente adecuada en contextos donde es importante evitar predicciones erróneas con alta confianza, como en el ámbito médico. En concreto, se utiliza una versión ponderada de la entropía cruzada, en la que se asigna un peso mayor a las clases minoritarias, con el objetivo de evitar que el modelo se sesgue hacia la clase mayoritaria. Esto es especialmente útil cuando el dataset está desbalanceado, es decir, cuando las clases no están representadas en proporciones similares.

Por otra parte, se emplea el optimizador `Adam`, con un parámetro de regularización `weight_decay` de 0,1, que penaliza los pesos excesivamente grandes para evitar que el modelo dependa demasiado de entradas concretas y reducir así el riesgo de sobreajuste. Como técnica adicional de regularización se emplea `MixUpCallback`, que combina imágenes y etiquetas durante el entrenamiento para mejorar la generalización del modelo. Además de este, se utilizan los siguientes *callbacks* —funciones auxiliares que se activan automáticamente en momentos clave del entrenamiento—: `CSVLogger` para registrar el historial de métricas por época de entrenamiento, y `SaveModelCallback`, que guarda automáticamente el modelo cuando se alcanza la mejor métrica de validación (en este caso, *accuracy*).

El modelo se construye siguiendo el enfoque habitual en el aprendizaje por transferencia, donde se reutiliza una red convolucional profunda preentrenada para extraer características generales —conocido como *backbone*—, en este caso `DenseNet121`, y se le añade una nueva capa de clasificación —*head*— adaptada a la tarea específica. En este caso, consiste en una serie de capas lineales diseñadas para realizar la predicción del subtipo histológico de sarcoma de partes blandas. Es decir, el *backbone* de la red se encarga de extraer patrones visuales generales y el *head* es el que aprende a asociar estas características con las clases específicas a predecir.

El entrenamiento se divide en dos fases:

1. **Entrenamiento del *head* con *backbone* congelado** (`RunPart 1`). En esta primera fase, las capas profundas del *backbone* preentrenado permanecen congeladas, de forma que solo se actualizan los pesos del *head*. Esto permite entrenar las capas más especializadas sin modificar el conocimiento general ya aprendido por la red preentrenada. Esta fase se realiza durante 25 épocas, con una tasa de aprendizaje de 1×10^{-4}, seleccionada mediante el método `lr_find()`. Concretamente, se calcula tres veces y se utiliza el promedio para compensar la variabilidad debida a la iniciación aleatoria de los parámetros. Cabe destacar el uso concreto de la función `learn.fit_one_cycle()` de FastAI, que a diferencia de `fit()`, permite ajustar dinámicamente la tasa de aprendizaje durante cada `epoch`.
2. **Entrenamiento de la red completa** (`RunPart 2`). En esta fase, se carga el mejor modelo guardado anteriormente y se descongelan también las capas del *backbone* para permitir el *fine-tuning* de toda la red, que consiste en entrenar también el *backbone* para el caso específico de clasificación de subtipos histológicos. Para evitar que las capas preentrenadas se desajusten bruscamente y «olviden» lo aprendido, se utilizan tasas de aprendizaje diferentes: una más baja para el *backbone* y otra mayor para el *head* —usando `slice` en `learn.fit_one_cycle()`—, con el objetivo de que las capas profundas aprendan más despacio en esta fase para no perder el conocimiento general. Concretamente, se usa el anterior `learning_rate` calculado para el `head` y se calcula uno nuevo tras descongelar el *backbone* para las capas preentrenadas. Se mantiene el mismo número de `epochs`.

En este caso, se observa que el modelo alcanza una mejor precisión tras la primera fase de entrenamiento, llegando a un valor máximo de *accuracy* del 60 %. Tras la segunda fase, la precisión baja ligeramente, lo que puede deberse a un sobreajuste durante el *fine-tuning* por haber usado una tasa de aprendizaje demasiado baja o por las características propias del dataset, que en este caso es muy reducido.

Evaluación

Una vez finalizado el entrenamiento, se carga el modelo que obtuvo la mejor precisión en el conjunto de validación para realizar la evaluación final. Idealmente, el modelo debería evaluarse utilizando un conjunto de datos externo (*test set*); sin embargo, en esta implementación de referencia se utilizan las imágenes del conjunto de validación a modo de demostración.

En primer lugar, se obtienen las predicciones del modelo de todos los *tiles* en forma de distribuciones de probabilidad para cada clase, obtenidas a partir de las salidas del modelo (logits) y transformadas mediante la función de activación *softmax*. A partir de estas probabilidades, se calcula la predicción final para cada *tile* seleccionando la clase con mayor probabilidad. Como métricas de evaluación se utilizan la exactitud (*accuracy*), la matriz de confusión y la visualización de las muestras con mayor pérdida. Esto es especialmente útil para analizar los errores del modelo e identificar posibles patrones sistemáticos de error.

Es importante destacar que, en esta implementación, cada paciente está representado por un único *tile*, por lo que las predicciones a nivel de *tile* coinciden con las predicciones a nivel de paciente. No obstante, en un entorno real, lo habitual es tener más de un *tile* por paciente.

A partir de la versión de referencia, se han realizado diversas adaptaciones a lo largo de sucesivos experimentos hasta alcanzar una solución final. En las próximas secciones se desarrollan las distintas fases del proyecto, incluyendo la obtención y preprocesamiento de los datos y los distintos experimentos realizados.

3.2. Generación del dataset

3.2.1. Obtención de datos

Aunque en el repositorio de la implementación de referencia se proporciona una carpeta con 75 *tiles* en total —50 para entrenamiento y 25 para validación— y un archivo `.csv` con las etiquetas correspondientes, este dataset resulta insuficiente para entrenar un modelo robusto y obtener resultados comparables.

Por ello, se decidió obtener las imágenes histológicas digitalizadas —WSI— directamente del dataset TCGA-SARC, parte de la iniciativa TCGA del *National Cancer Institute* (NCI), cuyo objetivo es caracterizar diversos tipos de cáncer a nivel genómico, clínico y patológico [35]. En concreto, TCGA-SARC recopila datos de pacientes con sarcomas de partes blandas y está disponible a través del portal *Genomic Data Commons* (GDC) [36].

Estos datos, y en concreto las WSI, proceden de múltiples centros hospitalarios, lo que introduce una variabilidad significativa en aspectos como la tinción, el escaneado y la calidad de imagen. Esta heterogeneidad es deseable, ya que permite generar un dataset más representativo, favoreciendo así la generalización del modelo. Cada WSI tiene un identificador único de paciente, lo que permite relacionar cada imagen con sus correspondientes datos clínicos y moleculares, disponibles en la base de datos genómica cBioPortal [37]. En el contexto de este trabajo, esto no solo resulta útil para consultar los detalles de diagnóstico, sino también para la futura incorporación de nuevas funcionalidades, como la predicción de pronóstico o respuesta a tratamientos.

Selección de datos

La calidad de los datos condiciona directamente la capacidad de aprendizaje y generalización del modelo. Aunque lo ideal sería contar con la colaboración de patólogos especializados, en esta primera versión del trabajo se han tenido que tomar decisiones en base a otros criterios.

Se decidió trabajar con tres de los subtipos histológicos más frecuentes: leiomiosarcoma (LMS), liposarcoma desdiferenciado (DDLPS) y sarcoma pleomórfico indiferenciado (UPS), en lugar de los cinco subtipos contemplados en la implementación de referencia. Esta decisión responde a dos motivos principales: (1) los dos subtipos descartados —sarcoma sinovial y mixofibrosarcoma— estaban infrarrepresentados en la base de datos, lo que impedía mantener un número equitativo de casos por clase, y (2) reducir el número de clases facilitaba el análisis en esta primera versión del modelo.

Durante el análisis de los datos disponibles en los portales GDC y cBioPortal, se observó que algunas WSI no contaban con información clínica y/o molecular asociada. Estos casos se descartaron porque se consideró importante disponer de los informes patológicos para validar el diagnóstico, además de por la intención futura de aumentar las capacidades del modelo para predicción de pronóstico y respuesta al tratamiento.

También se identificaron pacientes con varias WSI asociadas. Al revisar las imágenes y los informes patológicos, se comprobó que en muchas ocasiones se correspondían a casos especialmente complejos a nivel clínico e histológico, en muchos casos con múltiples biopsias. Dada la dificultad para determinar la relevancia diagnóstica de cada una, finalmente se decidió descartar estos casos inicialmente para garantizar la homogeneidad del conjunto de datos.

A partir de estos criterios, se establecieron dos reglas principales:

1. Incluir únicamente WSI vinculadas en cBioPortal a datos clínicos y moleculares del paciente.
2. Incluir solo un WSI por paciente, descartando aquellos con múltiples WSI para reducir la complejidad del análisis.

Tras aplicar estos criterios, especialmente debido al segundo, muchas WSI quedaron descartadas. Algunas de ellas se reservaron para el conjunto de validación externa o *test* —datos no vistos durante el entrenamiento para evaluar la generalización del modelo—.

3.2.2. Preprocesamiento de los datos

Las WSI descargadas están en formato `.svs`, un formato TIFF propietario desarrollado por Leica Biosystems (Aperio). Este formato contiene una estructura piramidal de la imagen que contiene varias resoluciones de la misma imagen, lo que permite visualizarla rápidamente a niveles diferentes de *zoom* sin cargar la resolución completa. Esto es muy útil para la visualización, pero en el contexto concreto de proyectos de aprendizaje profundo, además de ser archivos muy grandes —pueden superar varios GB, ya que contienen la lámina del portaobjetos entera a resolución del microscopio—, no todos los *frameworks* admiten este formato.

Por ello, es muy común la generación de *tiles* a partir de las WSI, que consiste en extraer subimágenes (*tiles*) en formato JPG o PNG del archivo `.svs`. Este proceso puede hacerse basándose en regiones de interés (ROI), de forma que se seleccionan las zonas más relevantes a nivel diagnóstico (zonas con mayor heterogeneidad, necrosis, densidad celular...). Aunque esto es muy interesante, requiere la colaboración de un patólogo, por lo que en esta implementación se ha optado por la generación de *tiles* de la imagen completa.

Selección del nivel de resolución

Como se ha comentado previamente, cada imagen `.svs` contiene múltiples niveles jerárquicos de resolución. Con el objetivo de crear un dataset lo más homogéneo posible, se examinó la resolución de todos los niveles de cada imagen usando la biblioteca `OpenSlide` para elegir el nivel más adecuado en cada imagen. Se observó una reducción de resolución por un factor de 4 en cada nivel: el nivel 0 contiene la imagen completa con resolución máxima, mientras que los niveles superiores (1, 2 y 3) representan versiones reducidas por factores de 4, 16 y 64 respectivamente.

La resolución del nivel 0 oscilaba generalmente entre 70.000 y 120.000 píxeles por lado. Usar esta resolución para el entrenamiento del modelo implicaría un coste computacional excesivo. Por otra parte, los niveles 2 y 3 perdían excesivo detalle, por lo que se optó por el nivel 1 como equilibrio entre resolución diagnóstica y coste computacional, con resoluciones típicas entre 14.000 y 36.000 píxeles por lado.

No obstante, hubo excepciones en las que el nivel de resolución original era excesivamente grande (superior a 150.000 píxeles por lado), y otras en las que era demasiado pequeño (inferior a 50.000 píxeles por lado). En el caso de las resoluciones excesivamente grandes, fueron descartadas para evitar el excesivo coste computacional y mantener la homogeneidad del dataset. En los casos de resolución insuficiente, si la resolución de nivel 0 era adecuada —entre 14.000 y 36.000 píxeles por lado—, se hacía la excepción de utilizar este nivel; si no, se descartaba.

Tras este descarte, se descargaron y analizaron nuevos datos para completar de nuevo el dataset. En total, se descargaron un total de 33 WSI —cada una correspondiente a un paciente diferente—, 11 de cada subtipo histológico (LMS, DDLPS o UPS). Aunque se podrían haber descargado más WSI de LMS y DDLPS, no había más casos disponibles de UPS, por lo que se fijó en 11 muestras por subtipo para asegurar el equilibrio de clases.

Generación de *tiles*

Al igual que en la implementación de referencia, se optó por dividir estas imágenes en *tiles* de 1024×1024 píxeles, con el objetivo principal de conservar contexto tisular y detalles importantes, así como evitar la generación excesiva de *tiles*.

Este proceso se llevó a cabo utilizando las bibliotecas `OpenSlide` y `Pillow`. Al hacer las primeras pruebas, se observó que un gran número de *tiles* no tenía tejido por contener áreas vacías del portaobjetos. Esto podía suponer un problema al introducir ruido durante el entrenamiento. Para solventarlo, se implementó una función de filtrado de *tiles* con un porcentaje de contenido blanco superior al 80 %. Se optó por usar este porcentaje porque, en las pruebas realizadas, porcentajes más agresivos descartaban imágenes con poca densidad de tejido tumoral, pero que podían ser importantes a nivel diagnóstico. Se destaca de nuevo la importancia de conformar un dataset junto a patólogos expertos en sarcomas.

Por último, durante la revisión manual se descartaron los *tiles* de mala calidad por contener partes borrosas u otros artefactos.

Como resultado, se obtiene un dataset formado por 33 carpetas —nombradas con el identificador del paciente—, cada una con los *tiles* del paciente correspondiente, nombrados con el identificador del paciente y las coordenadas del WSI correspondientes.

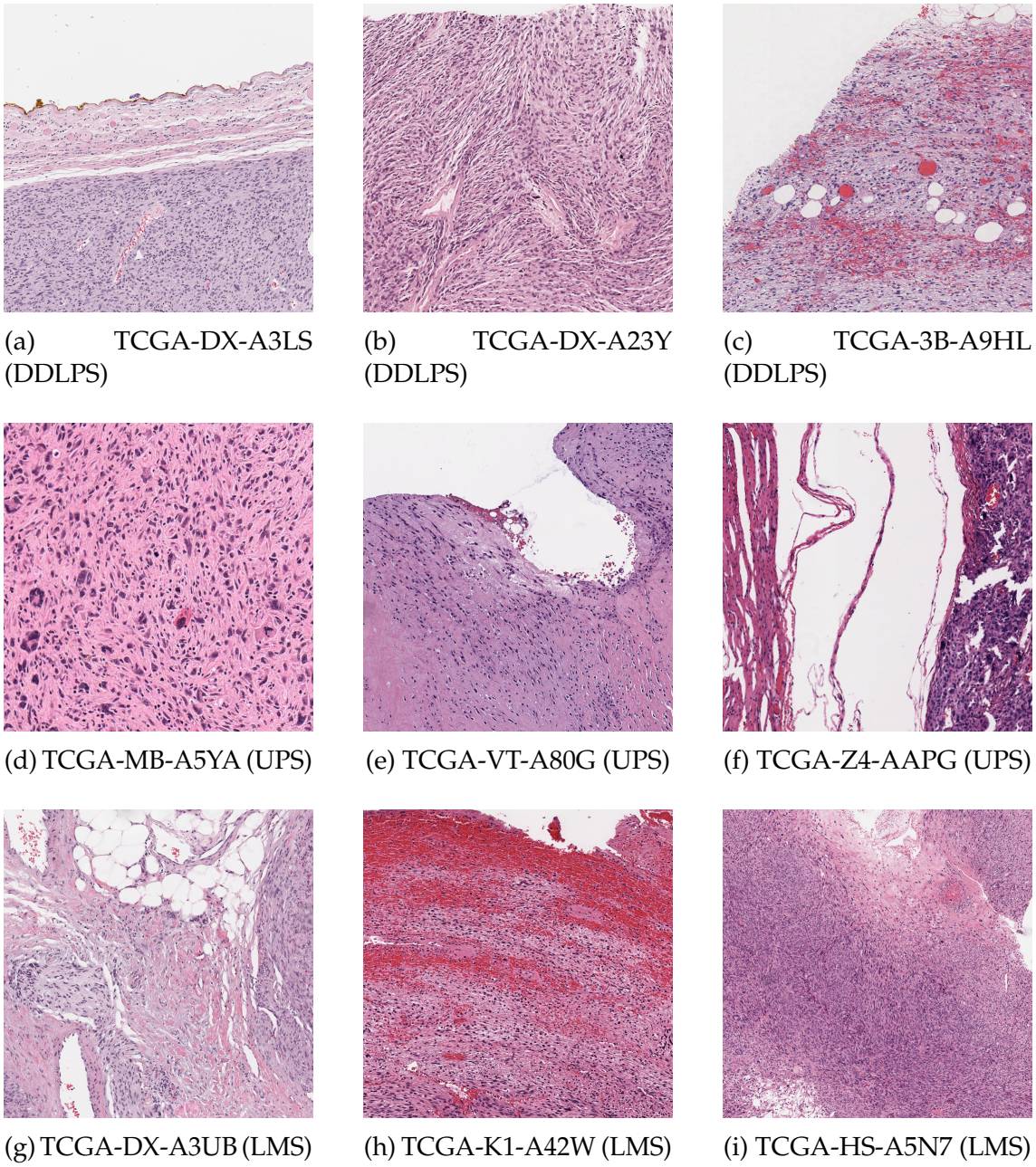
En total se generaron 8058 *tiles* distribuidos de la siguiente forma: 3492 para LMS, 1960 para DDLPS y 2606 para UPS. Como se observa, aunque se mantuvo el mismo número de pacientes por caso, el número de *tiles* generados varió notablemente entre clases. Esta diferencia no puede explicarse únicamente por la resolución promedio de las imágenes (LMS: 8539 píxeles, DDLPS: 7775 píxeles, UPS: 6854 píxeles). Se atribuye principalmente a diferencias en la proporción de tejido en cada lámina histológica, condicionadas por factores histológicos, anatómicos y técnicos.

La Tabla 3.1 muestra la distribución exacta de *tiles* por paciente.

Paciente	**Subtipo histológico**	**Nº de tiles**
TCGA-3B-A9HO	DDLPS	148
TCGA-3B-A9HL	DDLPS	146
TCGA-DX-A23Y	DDLPS	245
TCGA-DX-A240	DDLPS	91
TCGA-DX-A1KW	DDLPS	93
TCGA-DX-A2J1	DDLPS	176
TCGA-DX-A23T	DDLPS	311
TCGA-DX-A3LS	DDLPS	331
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A48N	DDLPS	165
TCGA-DX-A3U5	DDLPS	47
TCGA-MB-A5YA	UPS	207
TCGA-VT-AB3D	UPS	186
TCGA-Z4-AAPG	UPS	78
TCGA-IE-A4EJ	UPS	111
TCGA-QC-A6FX	UPS	56
TCGA-DX-A3U8	UPS	424
TCGA-DX-A8BZ	UPS	533
TCGA-SG-A6Z4	UPS	161
TCGA-MB-A5Y9	UPS	314
TCGA-VT-A80G	UPS	72
TCGA-DX-A1KX	UPS	464
TCGA-DX-A3UC	LMS	471
TCGA-DX-A3U7	LMS	438
TCGA-HS-A5N7	LMS	123
TCGA-K1-A3PO	LMS	50
TCGA-3B-A9HX	LMS	252
TCGA-3B-A9I3	LMS	259
TCGA-K1-A42W	LMS	573
TCGA-DX-A3UF	LMS	298
TCGA-DX-A48J	LMS	189
TCGA-DX-A48R	LMS	486
TCGA-DX-A3UB	LMS	353

Tabla 3.1: Resumen del número de *tiles* por paciente

En la Figura 22 se muestran algunos ejemplos de *tiles* de distintos pacientes, seleccionados con fines ilustrativos. Se puede observar la heterogeneidad del dataset.



(a) TCGA-DX-A3LS (DDLPS) (b) TCGA-DX-A23Y (DDLPS) (c) TCGA-3B-A9HL (DDLPS)

(d) TCGA-MB-A5YA (UPS) (e) TCGA-VT-A80G (UPS) (f) TCGA-Z4-AAPG (UPS)

(g) TCGA-DX-A3UB (LMS) (h) TCGA-K1-A42W (LMS) (i) TCGA-HS-A5N7 (LMS)

Figura 22: Ejemplos del dataset por subtipo histológico

Posteriormente, se generó un archivo `.csv` con las etiquetas correspondientes al subtipo tumoral y la partición asignada —entrenamiento o validación—. En todos los experimentos se siguió un enfoque de división estratificada por paciente, lo que garantiza que todos los *tiles* de un mismo paciente se asignen exclusivamente a una de las particiones, evitando así cualquier solapamiento entre entrenamiento y validación. De lo contrario, el modelo sería evaluado con los datos del propio entrenamiento, impidiendo evaluar su capacidad real de generalización a nuevos casos.

Este proceso de etiquetado se detalla en las siguientes secciones, ya que, si bien en todos los experimentos se respetó la estratificación por clase y la agrupación por paciente, el tipo de partición varió según el experimento: en algunos casos se empleó una única división fija (*split* 90/10), y en otros, validación cruzada.

Por último, tras formar el dataset para el entrenamiento, se procedió a descargar las últimas WSI para conformar un conjunto de evaluación externa o *test*. Concretamente, se definieron dos conjuntos, como se muestra en las Tabla 3.2 y Tabla 3.3. Dado que no quedaban más casos disponibles del subtipo UPS, se usaron los casos inicialmente descartados, bien por tener varios WSI asociados (`TCGA-VT-A80J`), o bien por tener una resolución demasiado baja (`TCGA-MB-A5Y8`). Aunque el paciente `TCGA-VT-A80J` tiene varios WSI asociados (DX1 y DX2), uno no pudo usarse por no generarse ningún *tile* útil. Además, se usó un caso de LMS con múltiples WSI asociados, con el objetivo de observar cómo se comporta el modelo ante escenarios más complejos y heterogéneos, donde un mismo paciente presenta muestras histológicas distintas.

Paciente	**Subtipo**	**Nº tiles**
TCGA-DX-A2J0	DDLPS	31
TCGA-QQ-A5VB	LMS	289
TCGA-FX-A3NJ	LMS	64
TCGA-VT-A80J (DX2)	UPS	154
TCGA-QQ-A5V9	UPS	233

Tabla 3.2: Conjunto de evaluación externa TestSet-1

Paciente	**Subtipo**	**Nº tiles**
TCGA-DX-A2IZ	DDLPS	215
TCGA-DX-A6BA (DX1)	LMS	417
TCGA-DX-A6BA (DX2)	LMS	145
TCGA-DX-A48U	LMS	61
TCGA-MB-A5Y8	UPS	3

Tabla 3.3: Conjunto de evaluación externa TestSet-2

3.3. Primer experimento: migración de versiones

Como punto de partida, se optó por replicar la implementación de referencia, realizando únicamente las modificaciones necesarias para su ejecución. El objetivo de este primer experimento era evaluar el comportamiento del modelo original usando el dataset elaborado.

3.3.1. Conjunto de entrenamiento y validación

En este primer experimento se realiza una única partición del dataset en entrenamiento y validación, replicando la estrategia original. La división se realiza de forma estratificada por paciente, garantizando que todos los *tiles* de un mismo caso se encuentren exclusivamente en uno de los dos conjuntos. Esto es fundamental para evitar *data leakage*, ya que la presencia de datos compartidos entre entrenamiento y validación introduciría sesgos y comprometería la capacidad del modelo para generalizar a nuevos casos, como se ha comentado previamente.

Se decide utilizar un 90 % de los casos para entrenamiento y un 10 % para validación, como se muestra en las Tabla 3.4 y Tabla 3.5. Se elige esta proporción para aprovechar al máximo los datos disponibles para el aprendizaje del modelo, dado el número limitado de casos, además de considerarse suficiente para obtener una estimación inicial del rendimiento del modelo.

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HL	DDLPS	146
TCGA-3B-A9HO	DDLPS	148
TCGA-VT-AB3D	UPS	186
TCGA-Z4-AAPG	UPS	78
TCGA-HS-A5N7	LMS	123
TCGA-DX-A3UC	LMS	471
TCGA-DX-A3U7	LMS	438
TCGA-K1-A3PO	LMS	50
TCGA-DX-A23Y	DDLPS	245
TCGA-IE-A4EJ	UPS	111
TCGA-3B-A9HX	LMS	252
TCGA-3B-A9I3	LMS	259
TCGA-DX-A240	DDLPS	91
TCGA-DX-A1KW	DDLPS	93
TCGA-DX-A2J1	DDLPS	176
TCGA-DX-A3U8	UPS	424
TCGA-K1-A42W	LMS	573
TCGA-DX-A3LS	DDLPS	331
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A23T	DDLPS	311
TCGA-DX-A48N	DDLPS	165
TCGA-DX-A3UF	LMS	298
TCGA-DX-A8BZ	UPS	533
TCGA-SG-A6Z4	UPS	161
TCGA-DX-A48R	LMS	486
TCGA-VT-A80G	UPS	72
TCGA-DX-A3UB	LMS	353
TCGA-MB-A5Y9	UPS	314
TCGA-DX-A1KX	UPS	464

Tabla 3.4: Experimento 1: subconjunto de entrenamiento

Paciente	Subtipo	Nº tiles
TCGA-DX-A3U5	DDLPS	47
TCGA-DX-A48J	LMS	189
TCGA-MB-A5YA	UPS	207
TCGA-QC-A6FX	UPS	56

Tabla 3.5: Experimento 1: subconjunto de validación

3.3.2. Migración de versiones

En la implementación de referencia se usaron versiones actualmente obsoletas, siendo especialmente críticas las de `FastAI`, `PyTorch`, `NumPy` y `Python`. Aunque en un entorno local se podrían instalar estas versiones antiguas, Google Colab utiliza versiones modernas de las bibliotecas y no permite revertir a versiones anteriores. Inicialmente se intentó mantener el uso de `FastAI v1` usando las versiones `fastai=1.0.61`, `pytorch=2.0.1` y `python=3.11` y `numpy=1.24.4`, pero finalmente se observó que `FastAI v1` no es compatible con versiones actuales de `PyTorch`, `NumPy` y `Python`. Por ejemplo, algunas funciones de `FastAI v1` como `.apply_tfms()` o `transform()`, utilizadas para aplicar las transformaciones de *data augmentation*, dependen internamente de funciones como `torch.solve()`, eliminadas en `Pytorch v2`, generando errores durante la ejecución. Por otra parte, también se observaron errores al ejecutar funciones de módulos que habían sido compilados con versiones antiguas de NumPy (v1.x).

En vista de que el proyecto estaba obsoleto, se decidió migrar a las versiones más recientes de todas las bibliotecas. La migración a la versión 2 de `FastAI` (2.7.19) supuso cambios en el código de la implementación de referencia en aspectos como la carga de datos, la aplicación de transformaciones, la creación del *learner* y el uso de *callbacks*. Cabe mencionar que se mantuvieron las transformaciones originales en la medida de lo posible, adaptando aquellas obsoletas o incompatibles con la sintaxis o el funcionamiento de la versión 2 de `FastAI`.

3.3.3. Otras adaptaciones

Tras adaptar el código a la migración de versiones, se tomaron otras decisiones críticas. Por ejemplo, aunque en la implementación de referencia se usa un `batch_size` de 5 imágenes y 16 procesos paralelos, esto no pudo replicarse debido a las limitaciones de las GPUs ofrecidas por Google Colab. Inicialmente se utilizó la GPU NVIDIA Tesla T4, de uso gratuito, probando distintos `batch_size`. Con valores superiores a 5 imágenes y más de 4 procesos paralelos, se producía desbordamiento de la memoria (error *out of memory*) o se producía un cuello de botella en la carga de imágenes. Este problema se agrava por la carga de datos desde Google Drive, que es inherentemente lenta, y el uso de imágenes con una alta resolución (1024x1024 px).

Estas limitaciones de la GPU llevaron a tomar la decisión de adquirir el plan de pago Google Colab Pro, que permitía usar, entre otras, la GPU NVIDIA L4, con 24 GB de memoria frente a los 16 de la Tesla T4. Tras varias pruebas, la máxima configuración posible fue con `batch_size` de 6 y 8 procesos paralelos. Cabe destacar que idealmente se debería usar un `batch_size` similar al de la implementación de referencia —en torno a 16-64 imágenes—, no solo para una mayor velocidad en el entrenamiento, sino también para contribuir a un entrenamiento más estable. No obstante, el optimizador empleado, Adam, tolera mejor los tamaños de lote pequeños que otros optimizadores.

Por otra parte, se decidió usar semilla aleatoria (`seed`) para facilitar la reproducibilidad de los experimentos. Esta técnica permite controlar la aleatoriedad en procesos como la inicialización de pesos en capas no preentrenadas (*head*), el orden de los datos o las transformaciones aleatorias aplicadas durante el entrenamiento, facilitando la reproducibilidad de los experimentos —aunque no de forma exacta, ya que no fija la aleatoriedad de todos los procesos—. Aunque limitar la aleatoriedad puede restringir la exploración de ciertas variaciones, resulta especialmente útil en fases iniciales para poder comparar los resultados del modelo al realizar adaptaciones.

En cuanto al entrenamiento, en lugar de utilizar 25 `epochs` para las dos fases como en la implementación de referencia, se emplearon 5 épocas para el entrenamiento del `head` y 10 para el `fine-tuning` de toda la red. Aunque usar más `epochs` hubiera sido lo ideal, el tiempo de entrenamiento podía superar las 13 horas, lo que hacía inviable supervisar el proceso de forma continua y agotaba rápidamente el tiempo disponible de GPU en `Google Colab`.

Además, se añadió el *callback* `EarlyStopping` con un valor `patience=3`, que permite interrumpir el entrenamiento automáticamente cuando no se produce una mejora en la métrica monitorizada —en este caso, `accuracy`— durante un número consecutivo de épocas igual al valor de `patience`.

Por último, en la implementación de referencia se utiliza un único *tile* por paciente, por lo que las predicciones a nivel de *tile* y de `paciente` coinciden. Sin embargo, en el dataset que se ha generado para este trabajo, cada paciente tiene decenas o cientos de *tiles*, lo que hace necesario calcular métricas adicionales de validación por paciente, ya que el objetivo es predecir el subtipo histológico del paciente, no de cada *tile* individual. Para ello, se aplica votación mayoritaria por *tile* (*hard voting*) para cada paciente, asignando como categoría final aquella más repetida entre sus *tiles*.

Además, se añade el cálculo de la desviación estándar de las probabilidades por clase entre los *tiles* de cada paciente para cuantificar la variabilidad en las predicciones intra-paciente. Un valor bajo sugiere que los tiles son coherentes entre sí y que el modelo tiene una alta confianza en la predicción final, mientras que una desviación alta puede indicar heterogeneidad morfológica, incertidumbre del modelo o presencia de tiles poco representativos. Esto es especialmente útil para el caso de uso de esta implementación, ya que los sarcomas típicamente presentan heterogeneidad en un mismo paciente.

3.3.4. Resultados del entrenamiento

Al terminar el entrenamiento, se analizó el archivo generado por `CSVLogger` correspondiente a la segunda fase (`RunPart 2`).

Como se observa en la Tabla 3.6, la primera época presentó un rendimiento sospechosamente bueno en validación, con una `accuracy` de 0,99 y `valid_loss` de 0,07. Este elevado valor de exactitud, junto con la baja pérdida de validación podría indicar que el conjunto de validación —formado por solo cuatro pacientes— es poco representativo o más sencillo de clasificar.

En las siguientes épocas se observa un empeoramiento significativo del rendimiento en validación. En la época 1, la `valid_loss` aumenta más del doble, y la `accuracy` cae ligeramente a 0,95. En la época 2, la `valid_loss` alcanza su máximo (0,59) y la `accuracy` cae a 0,68. Esta evolución apunta a un sobreajuste (*overfitting*), en el que el modelo se ajusta demasiado a los datos de entrenamiento y pierde capacidad de generalización.

En la época 3, se observa una ligera recuperación del rendimiento con una `accuracy` de 0,79 y una `valid_loss` de 0,41, aunque sin alcanzar los resultados de la época 0. Como han transcurrido 3 épocas sin mejora en *accuracy*, `EarlyStopping` interrumpe el entrenamiento.

Durante todo el proceso, la pérdida de entrenamiento (`train_loss`) decrece progresivamente (de 0,37 a 0,25), lo que indica que el modelo está aprendiendo patrones presentes en los datos de entrenamiento. Sin embargo, la inestabilidad de la pérdida de validación indica una posible falta de representatividad del conjunto de validación, o una sensibilidad del modelo a pequeñas variaciones en los datos.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.3737	0.0681	0.9920	0.0080	16:38
1	0.3972	0.1697	0.9499	0.0501	16:39
2	0.2966	0.5945	0.6834	0.3166	16:39
3	0.2482	0.4121	0.7936	0.2064	16:38

Tabla 3.6: Experimento 1: Resumen del entrenamiento

Métricas de validación interna

La validación interna durante el entrenamiento se hizo con 4 pacientes, como se ha indicado previamente. Se calcularon la matriz de confusión por *tile* (Figura 23) y por paciente (Figura 24), así como los 9 *tiles* con mayor función de pérdida (Figura 25) aplicando la técnica *hard voting* explicada anteriormente.

Como se observa, el modelo clasificó correctamente el subtipo histológico de los 4 pacientes, lo que indica un buen rendimiento sobre el conjunto de validación. A nivel de *tile*, únicamente se cometieron 4 errores: 1 *tile* de DDLPS fue clasificado como UPS y 3 *tiles* de UPS se clasificaron como DDLPS.

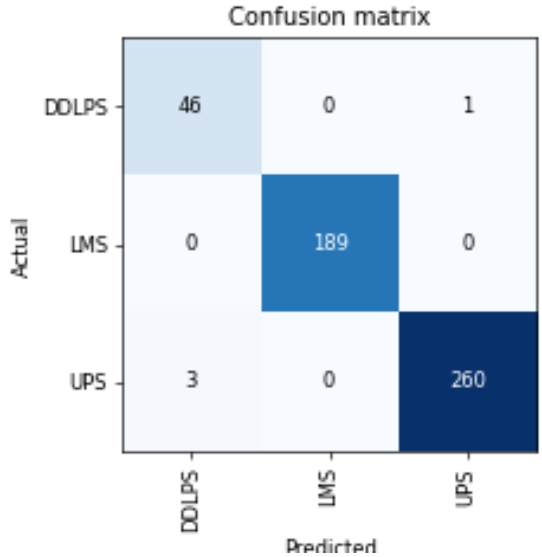


Figura 23: Experimento 1. Validación interna: Matriz de confusión a nivel de *tile*

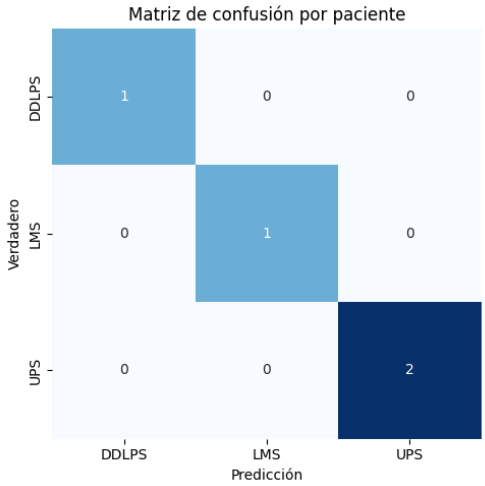


Figura 24: Experimento 1. Validación interna: Matriz de confusión a nivel de paciente

En cuanto a los *tiles* con mayor pérdida, se observa que solo falla en dos predicciones. UPS se clasifica erróneamente como DDLPS con una pérdida de 1,46 y una probabilidad de 0,76, y DDLPS se clasifica erróneamente como UPS con una pérdida de 0,73 y una probabilidad de 0,52. La pérdida más alta (1,46) corresponde a una predicción errónea realizada con alta probabilidad (0,75), es decir, el modelo estaba seguro de su decisión, pero se equivocó. Esto es especialmente problemático en el contexto clínico, ya que supone un falso positivo con alta confianza.

En el resto de *tiles*, especialmente los correctamente clasificados, la probabilidad media se sitúa en torno a 0,5. En estos casos, el modelo acierta, pero con poca confianza, lo que puede indicar que el *tile* no contiene suficientes características discriminativas o representa zonas poco informativas del tumor. Por ejemplo, la segunda imagen de la primera fila de la Figura 25 corresponde a un borde del tumor con poca proporción de tejido.

Por otra parte, se observa que hay muchos *tiles* del subtipo UPS, lo que podría indicar una mayor dificultad del modelo para predecir este subtipo. Además, se observa que muchos de estos *tiles* tienen un aspecto similar, aparentemente con tejido adiposo o escasa celularidad, lo que puede dificultar la diferenciación entre DDLPS y UPS. No obstante, es fundamental realizar este análisis con la ayuda de un patólogo experto.

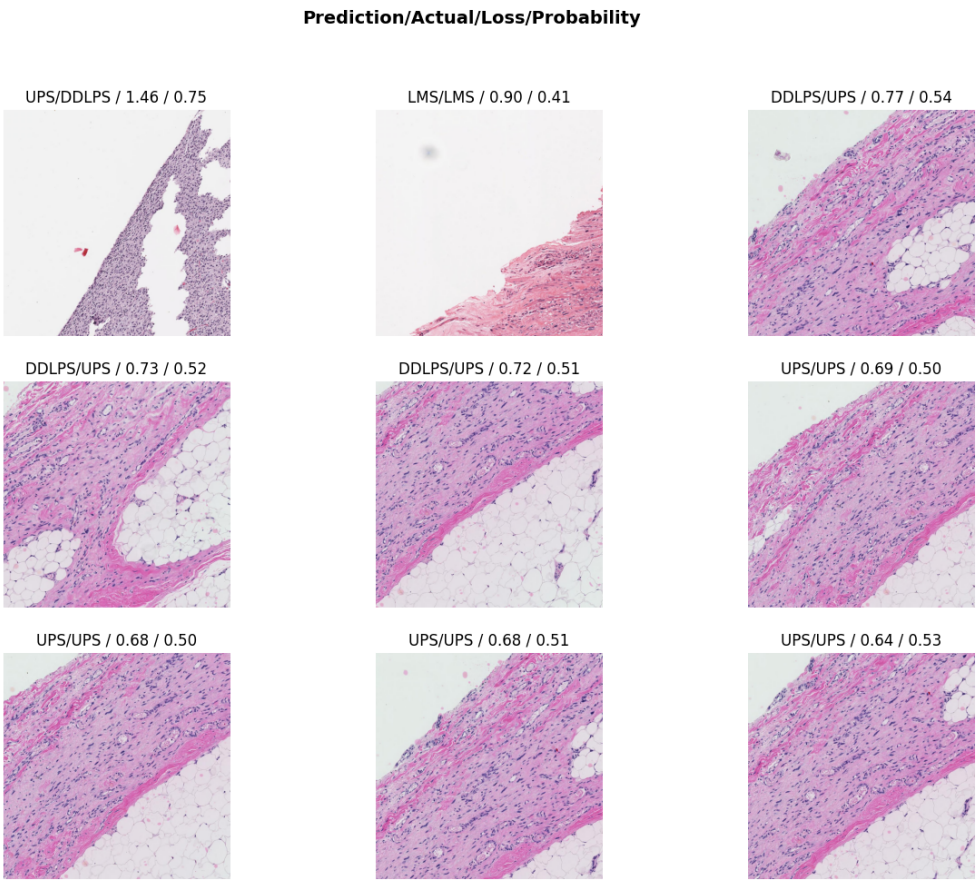


Figura 25: Experimento 1. Validación interna: *Tiles* con mayor pérdida

3.3.5. Evaluación externa

Aunque en la implementación de referencia no se implementa la evaluación externa del modelo —es decir, la evaluación del modelo usando datos no vistos durante el entrenamiento—, es fundamental hacerlo para evaluar la capacidad real de generalización del modelo.

Como se ha comentado previamente, el modelo hace la predicción a nivel de *tile*, por lo que es necesario agregar estas predicciones para obtener una única clasificación por paciente. Para los datos de validación interna se empleó votación por mayoría (*hard voting*), como se ha explicado anteriormente. Sin embargo, en la inferencia sobre datos externos se optó por emplear votación suave (*soft voting*), que considera las distribuciones de probabilidad generadas por el modelo en lugar de únicamente las clases predichas. Concretamente, se aplicó la función `softmax` a la salida del modelo para cada *tile*, obteniendo una distribución de probabilidad por clase. A continuación, se calcularon las medias de estas probabilidades entre todos los *tiles* del paciente, y se seleccionó como predicción final la clase con mayor probabilidad promedio. De esta forma, cada *tile* contribuye al resultado global en función del grado de confianza del modelo, a diferencia del *hard voting*, donde cada *tile* aporta un voto con el mismo peso independientemente de su incertidumbre.

Se evaluó el modelo utilizando el conjunto de evaluación externa `TestSet-1`, observando un empeoramiento del rendimiento con respecto al conjunto de validación interna. Mientras que en validación se obtuvo una `accuracy` del 100 %, como se muestra en la Figura 26, en inferencia con datos no vistos se obtuvo un 40 % de exactitud. Esta escasa capacidad de generalización indicaba un sobreajuste al conjunto de entrenamiento y validación.

Solo se clasificó correctamente el único paciente de DDLPS y 1 de los 2 pacientes de LMS, fallando hacia DDLPS, como se observa en la Tabla 3.7. Los 2 pacientes de UPS también se clasificaron erróneamente como DDLPS, lo que sugiere un sesgo del modelo hacia este subtipo. Además, como se muestra en la Tabla 3.7, algunos de los errores se cometieron con alta confianza, especialmente en el paciente `TCGA-QQ-A5V9` (0,95). Por último, no se observa una relación entre el número de *tiles* y la certeza de la predicción. Por ejemplo, el modelo falla con `TCGA-QQ-A5V9` (233 *tiles*) y acierta con `TCGA-DX-A2J0` (31 *tiles*).

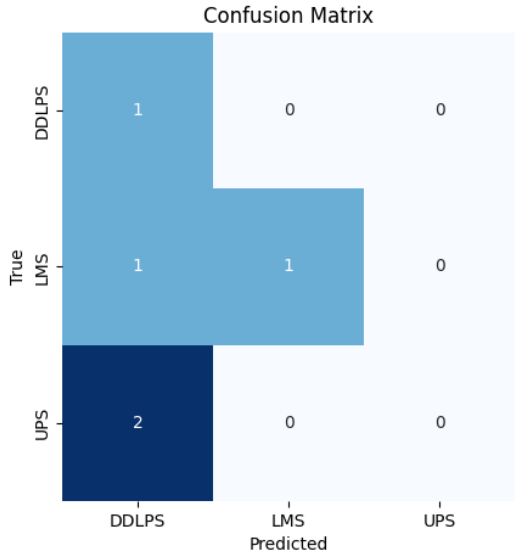


Figura 26: Experimento 1. Evaluación externa sobre TestSet-1: Matriz de confusión a nivel de paciente

Paciente	**Nº tiles**	**Predicción**	**Real**	**Confianza**	**Tiempo (s)**
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.87	1.75
TCGA-FX-A3NJ	64	DDLPS	LMS	0.59	3.60
TCGA-QQ-A5V9	233	DDLPS	UPS	0.95	13.54
TCGA-QQ-A5VB	289	LMS	LMS	0.49	16.07
TCGA-VT-A80J (DX2)	154	DDLPS	UPS	0.53	8.72

Tabla 3.7: Experimento 1: Desviación estándar por clase y paciente (variabilidad entre *tiles*)

También se calculó la distribución de probabilidades y desviación estándar por clase, como se observa en la Tabla 3.8. En primer lugar, el paciente `TCGA-DX-A2J0`, correctamente clasificado con una confianza de 0,87 presenta confianzas bajas para el resto de subtipos, lo que indica un buen desempeño para el subtipo DDLPS. Además, las distribuciones estándar son intermedias, mostrando relativamente poca variabilidad en la predicción entre *tiles*, sobre todo en el caso de UPS.

No ocurre lo mismo con el otro paciente correctamente clasificado, `TCGA-QQ-A5VB`, donde se observan dudas entre LMS, con una confianza del 49 %, y UPS, con un 35 %. Además, como se muestra en la Figura 27, este paciente presenta la mayor dispersión en las predicciones. Esto podría deberse a que se trata de un caso más complicado a nivel histológico, lo cual podría tener sentido teniendo en cuenta que se trata del subtipo UPS.

Por otro lado, en los casos mal clasificados, en general se observa que el modelo presenta una alta confianza y desviaciones estándar bajas. Este comportamiento sugiere que el modelo ha aprendido durante el entrenamiento patrones que realmente no eran discriminativos. Esto probablemente se debe principalmente al reducido tamaño del dataset, en especial del conjunto de validación. Con solo cuatro pacientes de validación —escogidos además sin ayuda de un patólogo experto que escoja los casos más representativos—, es difícil que el modelo aprenda de forma efectiva a generalizar sobre datos no vistos.

Paciente	DDLPS	LMS	UPS	Std DDLPS	Std LMS	Std UPS
TCGA-DX-A2J0	0.87	0.08	0.05	0.19	0.12	0.07
TCGA-FX-A3NJ	0.59	0.41	0.00	0.13	0.13	0.01
TCGA-QQ-A5V9	0.95	0.01	0.04	0.10	0.03	0.08
TCGA-QQ-A5VB	0.35	0.49	0.16	0.22	0.33	0.23
TCGA-VT-A80J (DX2)	0.53	0.47	0.00	0.20	0.20	0.01

Tabla 3.8: Experimento 1. TestSet-1: Distribución de probabilidades y desviación estándar por clase

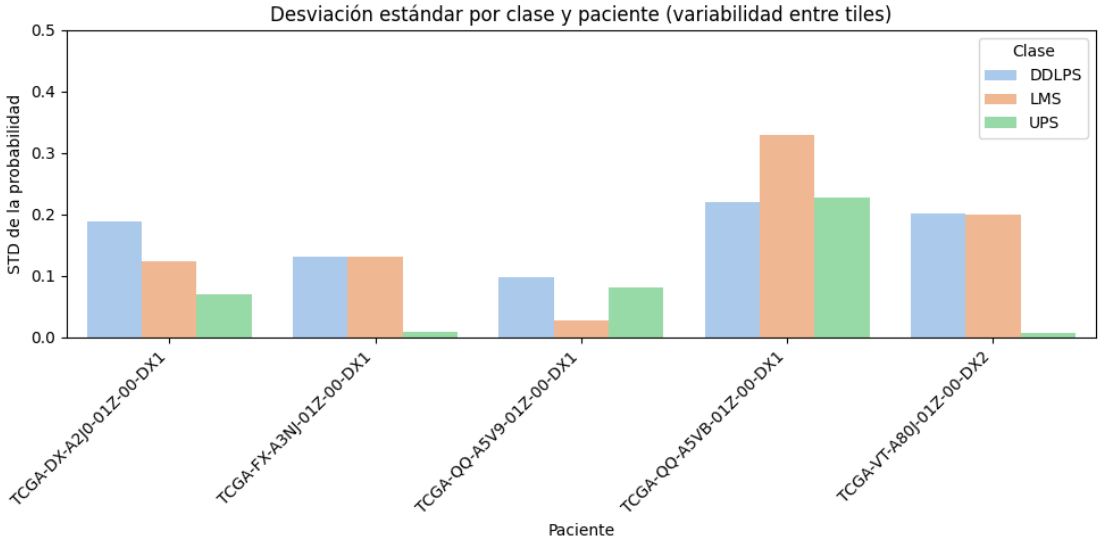


Figura 27: Experimento 1. Evaluación externa sobre TestSet-1: Desviación estándar por clase y paciente

A la vista de estos resultados, era evidente que el tamaño reducido del dataset representaba una limitación importante. Por un lado, el modelo se entrenaba sobre un conjunto de datos muy limitado, dificultando el aprendizaje de patrones generalizables. Por otro, el conjunto de validación era poco representativo, lo que dificultaba una validación fiable del modelo. En consecuencia, el modelo estaba sobreajustado y presentaba una escasa capacidad de generalización.

3.4. Segundo experimento: validación cruzada

Tras los resultados del primer experimento, se hizo evidente que el tamaño del dataset era demasiado limitado, siendo especialmente problemático el reducido tamaño de pacientes en el conjunto de validación —solo cuatro—. Esto implicaba que los hiperparámetros del modelo se ajustaban en función de un conjunto de datos poco representativo, lo que comprometía su capacidad de generalización.

Aunque una posible solución podría haber sido aumentar el porcentaje de pacientes destinados a la validación, esto habría reducido aún más el número de casos para el entrenamiento, que ya era escaso, dificultando el aprendizaje del modelo. Ante la imposibilidad de obtener más datos, se optó por seguir la estrategia de validación cruzada, concretamente *Stratified Group K-fold*. Esta técnica permite validar el modelo sobre más pacientes manteniendo la partición por pacientes y la distribución estratificada de clases en cada *fold*.

En el caso concreto de este proyecto, el objetivo de aplicar validación cruzada es entrenar k modelos independientes, cada uno validado sobre una partición distinta del conjunto de pacientes (*folds*), para finalmente combinar los k modelos en un único modelo ensamblado con mayor capacidad de generalización.

3.4.1. Conjunto de entrenamiento y validación

Se optó por usar 5 *folds* para mantener el equilibrio entre beneficio-coste, concretamente tiempo de entrenamiento y coste computacional. Como se ha comentado previamente, se emplea *Stratified Group k-fold* para mantener la agrupación de *tiles* por paciente y evitar que estos aparezcan en entrenamiento y validación en el mismo *fold*.

Como resultado, para cada uno de los cinco folds, se generaron los conjuntos de entrenamiento y validación que se muestran de la Tabla 3.9 a la Tabla 3.18.

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HO	DDLPS	148
TCGA-3B-A9HX	LMS	252
TCGA-3B-A9I3	LMS	259
TCGA-DX-A1KW	DDLPS	93
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A23T	DDLPS	311
TCGA-DX-A23Y	DDLPS	245
TCGA-DX-A240	DDLPS	91
TCGA-DX-A2J1	DDLPS	176
TCGA-DX-A3U5	DDLPS	47
TCGA-DX-A3U7	LMS	438
TCGA-DX-A3U8	UPS	424
TCGA-DX-A3UB	LMS	353
TCGA-DX-A3UF	LMS	298
TCGA-DX-A48J	LMS	189
TCGA-DX-A48N	DDLPS	165
TCGA-DX-A48R	LMS	486
TCGA-HS-A5N7	LMS	123
TCGA-IE-A4EJ	UPS	111
TCGA-K1-A3PO	LMS	50
TCGA-K1-A42W	LMS	573
TCGA-MB-A5YA	UPS	207
TCGA-QC-A6FX	UPS	56
TCGA-SG-A6Z4	UPS	161
TCGA-VT-A80J	UPS	72
TCGA-VT-AB3D	UPS	186

Tabla 3.9: Fold 0: subconjunto de entrenamiento

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HL	DDLPS	146
TCGA-DX-A1KX	UPS	464
TCGA-DX-A3LS	DDLPS	331
TCGA-DX-A3UC	LMS	471
TCGA-DX-A8BZ	UPS	533
TCGA-MB-A5Y9	UPS	314
TCGA-Z4-AAPG	UPS	78

Tabla 3.10: Fold 0: subconjunto de validación

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HL	DDLPS	146
TCGA-VT-AB3D	UPS	186
TCGA-K1-A42W	LMS	573
TCGA-DX-A3U8	UPS	424
TCGA-DX-A3UC	LMS	471
TCGA-DX-A3U7	LMS	438
TCGA-DX-A23T	DDLPS	311
TCGA-3B-A9I3	LMS	259
TCGA-DX-A1KW	DDLPS	93
TCGA-QC-A6FX	UPS	56
TCGA-IE-A4EJ	UPS	111
TCGA-DX-A240	DDLPS	91
TCGA-DX-A48N	DDLPS	165
TCGA-3B-A9HX	LMS	252
TCGA-DX-A23Y	DDLPS	245
TCGA-DX-A48J	LMS	189
TCGA-K1-A3PO	LMS	50
TCGA-DX-A3UB	LMS	353
TCGA-DX-A2J1	DDLPS	176
TCGA-DX-A8BZ	UPS	533
TCGA-DX-A48R	LMS	486
TCGA-DX-A1KX	UPS	464
TCGA-MB-A5Y9	UPS	314
TCGA-SG-A6Z4	UPS	161
TCGA-DX-A3LS	DDLPS	331
TCGA-Z4-AAPG	UPS	78

Tabla 3.11: Fold 1: subconjunto de entrenamiento

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HO	DDLPS	148
TCGA-MB-A5YA	UPS	207
TCGA-HS-A5N7	LMS	123
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A3UF	LMS	298
TCGA-VT-A80J	UPS	72
TCGA-DX-A3U5	DDLPS	47

Tabla 3.12: Fold 1: subconjunto de validación

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HL	DDLPS	146
TCGA-3B-A9HO	DDLPS	148
TCGA-3B-A9I3	LMS	259
TCGA-DX-A1KW	DDLPS	93
TCGA-DX-A1KX	UPS	464
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A23Y	DDLPS	245
TCGA-DX-A240	DDLPS	91
TCGA-DX-A2J1	DDLPS	176
TCGA-DX-A3LS	DDLPS	331
TCGA-DX-A3U5	DDLPS	47
TCGA-DX-A3UB	LMS	353
TCGA-DX-A3UC	LMS	471
TCGA-DX-A3UF	LMS	298
TCGA-DX-A8BZ	UPS	533
TCGA-DX-A3U8	UPS	424
TCGA-DX-A48R	LMS	486
TCGA-HS-A5N7	LMS	123
TCGA-IE-A4EJ	UPS	111
TCGA-K1-A3PO	LMS	50
TCGA-K1-A42W	LMS	573
TCGA-MB-A5Y9	UPS	314
TCGA-MB-A5YA	UPS	207
TCGA-VT-AB3D	UPS	186
TCGA-VT-A80J	UPS	72
TCGA-Z4-AAPG	UPS	78
TCGA-SG-A6Z4	UPS	161

Tabla 3.13: Fold 2: subconjunto de entrenamiento

Paciente	Subtipo	Nº tiles
TCGA-DX-A3U7	LMS	438
TCGA-3B-A9HX	LMS	252
TCGA-QC-A6FX	UPS	56
TCGA-DX-A23T	DDLPS	311
TCGA-DX-A48N	DDLPS	165
TCGA-DX-A48J	LMS	189

Tabla 3.14: Fold 2: subconjunto de validación

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HL	DDLPS	146
TCGA-3B-A9HO	DDLPS	148
TCGA-3B-A9HX	LMS	252
TCGA-DX-A1KW	DDLPS	93
TCGA-DX-A1KX	UPS	464
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A23T	DDLPS	311
TCGA-DX-A3LS	DDLPS	331
TCGA-DX-A3U5	DDLPS	47
TCGA-DX-A3U7	LMS	438
TCGA-DX-A3U8	UPS	424
TCGA-DX-A3UB	LMS	353
TCGA-DX-A3UC	LMS	471
TCGA-DX-A3UF	LMS	298
TCGA-DX-A48J	LMS	189
TCGA-DX-A8BZ	UPS	533
TCGA-DX-A23Y	DDLPS	245
TCGA-DX-A48N	DDLPS	165
TCGA-HS-A5N7	LMS	123
TCGA-K1-A3PO	LMS	50
TCGA-K1-A42W	LMS	573
TCGA-MB-A5YA	UPS	207
TCGA-MB-A5Y9	UPS	314
TCGA-QC-A6FX	UPS	56
TCGA-Z4-AAPG	UPS	78
TCGA-VT-A80J	UPS	72

Tabla 3.15: Fold 3: subconjunto de entrenamiento

Paciente	Subtipo	Nº tiles
TCGA-DX-A240	DDLPS	91
TCGA-DX-A2J1	DDLPS	176
TCGA-IE-A4EJ	UPS	111
TCGA-VT-AB3D	UPS	186
TCGA-3B-A9I3	LMS	259
TCGA-SG-A6Z4	UPS	161
TCGA-DX-A48R	LMS	486

Tabla 3.16: Fold 3: subconjunto de validación

Paciente	Subtipo	Nº tiles
TCGA-3B-A9HL	DDLPS	146
TCGA-3B-A9HO	DDLPS	148
TCGA-3B-A9HX	LMS	252
TCGA-3B-A9I3	LMS	259
TCGA-DX-A1KX	UPS	464
TCGA-DX-A1L3	DDLPS	207
TCGA-DX-A23T	DDLPS	311
TCGA-DX-A240	DDLPS	91
TCGA-DX-A2J1	DDLPS	176
TCGA-DX-A3LS	DDLPS	331
TCGA-DX-A3U5	DDLPS	47
TCGA-DX-A3U7	LMS	438
TCGA-DX-A3UC	LMS	471
TCGA-DX-A3UF	LMS	298
TCGA-DX-A48J	LMS	189
TCGA-DX-A48N	DDLPS	165
TCGA-DX-A48R	LMS	486
TCGA-DX-A8BZ	UPS	533
TCGA-HS-A5N7	LMS	123
TCGA-IE-A4EJ	UPS	111
TCGA-MB-A5Y9	UPS	314
TCGA-MB-A5YA	UPS	207
TCGA-VT-AB3D	UPS	186
TCGA-Z4-AAPG	UPS	78
TCGA-QC-A6FX	UPS	56
TCGA-SG-A6Z4	UPS	161
TCGA-VT-A80J	UPS	72

Tabla 3.17: Fold 4: subconjunto de entrenamiento

Paciente	Subtipo	Nº tiles
TCGA-K1-A3PO	LMS	50
TCGA-DX-A23Y	DDLPS	245
TCGA-DX-A1KW	DDLPS	93
TCGA-DX-A3U8	UPS	424
TCGA-K1-A42W	LMS	573
TCGA-DX-A3UB	LMS	353

Tabla 3.18: Fold 4: subconjunto de validación

3.4.2. Resultados del entrenamiento

Se entrenaron cinco modelos independientes, uno por cada *fold*, replicando el entrenamiento del primer experimento, pero adaptado a la estructura iterativa de la validación cruzada. El número de épocas se redujo a 1 en la primera fase y a 6 en la segunda, no solo para reducir el coste computacional asociado al entrenamiento de múltiples modelos, sino también para minimizar el riesgo de sobreajuste. Aunque cada modelo se entrena exclusivamente sobre su partición correspondiente, todas las particiones provienen del mismo dataset global, lo que implica que ciertos patrones pueden aparecer repetidamente en distintas combinaciones. Por este motivo, se busca evitar que los modelos se ajusten en exceso a las particularidades del conjunto de entrenamiento de cada *fold*. A cambio, este enfoque permite validar el rendimiento del modelo de forma más robusta y favorece una mejor generalización, ya que se evalúa su comportamiento frente a múltiples combinaciones de los 33 pacientes.

Igual que en el primer experimento, se analizó el archivo generado por `CSVLogger` correspondiente a la segunda fase de entrenamiento.

El *fold 0* fue uno de los peores modelos en rendimiento y estabilidad. Como se observa en la Tabla 3.19, `train_loss` desciende progresivamente, lo que indica que está aprendiendo patrones del conjunto de entrenamiento. Sin embargo, `valid_loss` aumenta considerablemente y es inestable. Estos son signos claros de sobreajuste. Además, la `accuracy` final (44,76 %) es muy inferior a la inicial (66,92 %), mostrando una disminución en la capacidad de generalización.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	**0.4629**	**1.3902**	**0.6192**	**0.3808**	**13:59**
1	0.3454	2.8063	0.4074	0.5926	13:59
2	0.2220	1.8106	0.4942	0.5058	13:59
3	0.1087	3.2706	0.4476	0.5524	13:59

Tabla 3.19: Experimento 2. `Fold 0`: Resumen del entrenamiento

En contraste, el *fold 1* es el más exitoso en cuanto a generalización y estabilidad. Como se muestra en la Tabla 3.20, tanto `train_loss` como `valid_loss` descienden progresivamente, alcanzando una `accuracy` final de 91,74 %, mejorando todas las anteriores.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.5922	0.7135	0.7641	0.2359	15:41
1	0.4010	0.8444	0.7359	0.2641	15:41
2	0.2200	0.5812	0.7922	0.2078	15:42
3	0.1613	0.2798	0.9074	0.0926	15:42
4	0.1274	0.4257	0.9111	0.0889	15:41
5	**0.0679**	**0.3318**	**0.9174**	**0.0826**	**15:41**

Tabla 3.20: Experimento 2. `Fold 1`: Resumen del entrenamiento

El *fold 2* muestra una capacidad de aprendizaje adecuada, ya que `train_loss` desciende progresivamente. Sin embargo, `valid_loss` presenta un comportamiento inestable y no presenta una mejora clara, siendo el valor final (1,25) peor que el inicial (0,84), como se muestra en la Tabla 3.21. Por lo tanto, aunque la `accuracy` final es aceptable (69,24 %), este *fold* muestra signos moderados de sobreajuste.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	**0.5735**	**0.8401**	**0.7186**	**0.2814**	**15:15**
1	0.5053	1.6082	0.6003	0.3997	15:15
2	0.4362	1.4019	0.6712	0.3288	15:16
3	0.1458	1.2551	0.6924	0.3076	15:16

Tabla 3.21: Experimento 2. `Fold 2`: Resumen del entrenamiento

Por otra parte, el *fold 3* refleja un claro sobreajuste y pérdida de generalización. Excepto `train_loss`, que desciende progresivamente —reflejo del aprendizaje del conjunto de entrenamiento—, `valid_loss` se mantiene elevada y aumenta progresivamente, y el valor final de `accuracy` (54,69 %) es bastante peor que el inicial (76,19 %), como se observa en la Tabla 3.22.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	**0.3789**	**1.3418**	**0.7619**	**0.2381**	**15:11**
1	0.2957	1.4428	0.4517	0.5483	15:11
2	0.2335	1.5925	0.5816	0.4184	15:12
3	0.1298	1.6444	0.5469	0.4531	15:11

Tabla 3.22: Experimento 2. `Fold 3`: Resumen del entrenamiento

El *fold 4* muestra un comportamiento similar al *fold 3*, aunque algo mejor en cuanto a rendimiento general. El valor de `train_loss` desciende progresivamente, pero `valid_loss` se mantiene elevada e inestable, mostrando signos de sobreajuste —aunque con un valor final (0,8782) ligeramente mejor que el inicial (0,9286)—. Por otro lado, el valor de `accuracy` final (60,70 %) es ligeramente peor que el inicial (63,41 %) y que el valor máximo alcanzado (65,82 %) en la época 3, como se muestra en la Tabla 3.23.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.5239	0.9286	0.6341	0.3659	14:48
1	0.3772	1.7952	0.4327	0.5673	14:48
2	0.2823	1.1152	0.6133	0.3867	14:49
3	**0.1237**	**0.9231**	**0.6582**	**0.3418**	**14:49**
4	0.0903	1.1174	0.5616	0.4384	14:48
5	0.0504	0.8782	0.6070	0.3930	14:48

Tabla 3.23: Experimento 2. `Fold 4`: Resumen del entrenamiento

Como se ha observado, la variabilidad entre folds es bastante alta, probablemente debido a diferencias en la complejidad del conjunto de pacientes asignado a cada partición. La *accuracy* final varía entre 44,76 % (*fold 0*) y 91,74 % (*fold 1*). Este último es el más estable y robusto, seguido por el *fold 2*. En cambio, los *folds* 0, 3 y 4 presentan signos de sobreajuste, especialmente los *folds* 0 y 3.

Métricas de validación interna

Para evaluar el rendimiento general del segundo experimento, se cargaron las predicciones generadas durante la validación cruzada de cada uno de los *folds* y se concatenaron para calcular métricas globales y generar la matriz de confusión por paciente.

Como se puede observar en la Figura 28, el modelo funciona especialmente bien reconociendo el subtipo LMS, clasificando correctamente 10 de los 11 pacientes. En cambio, el subtipo DDLPS presenta mayor dificultad: se clasificaron correctamente 7 de los 11 pacientes, confundiendo los 4 restantes con LMS. Por otra parte, el modelo también presenta dificultades con el subtipo UPS, con 7 aciertos y 4 errores, de los cuales 3 fueron clasificados como LMS y 1 como DDLPS.

Puede parecer que este modelo es peor comparado con el del experimento 1, en el que se obtuvo una *accuracy* del 100 % en el conjunto de validación, pero es importante recordar que estaba sobreajustado y que este comportamiento finalmente no se reflejaba en datos nuevos.

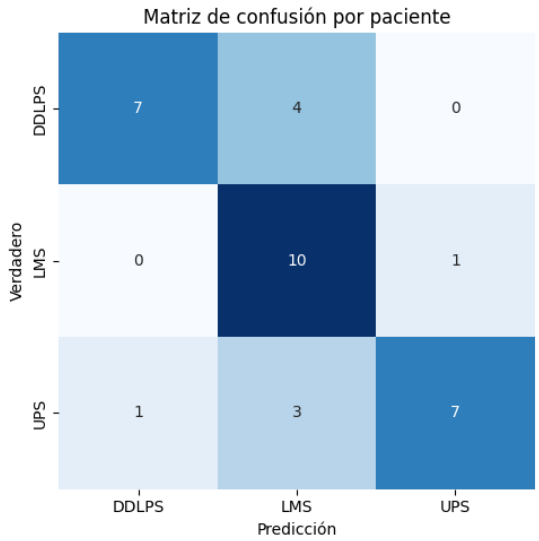


Figura 28: Experimento 2. Validación interna: Matriz de confusión a nivel de paciente

En las siguientes figuras, de la Figura 29 a la Figura 33, se muestran los *tiles* con mayor pérdida para cada *fold*. Como se observa, en el caso de los *folds* 0 y 3 los *tiles*, correspondientes al subtipo UPS, son confundidos con LMS. Además, son los que mayor pérdida presentan. Por otra parte, en los *folds* 1 y 2 se confunde el subtipo DDLPS con UPS y DDLPS respectivamente. Cabe destacar que en su mayoría se corresponden a bordes tumorales.

Aunque el análisis de los *tiles* con mayor pérdida resulta muy útil para identificar patrones más problemáticos, es necesaria la ayuda de un patólogo experto en sarcomas.

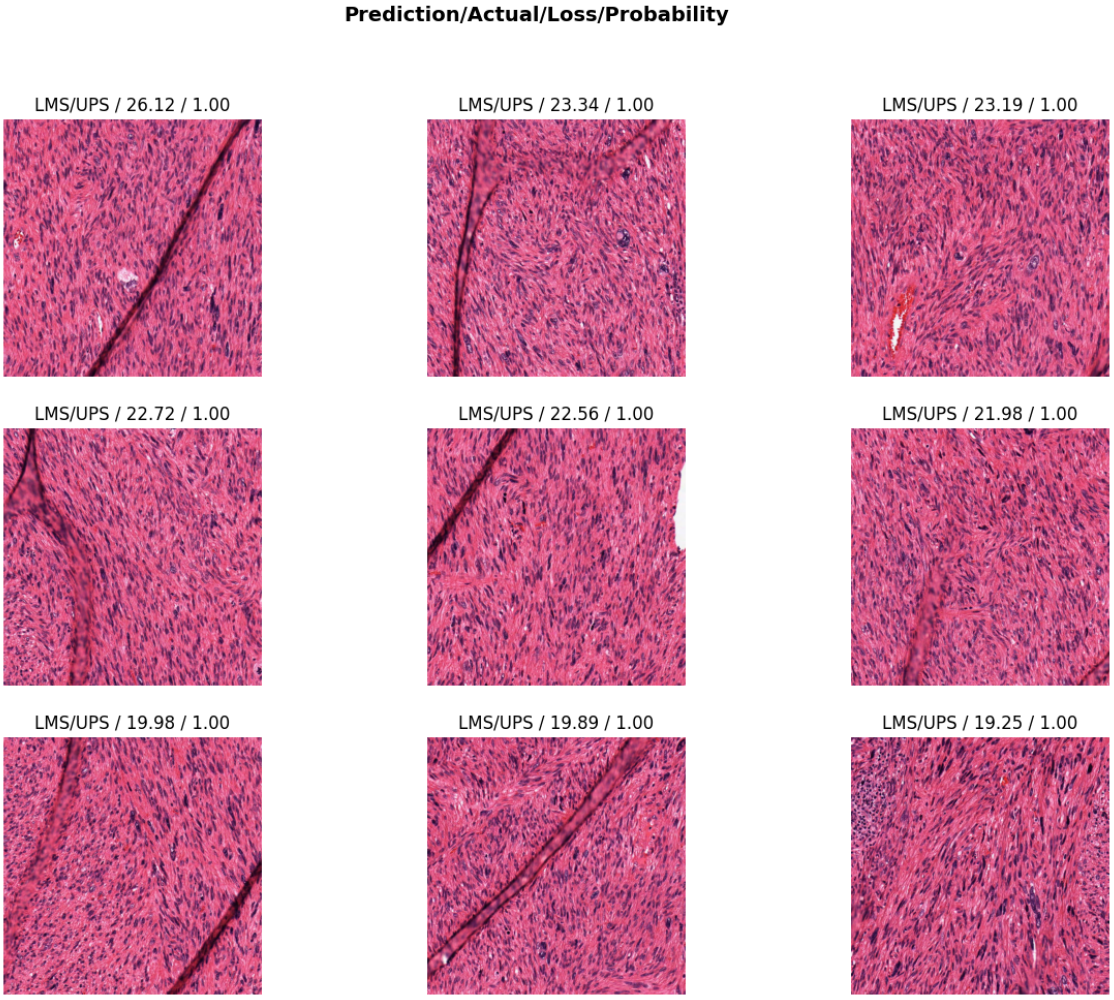


Figura 29: Experimento 2. `Fold 0`. Validación interna: *Tiles* con mayor pérdida

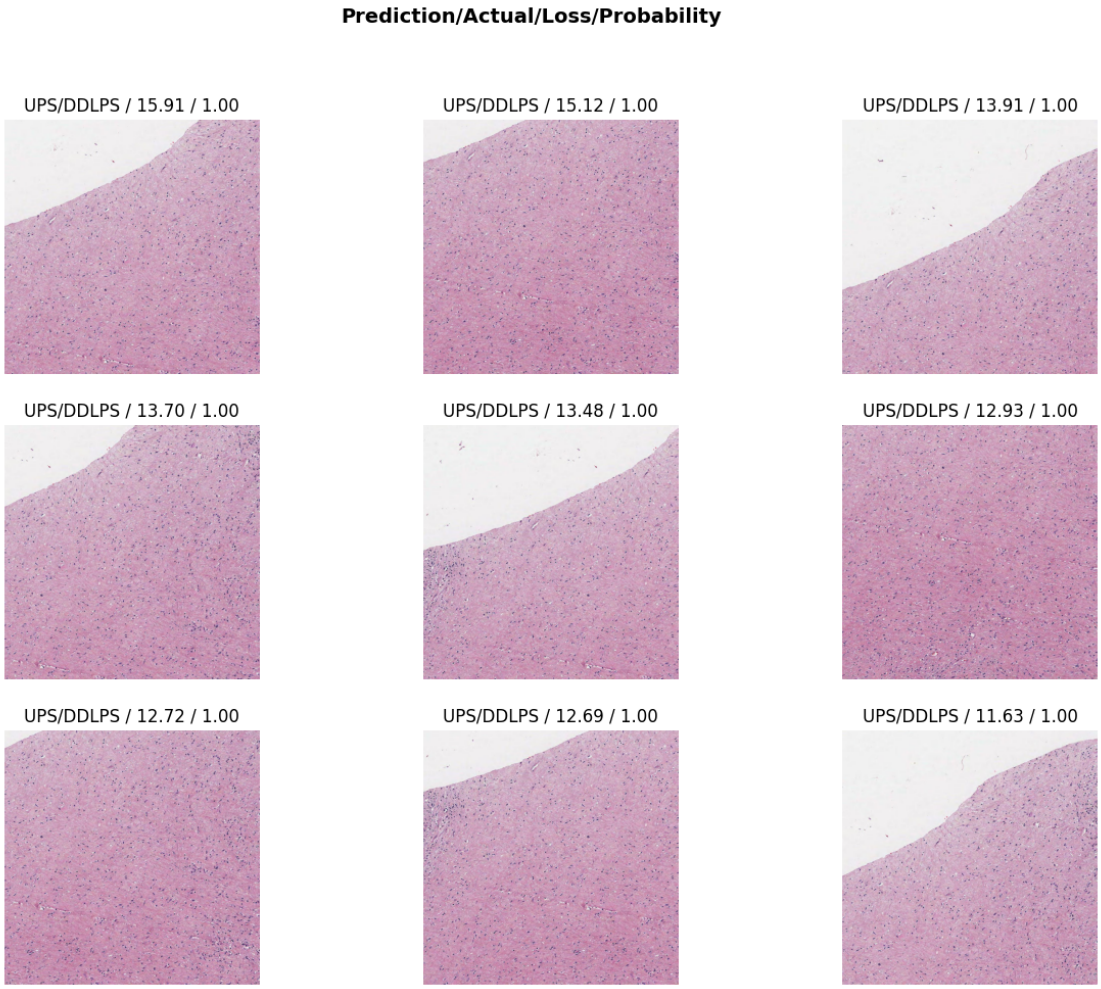


Figura 30: Experimento 2. `Fold 1`. Validación interna: *Tiles* con mayor pérdida

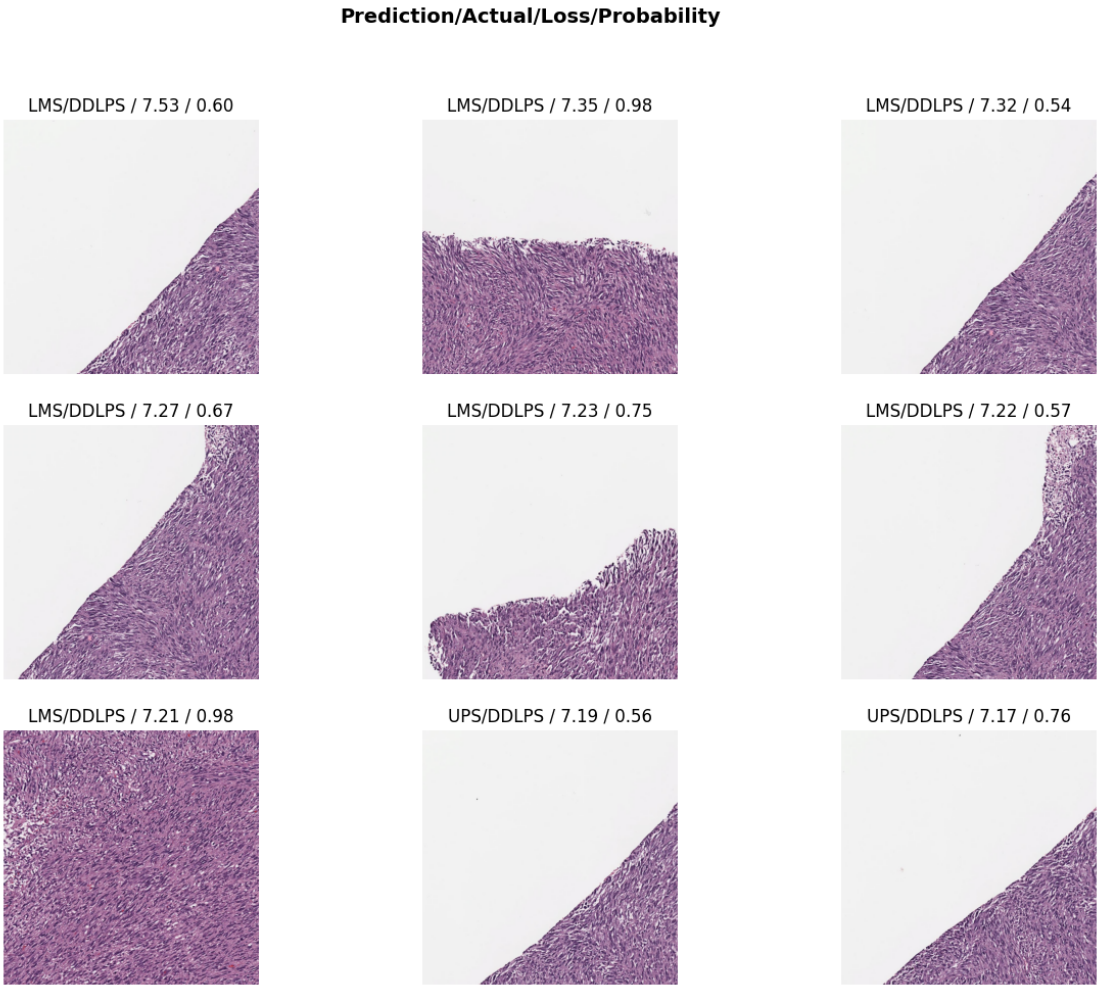


Figura 31: Experimento 2. `Fold 2`. Validación interna: *Tiles* con mayor pérdida

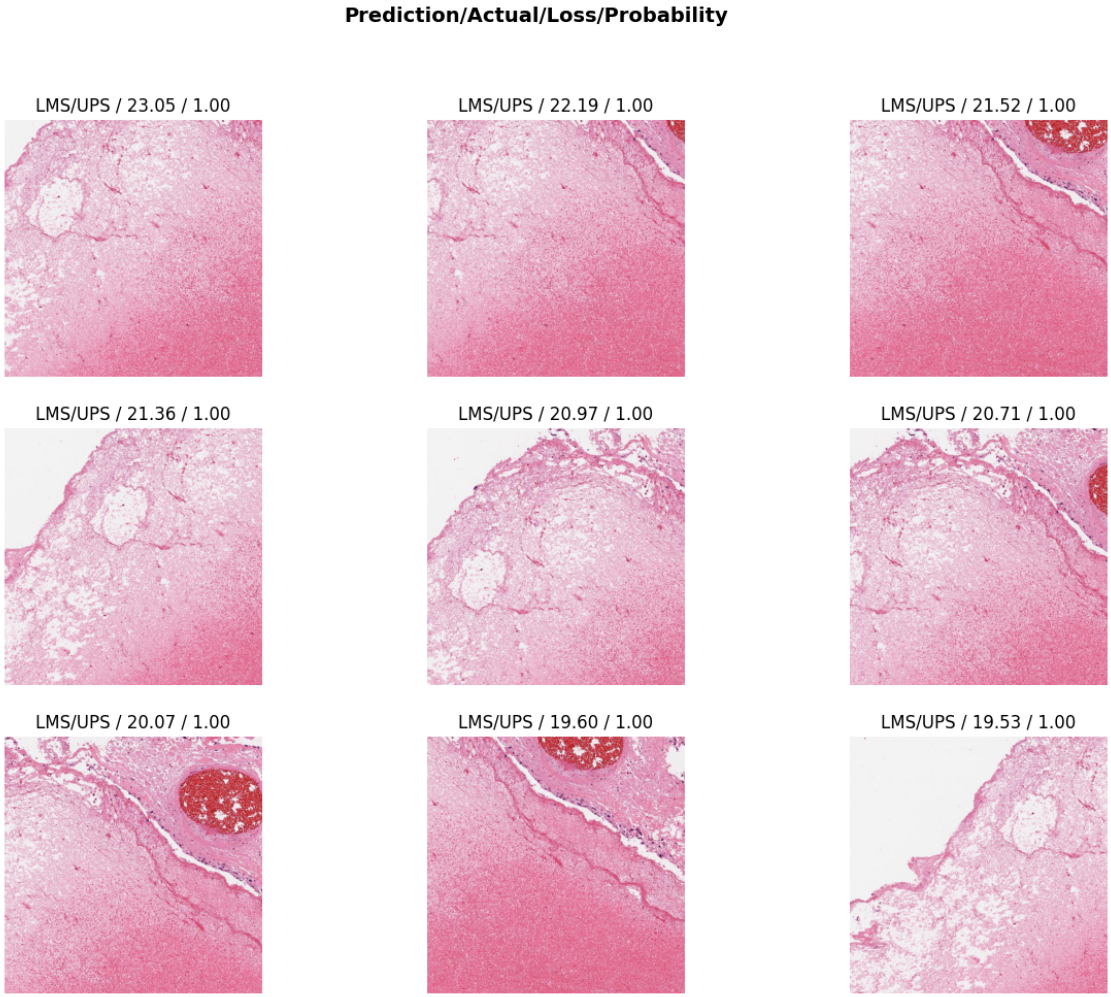


Figura 32: Experimento 2. `Fold 3`. Validación interna: *Tiles* con mayor pérdida

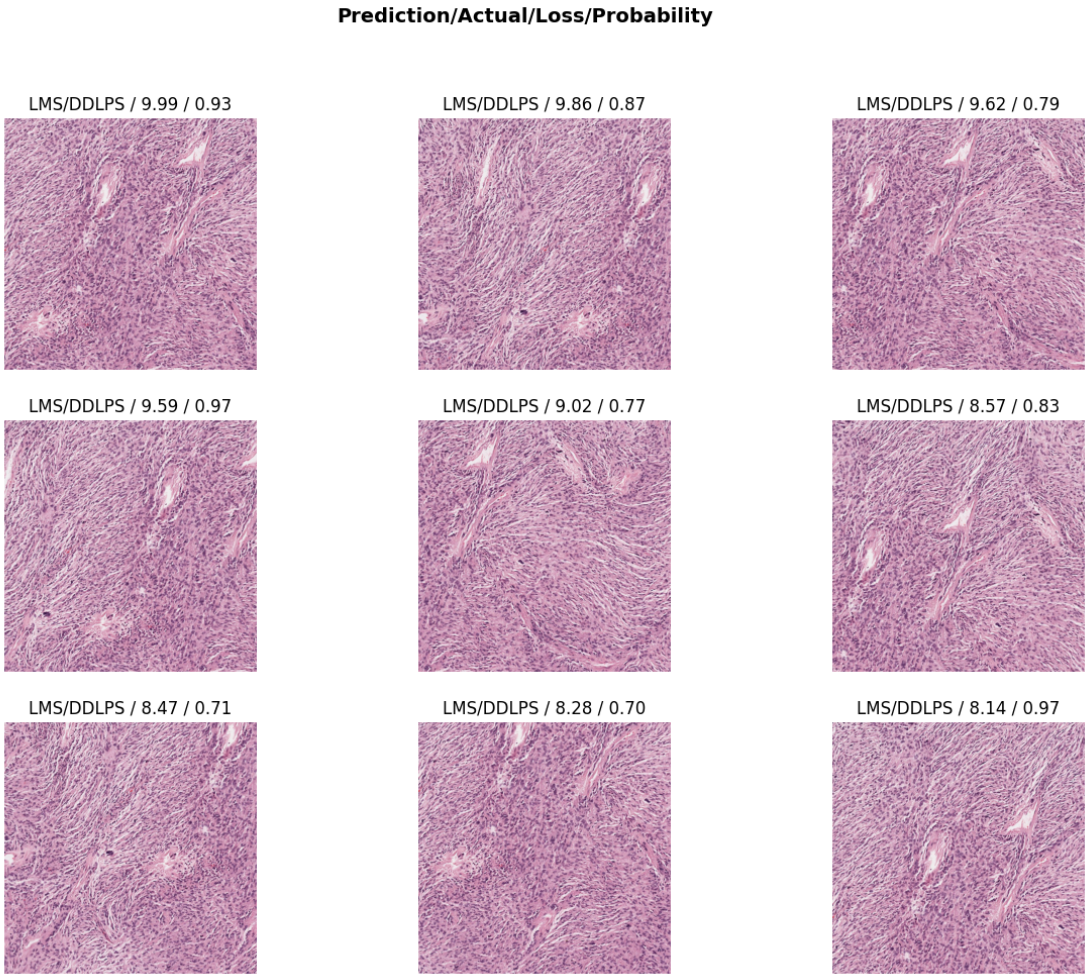


Figura 33: Experimento 2. `Fold` 4. Validación interna: *Tiles* con mayor pérdida

3.4.3. Evaluación externa

A la vista de los buenos resultados, se optó por el ensamblado de los cinco modelos con el objetivo de construir un modelo final más robusto. La idea es combinar las predicciones de los distintos modelos de forma que puedan aportar información complementaria. Cabe destacar que otra opción hubiera sido utilizar los resultados obtenidos durante la validación cruzada para ajustar los hiperparámetros y entrenar un único modelo sobre el dataset completo.

Para combinar las predicciones de los cinco modelos se adaptó la técnica de *soft voting* empleada en el primer experimento para integrar las predicciones de los cinco modelos. Para cada paciente, se evaluaron todos sus *tiles* con cada modelo del ensemble, transformando las salidas mediante la función *softmax* para obtener distribuciones de probabilidad por clase. Primero, se promediaron estas probabilidades entre los distintos modelos, obteniendo una predicción consenso por *tile*. A continuación, se promediaron las predicciones de todos los *tiles* del paciente y se asignó como predicción final la clase con mayor probabilidad media. De esta forma, cada *tile* y cada modelo contribuyen de forma ponderada a la decisión final.

Primero, se probó el conjunto TestSet-1 y se obtuvieron muy buenos resultados, como se muestra en la Figura 34, con una precisión del 80 %, que claramente mejoraba a la del primer experimento (40 %). Esto demostraba que la validación del segundo experimento había sido más exigente y mejoraba la generalización del modelo. Se observa que el modelo generaliza bien para los subtipos LMS y DDLPS, aunque el subtipo UPS sigue presentando dificultades. No obstante, esto puede ser buena señal, ya que el subtipo UPS es un subtipo de exclusión, como se mencionó en la Capítulo 2, y resulta difícil de caracterizar, ya que típicamente presenta mucha heterogeneidad entre casos.

Por otra parte, como se observa en la Tabla 3.24, la confianza del modelo en las predicciones es, en general, más modesta que en el Experimento 1. Esto es especialmente destacable en la única clasificación errónea (UPS confundido con LMS), donde la confianza es la menor de todas (0,46). Esto, como se ha mencionado anteriormente, es especialmente importante en el contexto clínico, donde los errores pueden ser críticos.

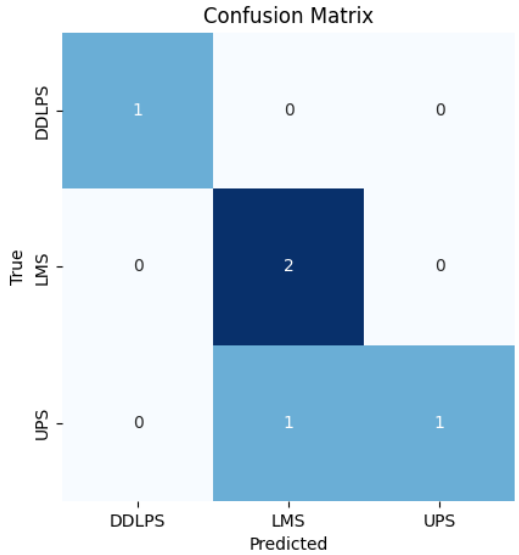


Figura 34: Experimento 2. Evaluación externa sobre TestSet-1: Matriz de confusión a nivel de paciente

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.56	7.81
TCGA-FX-A3NJ	64	LMS	LMS	0.53	16.08
TCGA-QQ-A5V9	233	UPS	UPS	0.76	137.39
TCGA-QQ-A5VB	289	LMS	LMS	0.79	145.58
TCGA-VT-A80J (DX2)	154	LMS	UPS	0.46	76.33

Tabla 3.24: Experimento 2: Predicción por paciente en el conjunto de evaluación externa TestSet-1

En la Tabla 3.25 y la Figura 35 se observa que las predicciones realizadas son bastante estables y que las desviaciones estándar por clase presentan una distribución más homogénea y valores más bajos que en el primer experimento, lo que sugiere una mejor calibración del modelo. Cabe mencionar que el único paciente clasificado erróneamente, `TCGA-VT-A80J (DX2)`, obtuvo una probabilidad de 0,38 para el subtipo correcto (UPS), la segunda más alta tras LMS (0,46), que fue la clase predicha. En general, la clase con mayor probabilidad suele presentar una desviación estándar igual o inferior a la del resto, lo que indica una mayor consistencia entre *tiles* a favor de la predicción dominante.

Paciente	**DDLPS**	**LMS**	**UPS**	**Std_DDLPS**	**Std_LMS**	**Std_UPS**
TCGA-DX-A2J0	0.56	0.25	0.19	0.14	0.03	0.12
TCGA-FX-A3NJ	0.22	0.53	0.25	0.09	0.11	0.1120
TCGA-QQ-A5V9	0.12	0.12	0.76	0.11	0.06	0.08
TCGA-QQ-A5VB	0.02	0.79	0.20	0.02	0.15	0.14
TCGA-VT-A80J (DX2)	0.16	0.46	0.38	0.06	0.16	0.13

Tabla 3.25: Experimento 2: Distribución de probabilidades por clase y desviaciones estándar por paciente

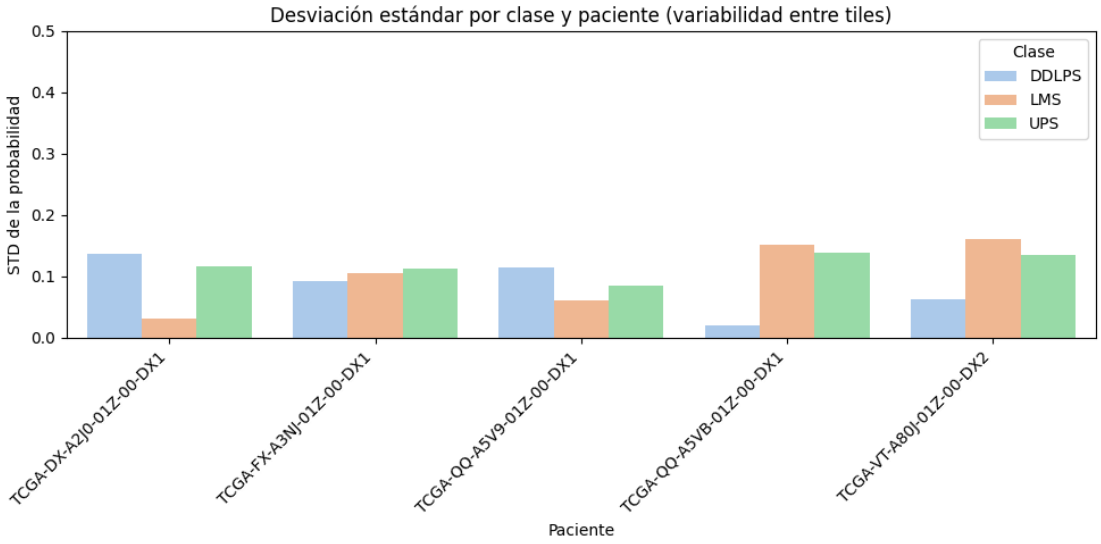


Figura 35: Experimento 2. Evaluación externa sobre TestSet-1: Desviación estándar por clase y paciente

Tras el buen rendimiento observado en el `TestSet-1`, se decidió evaluar el modelo en el `TestSet-2`, donde se obtuvieron peores resultados con una *accuracy* del 40 %. Como se observa en la Figura 36, el único caso de DDLPS es clasificado erróneamente como LMS. Lo mismo ocurre con el único caso de UPS, lo que sigue reflejando la dificultad del modelo para clasificar correctamente estos subtipos. Por otro lado, LMS se mantiene como la clase con más tasa de aciertos, aunque uno de los tres casos fue confundido con UPS.

No obstante, cabe destacar que las puntuaciones de confianza de las predicciones fueron coherentes con el acierto o error del modelo, incluso en las clasificaciones erróneas, como se observa en la Tabla 3.26. Esto permite establecer un umbral de confianza a partir del cual las predicciones puedan considerarse fiables.

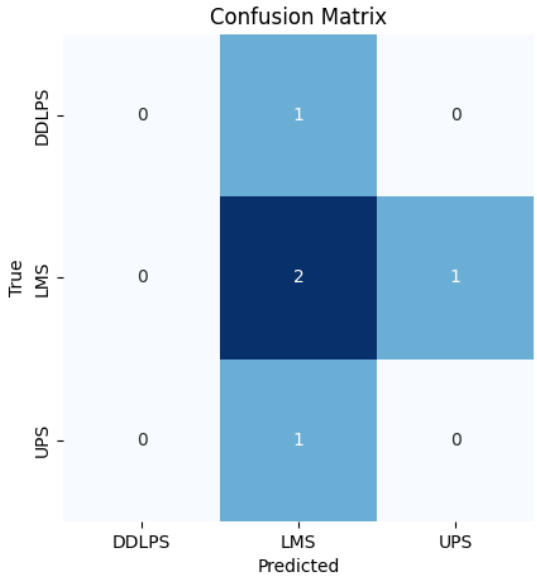


Figura 36: Experimento 2. Evaluación externa sobre TestSet-2: Matriz de confusión a nivel de paciente

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2IZ	215	LMS	DDLPS	0.48	96.55
TCGA-DX-A48U	61	LMS	LMS	0.69	62.63
TCGA-DX-A6BA (DX1)	417	UPS	LMS	0.39	160.43
TCGA-DX-A6BA (DX2)	145	LMS	LMS	0.63	55.30
TCGA-MB-A5Y8	3	LMS	UPS	0.48	2.79

Tabla 3.26: Experimento 2: Predicción por paciente en el conjunto de evaluación externa TestSet-2

Por otra parte, en la Tabla 3.27 y la Figura 37 se observa que los pacientes clasificados correctamente, `TCGA-DX-A48U` y `TCGA-DX-A6BA (DX2)`, presentan una puntuación de confianza alta (69 % y 63 % respectivamente) para el subtipo predicho correctamente y desviaciones estándar bajas, lo que indica una predicción coherente entre *tiles*. No obstante, en el caso de `TCGA-DX-A6BA (DX2)`, aunque la predicción fue correcta se observa que la desviación estándar para esa clase es más elevada que para las otras, lo que sugiere cierta heterogeneidad morfológica o incertidumbre entre *tiles*. Este paciente tiene dos WSI asociadas, y aunque la DX2 se clasifica correctamente, no ocurre lo mismo con DX1. En esta última, las puntuaciones de confianza por clase son intermedias y similares entre sí (máximo 0,36), y todas las desviaciones estándar son bajas, lo que sugiere que los tiles son consistentes entre sí, pero el modelo no es capaz de extraer patrones suficientemente distintivos, lo que puede indicar limitaciones en la representación aprendida.

En cuanto al paciente `TCGA-DX-A2IZ`, la predicción es incorrecta (LMS en lugar de DDLPS), y además la probabilidad asignada al subtipo correcto es la más baja de las tres. La baja dispersión entre tiles indica un consenso erróneo entre ellos, lo que sugiere que el modelo ha aprendido patrones no discriminativos o ha desarrollado un sesgo hacia LMS.

Paciente	**DDLPS**	**LMS**	**UPS**	**Std_DDLPS**	**Std_LMS**	**Std_UPS**
TCGA-DX-A2IZ	0.07	0.48	0.45	0.06	0.22	0.21
TCGA-DX-A48U	0.24	0.69	0.08	0.07	0.10	0.05
TCGA-DX-A6BA (DX1)	0.30	0.32	0.39	0.11	0.12	0.07
TCGA-DX-A6BA (DX2)	0.25	0.63	0.11	0.18	0.21	0.07
TCGA-MB-A5Y8	0.25	0.48	0.26	0.06	0.05	0.01

Tabla 3.27: Experimento 2. Evaluación externa sobre TestSet-2: Distribución de probabilidades por clase y desviaciones estándar por paciente

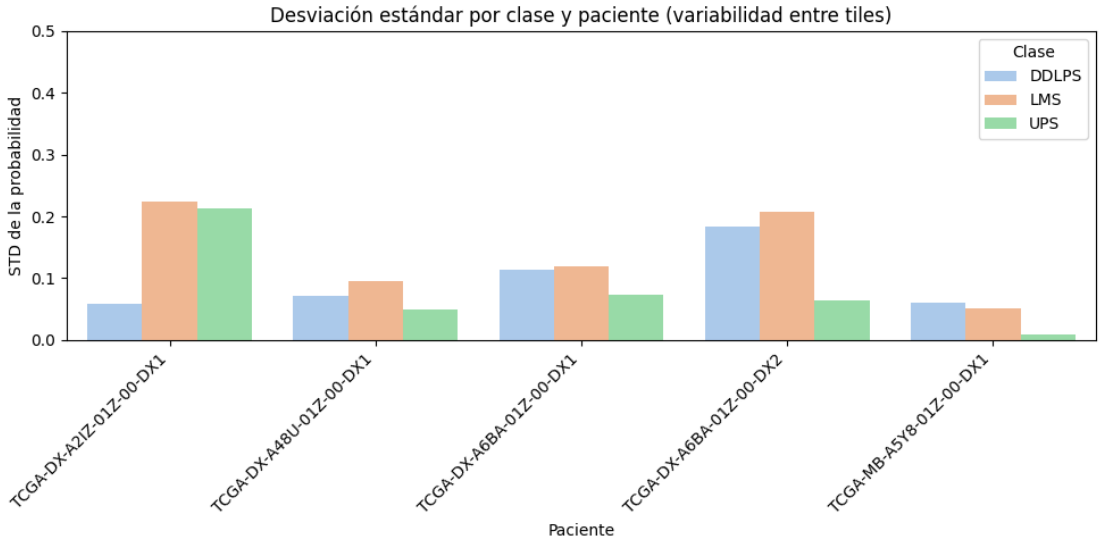


Figura 37: Experimento 2. Evaluación externa sobre TestSet-2: Desviación estándar por clase y paciente

A diferencia del experimento 1, donde la validación interna no reflejaba el comportamiento real del modelo sobre datos no vistos, en este segundo experimento se observa una mayor coherencia entre la validación y la evaluación externa. Además, las puntuaciones de confianza muestran una mayor consistencia con el acierto en la predicción, lo que sugiere un entrenamiento más robusto y permite establecer un umbral de confianza a partir del cual las predicciones pueden considerarse fiables.

3.5. Tercer experimento: reentrenamiento selectivo

En vista de que el ensamblado de los modelos entrenados con validación cruzada había dado buenos resultados, se optó por continuar explorando esta vía.

Aunque no se han incluido por su extensión, se guardaron las contribuciones de cada *fold* en la predicción final del modelo ensamblado. Estos datos, junto con el resto de la información analizada previamente, permitieron evaluar los *folds* que presentaban un peor comportamiento y afectaban negativamente a la tasa de aciertos. Tras este análisis, se optó por reentrenar los *folds* 3 y 4, que eran los modelos que presentaban mayor dispersión en la predicción a nivel de *tile*, además de confundir frecuentemente el subtipo UPS con LMS en el caso del *fold 3*. Se hicieron diversos experimentos, como aumentar el número de épocas de entrenamiento (de 1 a 3 en el `RunPart1` y de 6 a 8 en el `RunPart2`), descartar el uso de `MixUp`, o probar con distintos valores de tasa de aprendizaje. También se eliminó el uso de *seed* para explorar otros puntos de partida.

3.5.1. Resultados de entrenamiento

Tras reentrenar el *fold 3*, no se observó mejora con respecto al entrenamiento realizado en el experimento 2. Aunque presenta una menor `valid_loss` y `train_loss`, esto parece deberse más bien a un sobreajuste del modelo, ya que la máxima `accuracy` alcanzada (74,08 %) es ligeramente menor que en el segundo experimento (76,19 %), como se observa en la Tabla 3.22 y la Tabla 3.28. En conclusión, el modelo correspondiente al experimento 2 presentaba mayor capacidad de generalización, por lo que se decidió conservar esa versión y descartar esta.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.1884	0.6969	0.7272	0.2728	15:10
1	0.1205	1.0136	0.6810	0.3190	15:09
2	0.2203	0.8819	0.7408	0.2592	15:10
3	0.0727	1.0489	0.7156	0.2844	15:09
4	0.1290	1.1837	0.6816	0.3184	15:10
5	0.1149	1.0571	0.6810	0.3190	15:09

Tabla 3.28: Experimento 3. `Fold 3`: Resumen del reentrenamiento

En cuanto al reentrenamiento del *fold 4*, también se observó que empeoraba los resultados con respecto al entrenamiento inicial del segundo experimento. En este caso, no solo se obtuvo peor `accuracy` (57,48 %) que en el experimento 2 (65,82 %), sino también peores valores de `train_loss` y `valid_loss`.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.1965	0.9856	0.5627	0.4373	14:43
1	0.2284	1.4019	0.4666	0.5334	14:43
2	0.2000	1.6700	0.5552	0.4448	14:43
3	0.1231	1.1699	0.5748	0.4252	14:44
4	0.1251	1.2919	0.5731	0.4269	14:43
5	0.1387	1.1632	0.5184	0.4816	14:43
6	0.1205	1.2310	0.5012	0.4988	14:44

Tabla 3.29: Experimento 3. `Fold 4`: Resumen del entrenamiento

En vista de que los resultados no mejoraban, se descartaron estos reentrenamientos.

3.6. Cuarto experimento: reentrenamiento completo

Mientras se realizaban pruebas, se observó que estaba disponible la GPU NVIDIA A100, con hasta 80 GB de memoria. Esto suponía una gran ventaja, pues permitía aumentar el `batch_size`, lo que podía tener como consecuencia un mejor aprendizaje.

Se realizaron diversos experimentos, logrando una configuración estable con `batch_size` de 12 y 4 procesos paralelos. Se redujo el número de procesos paralelos con respecto a los anteriores experimentos —en los que se empleaban 8— porque se observó un cuello de botella que quedó resuelto al reducir el número de procesos paralelos. Aún así, se seguía observando un cuello de botella en la carga de datos debido a la carga lenta desde Google Drive y la alta resolución de las imágenes.

Por ello, se optó por cargar todas las imágenes en la memoria RAM del entorno usando `rsync`. Aunque el proceso de subida de datos duró algo más de una hora, esto permitió acelerar considerablemente el proceso de carga de datos. Esto, sumado al cambio de GPU, supuso una disminución considerable del tiempo de entrenamiento.

En vista del beneficio en rendimiento, se decidió reentrenar los cinco *folds*, aumentando el número de épocas de 3 a 5 en la primera fase y de 8 a 10 en la segunda. Por otra parte, se recuperó el uso de *seed* para poder ajustar los hiperparámetros y comparar resultados.

3.6.1. Resultados de entrenamiento

En el caso de *fold 0*, mejoró claramente el entrenamiento del Experimento 2, obteniendo en la mejor época una `accuracy` del 82,67 % y `valid_loss` de 0,6129, como se observa en la Tabla 3.30. Por otro lado, en el Experimento 2 se obtuvo una `accuracy` del 61,92 % y `valid_loss` de 1,39 en la mejor época, como se muestra en la Tabla 3.19.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.6560	0.9989	0.5665	0.4335	03:46
1	0.6068	2.4240	0.4416	0.5584	03:46
2	0.5198	1.4097	0.5323	0.4677	03:46
3	**0.4815**	**0.6129**	**0.8267**	**0.1733**	**03:46**
4	0.4103	1.5250	0.4305	0.5695	03:46
5	0.3933	0.6481	0.7826	0.2174	03:46
6	0.3796	1.3050	0.4660	0.5340	03:46

Tabla 3.30: Experimento 4. `Fold 0`: Resumen del entrenamiento

En el *fold 1*, se observa que `EarlyStoppingCallback` ha interrumpido el entrenamiento antes que en el Experimento 2. Aunque alcanza una mejor `accuracy` (92,83 % frente a 91,74 % del segundo experimento), cae bruscamente y no se recupera. Además, se observa que `valid_loss` se mantiene más elevado, lo que sugiere sobreajuste.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	**0.7146**	**0.1746**	**0.9383**	**0.0617**	**04:16**
1	0.6585	0.4102	0.8457	0.1543	04:16
2	0.5290	0.4706	0.8040	0.1960	04:16
3	0.4762	0.9149	0.6289	0.3711	04:16

Tabla 3.31: Experimento 4. `Fold 1`: Resumen del entrenamiento

En cuanto al reentrenamiento del *fold 2*, aunque alcanza un menor `valid_loss` final que en el Experimento 2, presenta una pérdida clara de `accuracy`, como se observa en la Tabla 3.21 y Tabla 3.32. Aunque el menor `valid_loss` sugiere un mejor ajuste a los datos, no supone necesariamente una mejor generalización si no se acompaña de una mejora en `accuracy`.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	**0.7409**	**1.1786**	**0.6442**	**0.3558**	**04:09**
1	0.6453	1.2460	0.6081	0.3919	04:08
2	0.5456	1.2640	0.5790	0.4210	04:08
3	0.4740	1.0026	0.5450	0.4550	04:09

Tabla 3.32: Experimento 4. `Fold 2`: Resumen del entrenamiento

El reentrenamiento del *fold 3* supone una clara mejora frente al entrenamiento del Experimento 2, como se muestra en la Tabla 3.22 y Tabla 3.33. Aunque `train_loss` baja mucho menos, se obtienen valores mucho menores de `valid_loss` y una `accuracy` ligeramente mejor (77,07 % frente a 76,19 % del Experimento 2) y bastante más estable.

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	0.4246	0.7628	0.7000	0.3000	04:08
1	0.4691	0.9775	0.7007	0.2993	04:07
2	0.4588	0.7611	0.7633	0.2367	04:08
3	0.4366	0.7873	0.6605	0.3395	04:08
4	**0.4327**	**0.7031**	**0.7707**	**0.2293**	**04:07**
5	0.4132	0.6918	0.7286	0.2714	04:07
6	0.3984	0.7850	0.5769	0.4231	04:07
7	0.3536	0.6681	0.7156	0.2844	04:07

Tabla 3.33: Experimento 4. `Fold 3`: Resumen del entrenamiento

Por último, el *fold 4* presenta peores resultados que en el Experimento 2, como se puede observar en la Tabla 3.23 y Tabla 3.34. En concreto, la `valid_loss` es más alta en todas las épocas y la `accuracy` cae drásticamente, siendo esta inferior (54,72 %) que la del segundo experimento (65,82 %).

Época	Train Loss	Valid Loss	Accuracy	Error Rate	Tiempo (min)
0	**0.5977**	**1.0239**	**0.5472**	**0.4528**	**04:01**
1	0.5672	1.1644	0.5259	0.4741	04:02
2	0.5072	1.0788	0.5121	0.4879	04:01
3	0.4622	1.1786	0.4148	0.5852	04:01

Tabla 3.34: Experimento 4. `Fold 4`: Resumen del entrenamiento

Métricas de validación interna

Como se observa en la Figura 38, el modelo parece tener un rendimiento ligeramente superior al del Experimento 2. Destaca especialmente que clasifica el subtipo DDLPS con mayor tasa de aciertos (10 aciertos, frente a los 7 del modelo del Experimento 2).

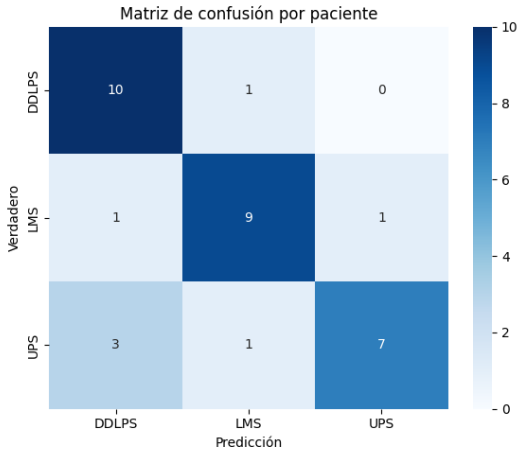


Figura 38: Experimento 4. Validación interna: Matriz de confusión a nivel de paciente

A continuación, de la Figura 39 a la Figura 43 se muestran los *tiles* con mayor pérdida. En los *folds* 0, 1 y 3 predomina la confusión del subtipo UPS, que se clasifica erróneamente como DDLPS. En el caso del *fold 2* se observa la clasificación errónea de DDLPS como LMS y presenta los valores más altos de pérdida. Por último, el *fold 4* confunde los subtipos LMS y DDLPS de forma bidireccional.

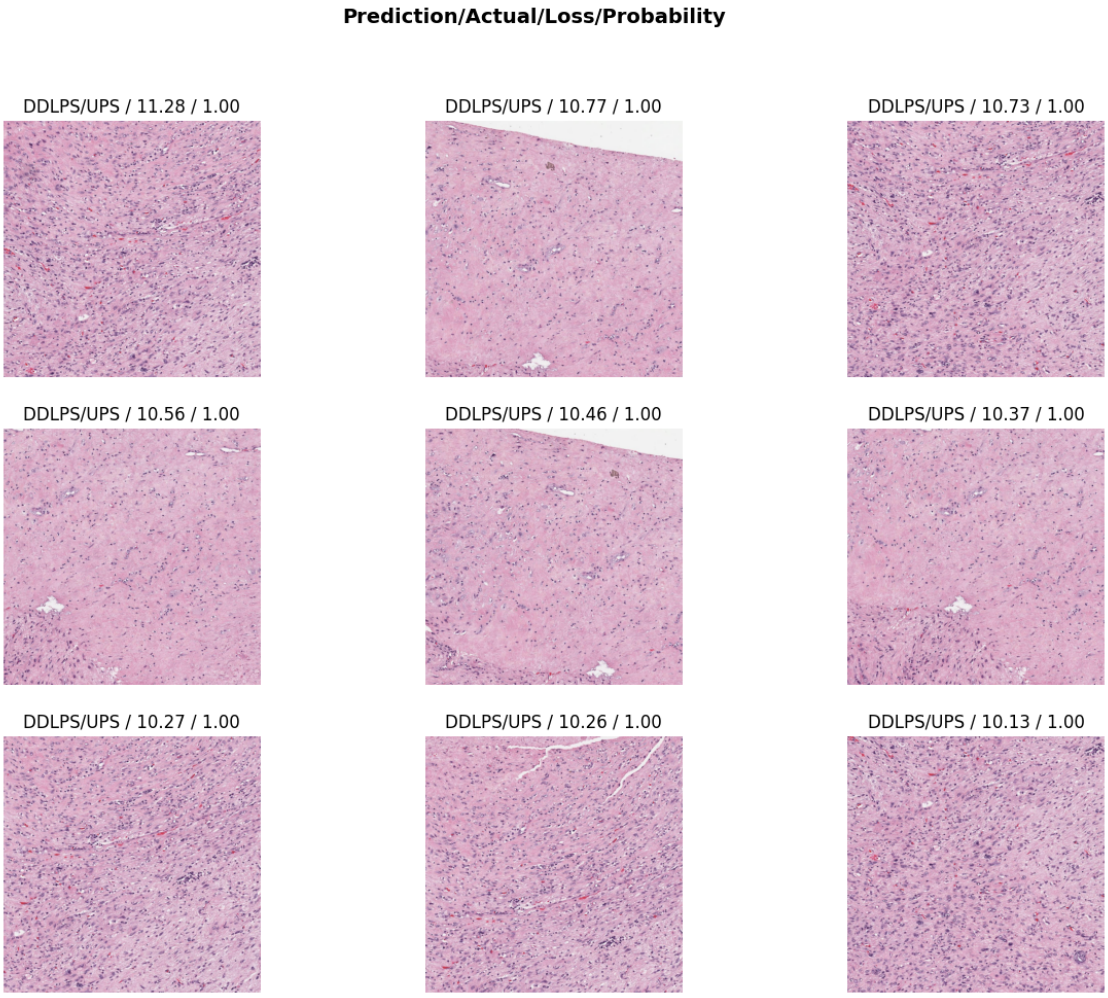


Figura 39: Experimento 4. `Fold 0`. Validación interna: *Tiles* con mayor pérdida

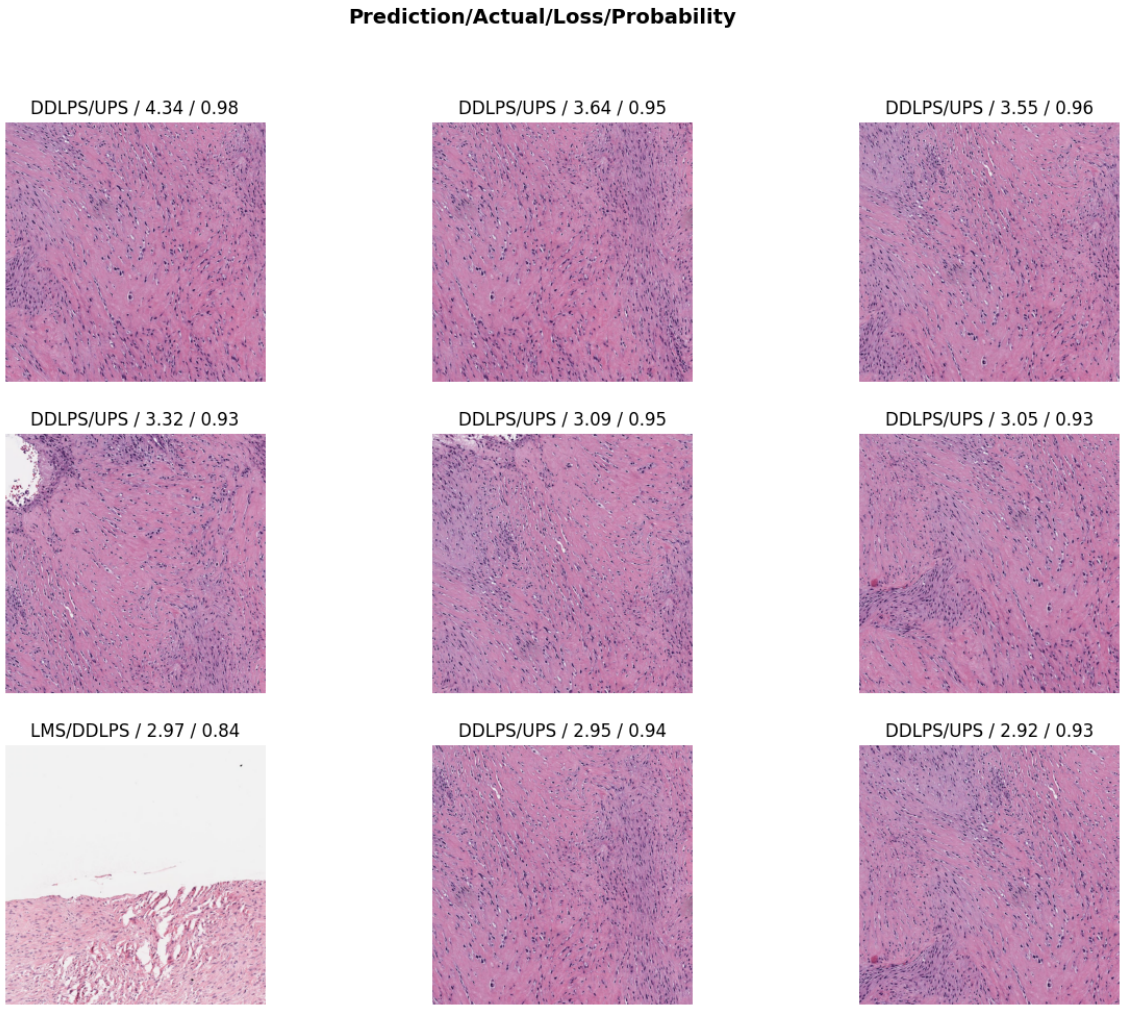


Figura 40: Experimento 4. `Fold 1`. Validación interna: *Tiles* con mayor pérdida

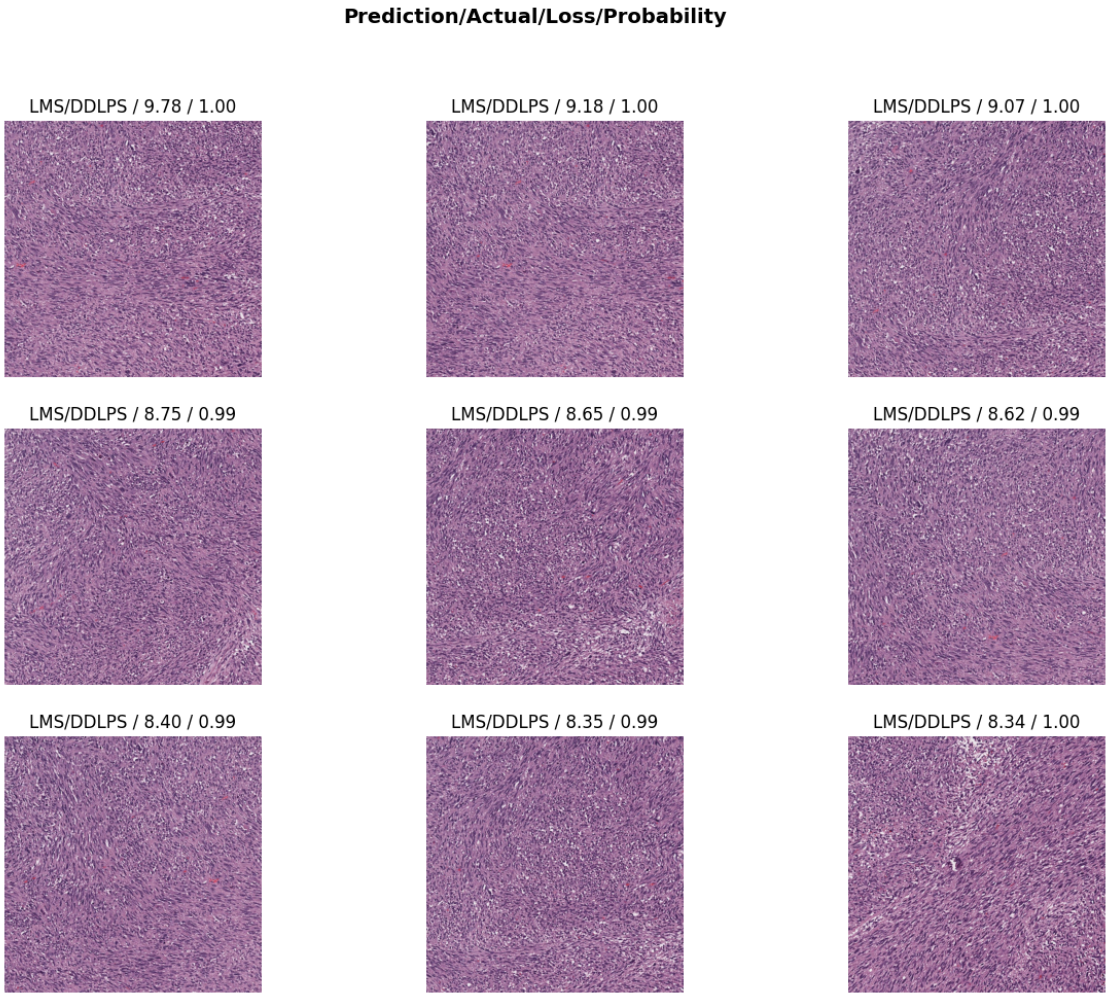


Figura 41: Experimento 4. `Fold 2`. Validación interna: *Tiles* con mayor pérdida

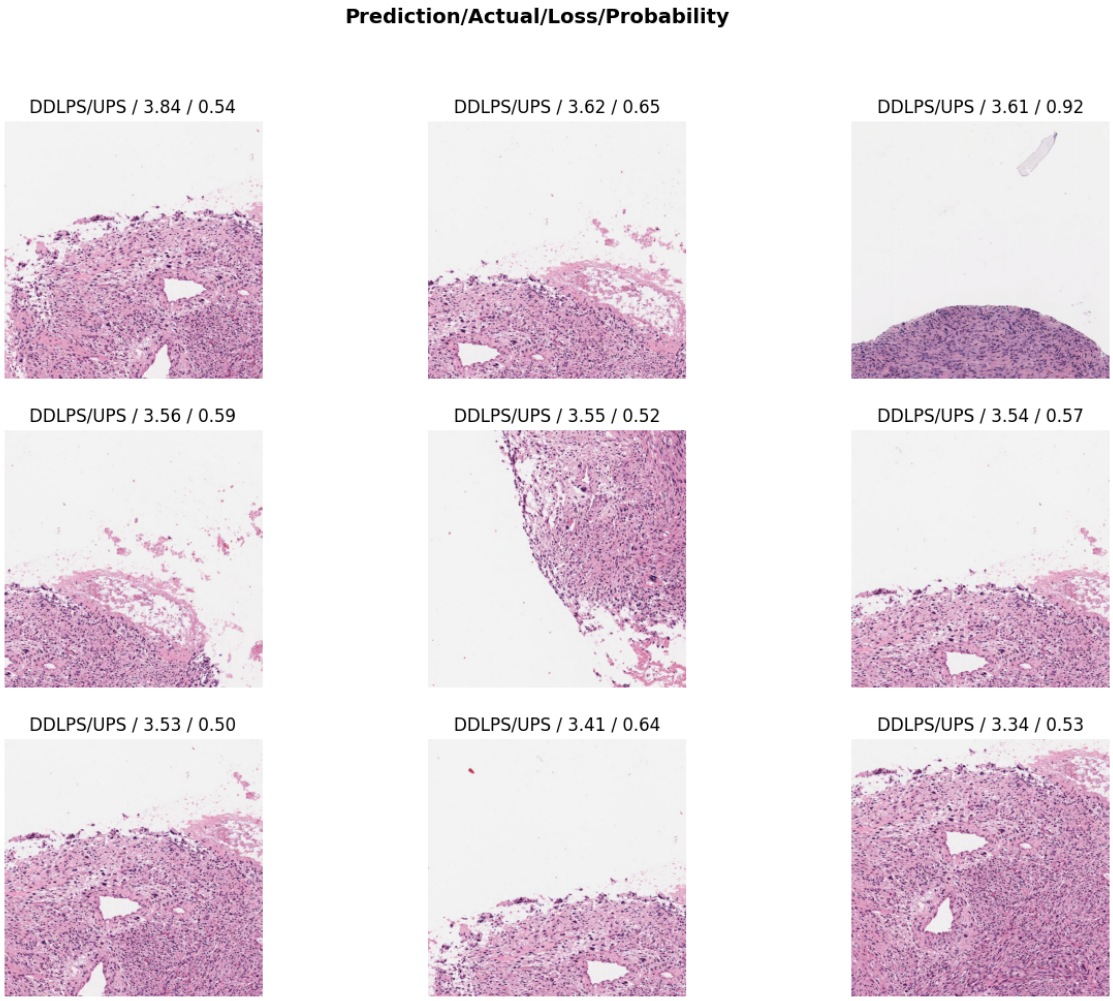


Figura 42: Experimento 4. `Fold 3`. Validación interna: *Tiles* con mayor pérdida

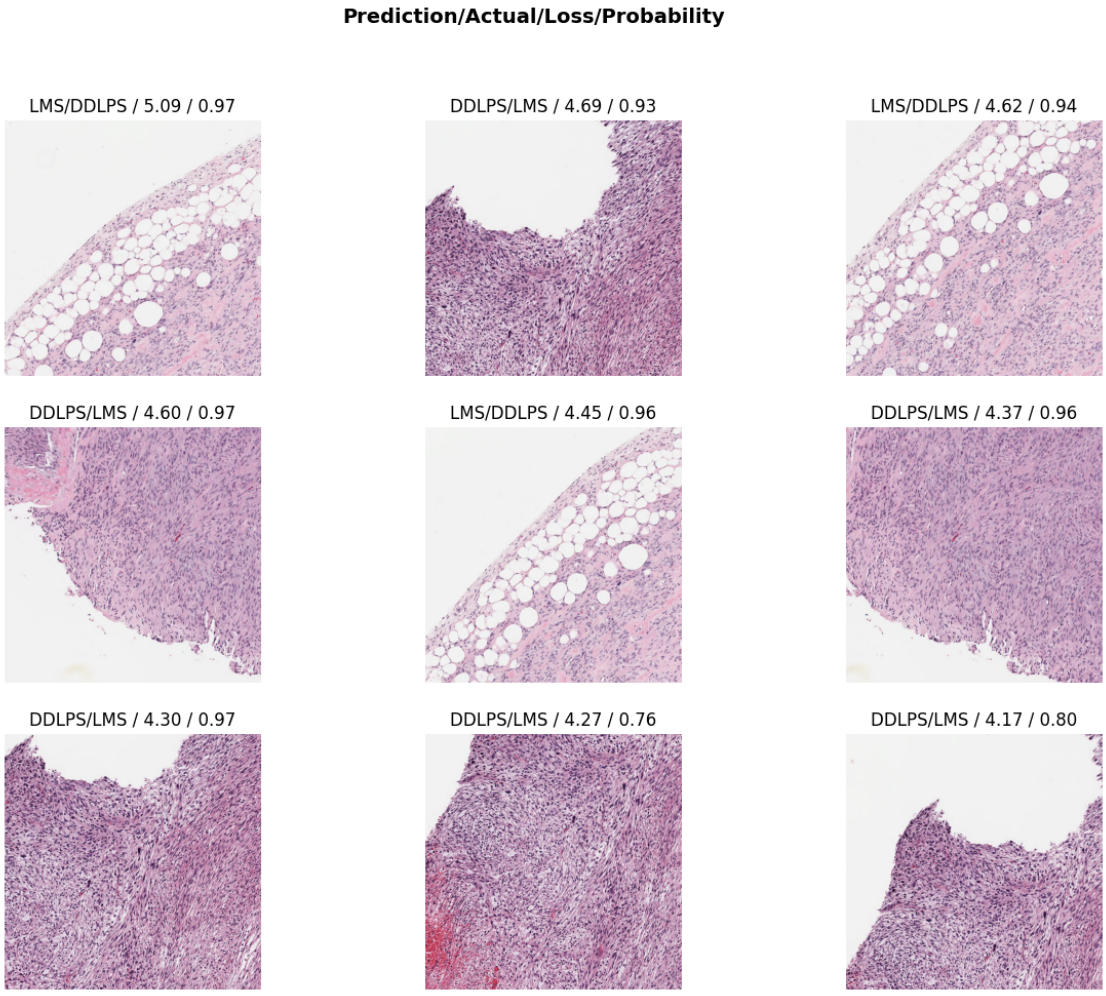


Figura 43: Experimento 4. `Fold 4`. Validación interna: *Tiles* con mayor pérdida

3.6.2. Evaluación externa

Se probó el ensamblado de los cinco modelos entrenados en este experimento para realizar predicciones tanto en el `TestSet-1` como en el `TestSet-2`.

Sobre el `TestSet-1` los resultados fueron significativamente malos, como se observa en la Figura 44, acertando solo una predicción, correspondiente a DDLPS. Además, se observa que los 2 casos de UPS se confunden con DDLPS, lo que sugiere un sesgo hacia este subtipo. Por otra parte, las puntuaciones de confianza son menos coherentes que en el Experimento 2, obteniendo una confianza del 62 % en la clasificación errónea del paciente `TCGA-FX-A3NJ`, como se observa en la Tabla 3.35.

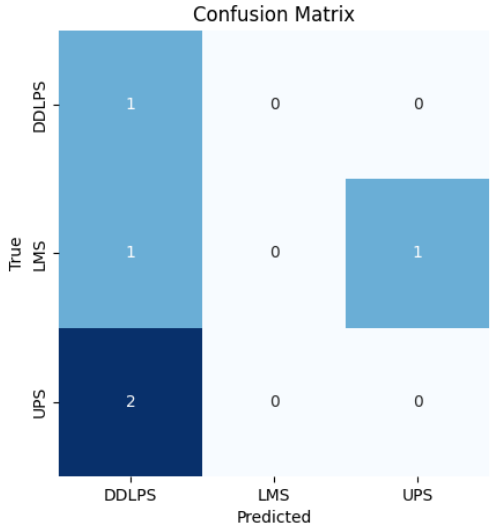


Figura 44: Experimento 4. Evaluación externa sobre TestSet-1: Matriz de confusión a nivel de paciente

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.67	36.59
TCGA-FX-A3NJ	64	DDLPS	LMS	0.62	74.05
TCGA-QQ-A5V9	233	DDLPS	UPS	0.49	153.67
TCGA-QQ-A5VB	289	UPS	LMS	0.51	91.88
TCGA-VT-A80J (DX2)	154	DDLPS	UPS	0.54	84.81

Tabla 3.35: Experimento 4: Predicción por paciente en el conjunto de evaluación externa TestSet-1

Cabe destacar que las desviaciones estándar son las más bajas de todos los experimentos, como se observa en la Tabla 3.36 y la Figura 45. El problema es que esto ocurre con las clasificaciones erróneas, no solo con el acierto, mostrando un consenso hacia la predicción errónea.

Paciente	DDLPS	LMS	UPS	Std_DDLPS	Std_LMS	Std_UPS
TCGA-DX-A2J0	0.67	0.29	0.04	0.09	0.04	0.06
TCGA-FX-A3NJ	0.62	0.28	0.10	0.06	0.06	0.05
TCGA-QQ-A5V9	0.49	0.04	0.47	0.10	0.03	0.10
TCGA-QQ-A5VB	0.32	0.18	0.51	0.10	0.07	0.13
TCGA-VT-A80J (DX2)	0.54	0.24	0.23	0.06	0.05	0.08

Tabla 3.36: Experimento 4. Evaluación externa sobre TestSet-1: Distribución de probabilidades y desviación estándar por clase para cada paciente

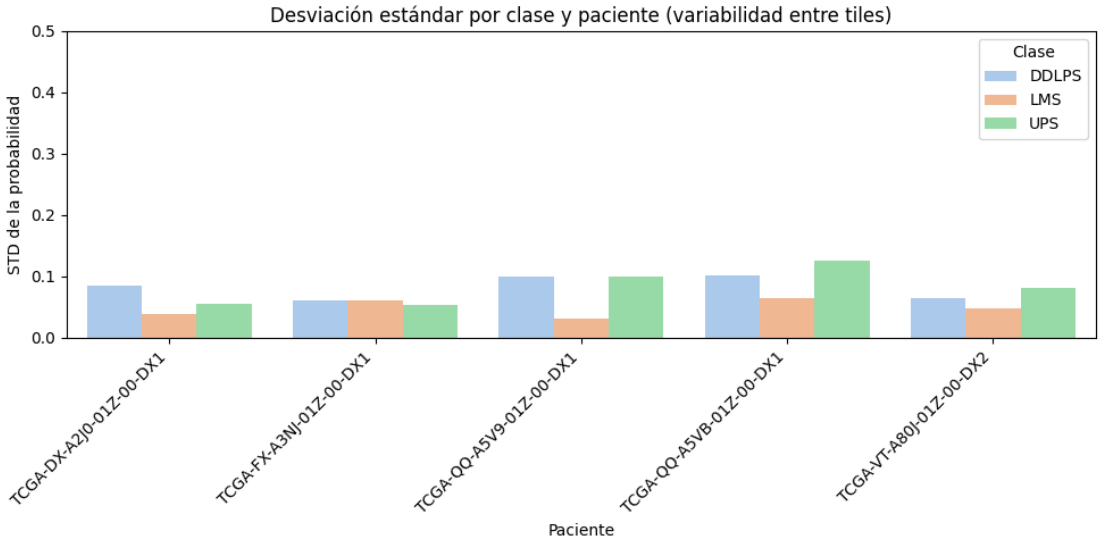


Figura 45: Experimento 4. Evaluación externa sobre TestSet-1: Desviación estándar por clase y paciente

Los resultados en `TestSet-2` tampoco fueron buenos, obteniendo cero clasificaciones correctas, como se muestra en la Figura 46, aunque con puntuaciones de confianza algo más bajas, excepto la correspondiente al paciente `TCGA-DX-A48U`, como se observa en la Tabla 3.37. Además, se observa el mismo comportamiento de sesgo hacia DDLPS.

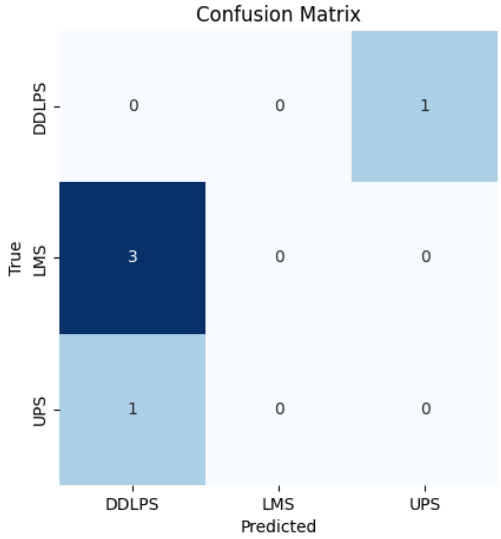


Figura 46: Experimento 4. Evaluación externa sobre TestSet-2: Matriz de confusión a nivel de paciente

Paciente	**Nº tiles**	**Predicción**	**Real**	**Confianza**	**Tiempo (s)**
TCGA-DX-A2IZ	215	UPS	DDLPS	0.44	56.73
TCGA-DX-A48U	61	DDLPS	LMS	0.59	15.04
TCGA-DX-A6BA (DX1)	417	DDLPS	LMS	0.49	108.05
TCGA-DX-A6BA (DX2)	145	DDLPS	LMS	0.44	35.71
TCGA-MB-A5Y8	3	DDLPS	UPS	0.53	0.71

Tabla 3.37: Experimento 4: Predicción por paciente en el conjunto de evaluación externa TestSet-2

Por otra parte, como se observa en la Tabla 3.38 y la Figura 47, aunque todas las predicciones son erróneas, se observan desviaciones estándar bajas, lo que indica que hay poca variabilidad entre *tiles*. Esto sugiere un aprendizaje de patrones no discriminativos.

Paciente	**DDLPS**	**LMS**	**UPS**	**Std_DDLPS**	**Std_LMS**	**Std_UPS**
TCGA-DX-A2IZ	0.4	0.16	0.44	0.07	0.11	0.12
TCGA-DX-A48U	0.59	0.38	0.03	0.04	0.04	0.02
TCGA-DX-A6BA (DX1)	0.49	0.27	0.24	0.07	0.09	0.08
TCGA-DX-A6BA (DX2)	0.44	0.37	0.19	0.13	0.09	0.11
TCGA-MB-A5Y8	0.53	0.32	0.15	0.08	0.04	0.04

Tabla 3.38: Experimento 4. Evaluación externa sobre TestSet-2: Distribución de probabilidades y desviación estándar por clase para cada paciente

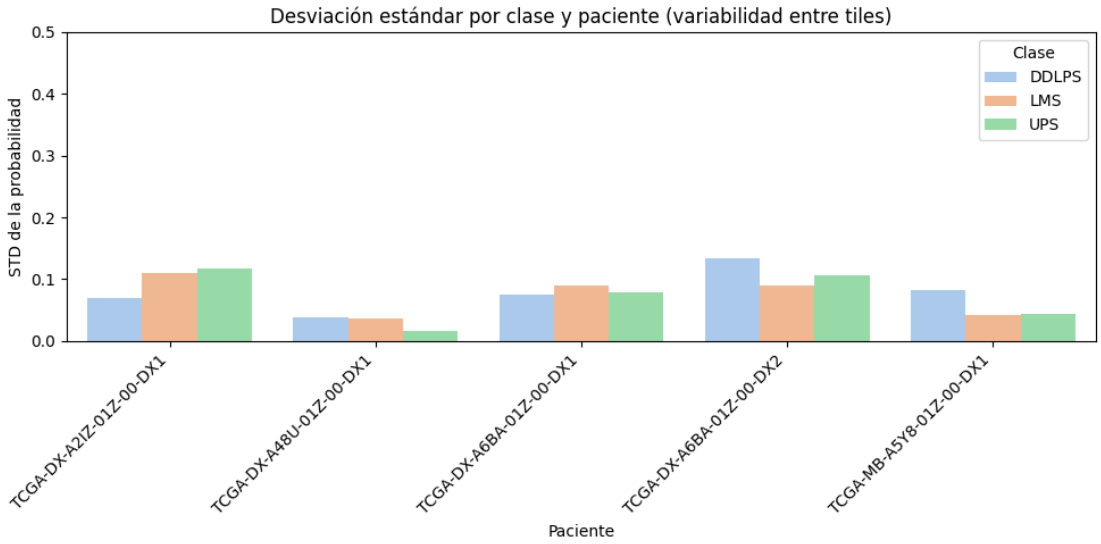


Figura 47: Experimento 4. Evaluación externa sobre TestSet-2: Desviación estándar por clase y paciente

3.7. Quinto experimento: pruebas de ensamble de modelos

En vista de los resultados obtenidos, se decidió construir un nuevo ensamble de modelos utilizando los *folds* 0 y 3 de este Experimento 4, que mejoraban a los correspondientes del Experimento 2, y los *folds* 1, 2 y 4 del Experimento 2.

Métricas de validación interna

Se calculó la matriz de confusión por paciente sobre el conjunto de validación concatenando los resultados de las predicciones de cada modelo. Como se observa en la Figura 48, se obtuvieron los mejores resultados hasta el momento, con 26 predicciones de paciente acertadas. En el caso de DDLPS, se cometió un único error, confundido con LMS. Por otro lado, 9 de los 11 pacientes de LMS fueron clasificados correctamente, con 1 error hacia DDLPS y otro hacia UPS. UPS es el que presentó más problemas, con 7 aciertos y 4 errores, 3 de ellos hacia DDLPS.

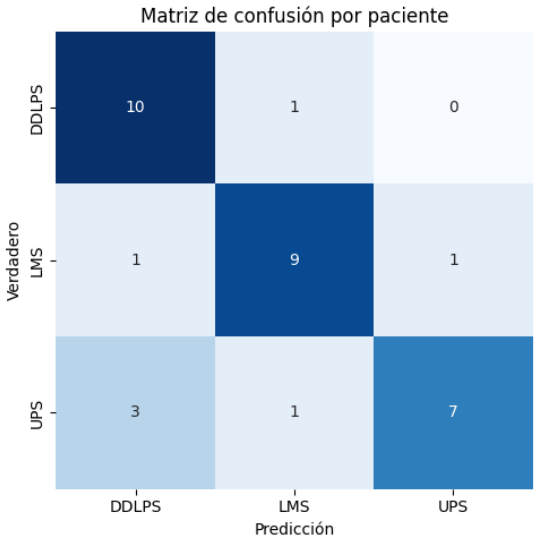


Figura 48: Experimento 5. Validación interna: Matriz de confusión a nivel de paciente

Evaluación externa

Sobre el `TestSet-1` se obtuvieron muy buenos resultados, igualando la tasa de aciertos del primer experimento, con una sola predicción errónea. No obstante, el error se distribuye de manera diferente. Mientras que en el segundo experimento se confundía el caso de UPS hacia LMS, en este experimento la confusión va de LMS hacia DDLPS, como se observa en la Figura 49.

Por otro lado, en la Tabla 3.39 se observa que la puntuación de confianza más baja (0,42) corresponde al paciente clasificado erróneamente, `TCGA-FX-A3NJ`, aunque presenta una menor diferencia con respecto a la segunda puntuación más baja (0,43) comparado con el modelo del segundo experimento, que fue de 0,46 para la predicción errónea, seguido de 0,53, correspondiente a `TCGA-FX-A3NJ`. Esto puede indicar un problema específico para clasificar correctamente este paciente.

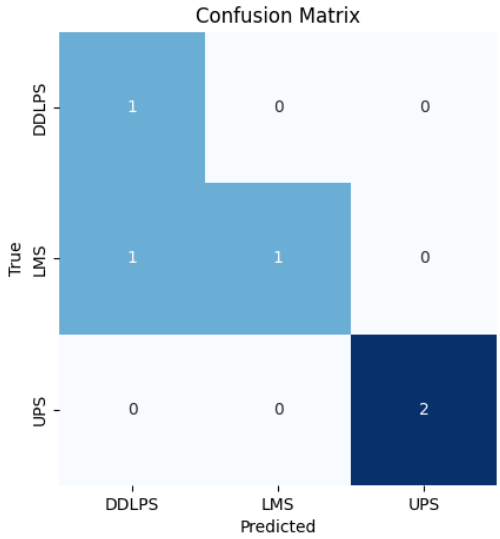


Figura 49: Experimento 5. Evaluación externa sobre TestSet-1: Matriz de confusión a nivel de paciente

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.57	38.72
TCGA-FX-A3NJ	64	DDLPS	LMS	0.42	74.28
TCGA-QQ-A5V9	233	UPS	UPS	0.74	153.21
TCGA-QQ-A5VB	289	LMS	LMS	0.50	112.49
TCGA-VT-A80J (DX2)	154	UPS	UPS	0.43	74.73

Tabla 3.39: Experimento 5: Predicción por paciente en el conjunto de evaluación externa TestSet-1

En los casos correctamente clasificados, las puntuaciones de confianza son razonables y las desviaciones son bajas, como se observa en la Tabla 3.40. En el caso clasificado erróneamente, `TCGA-FX-A3NJ` hay una clara falta de certeza, ya que todas las clases tienen probabilidades similares. Además, las desviaciones se mantienen bajas, indicando un consenso erróneo entre *tiles*.

Paciente	DDLPS	LMS	UPS	Std_DDLPS	Std_LMS	Std_UPS
TCGA-DX-A2J0	0.57	0.32	0.11	0.12	0.03	0.10
TCGA-FX-A3NJ	0.42	0.32	0.26	0.06	0.10	0.11
TCGA-QQ-A5V9	0.25	0.02	0.74	0.09	0.02	0.09
TCGA-QQ-A5VB	0.17	0.50	0.33	0.05	0.06	0.08
TCGA-VT-A80J (DX2)	0.35	0.22	0.43	0.07	0.09	0.10

Tabla 3.40: Distribución de probabilidades y desviación estándar por clase para cada paciente

En el TestSet-2 los resultados vuelven a ser peores, como se observa en la Figura 50, acertando solo 2 de las 5 predicciones. Este conjunto parece presentar dificultades concretas, seguramente debido a las características histológicas de los pacientes que lo conforman. El caso complejo, `TCGA-DX-A6BA`, es correctamente clasificado en una de las WSI, DX2, que es la que menos *tiles* tiene, por lo que una conclusión puede ser que los errores cometidos están relacionados con las dificultades histológicas.

Por otra parte, como se muestra en la Tabla 3.41, las puntuaciones de confianza son bajas, excepto para el paciente `TCGA-DX-A2IZ` (0,64). No obstante, se consideran más informativas las puntuaciones del segundo experimento.

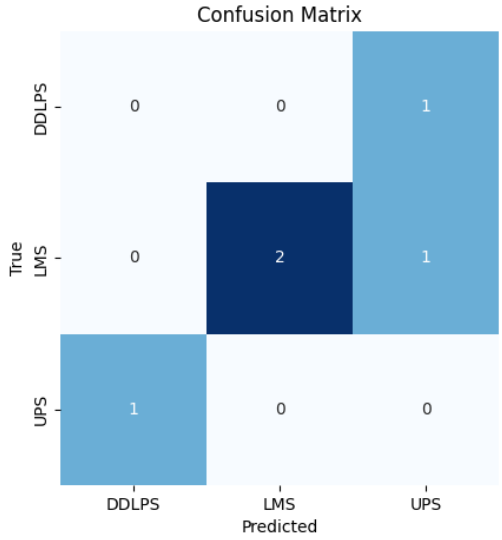


Figura 50: Experimento 5. Evaluación externa sobre TestSet-2: Matriz de confusión a nivel de paciente

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2IZ	215	UPS	DDLPS	0.64	114.95
TCGA-DX-A48U	61	LMS	LMS	0.51	43.63
TCGA-DX-A6BA (DX1)	417	UPS	LMS	0.44	228.64
TCGA-DX-A6BA (DX2)	145	LMS	LMS	0.47	79.92
TCGA-MB-A5Y8	3	DDLPS	UPS	0.38	2.34

Tabla 3.41: Experimento 5: Predicción por paciente en el conjunto de evaluación externa TestSet-2

Como se muestra en la Tabla 3.42, el paciente clasificado erróneamente `TCGA-DX-A2IZ` presenta una puntuación de confianza alta para el subtipo erróneo, además de desviaciones estándar relativamente bajas, lo que indica sobreconfianza del modelo. Por otro lado, en el paciente `TCGA-MB-A5Y8` se observan puntuaciones de confianza muy similares en todas las clases y bajas desviaciones típicas, lo que puede indicar el aprendizaje de patrones poco discriminativos. En todos los experimentos, TestSet-2 —conformado por casos más complejos que TestSet-1— ha evidenciado la falta de generalización de todos los modelos entrenados. Esto tiene mucho que ver con la heterogeneidad de los sarcomas, tanto intra-paciente como inter-paciente, lo que hace difícil entrenar un modelo consistente con pocos datos.

Paciente	**DDLPS**	**LMS**	**UPS**	**Std DDLPS**	**Std LMS**	**Std UPS**
TCGA-DX-A2IZ	0.19	0.18	0.64	0.19	0.20	0.23
TCGA-DX-A48U	0.41	0.51	0.08	0.05	0.06	0.05
TCGA-DX-A6BA (DX1)	0.34	0.22	0.44	0.07	0.08	0.08
TCGA-DX-A6BA (DX2)	0.33	0.47	0.20	0.07	0.13	0.10
TCGA-MB-A5Y8	0.38	0.31	0.31	0.08	0.05	0.03

Tabla 3.42: Experimento 5. Validación externa sobre TestSet-2: Distribución de probabilidades y desviación estándar por clase para cada paciente

Se probaron otras combinaciones de *folds* para formar otros ensambles de modelos, pero se obtuvieron peores resultados, por lo que se escogió este ensamble como modelo final.

3.7.1. Mejoras en calibración de confianza

Los resultados obtenidos con este ensamble habían sido los mejores hasta ahora. No obstante, en ocasiones se observaba una confianza excesiva en predicciones erróneas y baja en aciertos. Debido a las limitaciones de tiempo, en vez de reentrenar, se decidió mejorar el comportamiento del modelo en términos de calibración de la confianza con el objetivo de mejorar la confianza y estabilidad de las predicciones a nivel de paciente. De esta forma, se podría establecer un umbral de confianza a partir del cual los resultados del modelo pueden considerarse fiables.

Calibración adaptativa por paciente

En lugar de aplicar *soft voting*, que simplemente hace el promedio entre las probabilidades de cada modelo y luego entre *tiles*, se optó por implementar ponderación dinámica basada en el comportamiento de cada modelo sobre cada paciente. Se optó por este enfoque porque durante el análisis de resultados se observó que no todos los modelos del ensamble eran igual de fiables para todos los casos, y que ciertos modelos generalizaban mejor a unos subtipos que a otros.

Para calcular el peso adaptativo del modelo, se decidió tener en cuenta los siguientes criterios:

- Confianza media. Favorecer modelos que se muestran más seguros.
- Varianza intra-modelo. Penalizar modelos cuyas predicciones varían mucho entre *tiles* del mismo paciente.
- Discrepancia con otros modelos. Penalizar modelos que se desvían mucho del consenso del resto de modelos en cuanto a distribución de probabilidades.

Al realizar inferencia en TestSet-1, se observó que persistían las predicciones erróneas con confianza excesiva. De hecho, las puntuaciones de confianza eran mayores que en el Experimento 2, como se observa en la Tabla 3.43, lo que sugería un problema de sobreconfianza estructural en las salidas de los modelos. Esto realmente ya se había anticipado durante el análisis de los resultados de los experimentos, donde se observó que algunos modelos del ensamble presentaban una alta seguridad al hacer las predicciones erróneas. Además, como se muestra en la Tabla 3.44, las desviaciones estándar se mantenían bajas.

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.64	8.24
TCGA-FX-A3NJ	64	DDLPS	LMS	0.54	16.23
TCGA-QQ-A5V9	233	UPS	UPS	0.86	59.46
TCGA-QQ-A5VB	289	LMS	LMS	0.61	71.49
TCGA-VT-A80J (DX2)	154	UPS	UPS	0.40	39.06

Tabla 3.43: Experimento 5. Ponderación adaptativa: Predicción por paciente en el conjunto de evaluación externa TestSet-1

Paciente	DDLPS	LMS	UPS	Std_DDLPS	Std_LMS	Std_UPS
TCGA-DX-A2J0	0.64	0.27	0.09	0.09	0.02	0.09
TCGA-FX-A3NJ	0.54	0.27	0.19	0.05	0.07	0.08
TCGA-QQ-A5V9	0.12	0.01	0.86	0.06	0.02	0.07
TCGA-QQ-A5VB	0.13	0.61	0.26	0.04	0.05	0.07
TCGA-VT-A80J (DX2)	0.38	0.23	0.40	0.06	0.07	0.09

Tabla 3.44: Experimento 5. Ponderación adaptativa. Evaluación externa sobre TestSet-2: Distribución de probabilidades y desviación estándar por clase para cada paciente

Mejora de calibración adaptativa por paciente

La penalización de los modelos discrepantes se hacía teniendo en cuenta únicamente la distribución de probabilidades, sin tener en cuenta si la clase predicha era diferente o no. Esto llevó a aplicar un criterio más: no solo penalizar el desacuerdo en probabilidades, sino también si el modelo predice una clase distinta a la del consenso.

Como se observa en la Tabla 3.45, las puntuaciones de confianza mejoraron en términos de coherencia en las predicciones acertadas. Además, como se muestra en la Tabla 3.46, se observa una diferencia más amplia en las puntuaciones de confianza de las distintas clases, mostrando menos ambigüedad. No obstante, en la predicción errónea del paciente `TCGA-FX-A3NJ`, se obtuvo una puntuación de confianza mayor. Esto sugería un error estructural de los modelos, que presentaban dificultades especiales en clasificar este caso.

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.76	7.68
TCGA-FX-A3NJ	64	DDLPS	LMS	0.68	16.20
TCGA-QQ-A5V9	233	UPS	UPS	0.94	59.70
TCGA-QQ-A5VB	289	LMS	LMS	0.84	71.27
TCGA-VT-A80J (DX2)	154	UPS	UPS	0.58	39.10

Tabla 3.45: Experimento 5. Ponderación adaptativa v2: Predicción por paciente en el conjunto de evaluación externa TestSet-1

Paciente	DDLPS	LMS	UPS	Std_DDLPS	Std_LMS	Std_UPS
TCGA-DX-A2J0	0.76	0.17	0.07	0.11	0.04	0.08
TCGA-FX-A3NJ	0.68	0.20	0.12	0.06	0.05	0.06
TCGA-QQ-A5V9	0.05	0.01	0.94	0.05	0.03	0.06
TCGA-QQ-A5VB	0.03	0.84	0.13	0.01	0.07	0.07
TCGA-VT-A80J (DX2)	0.23	0.19	0.58	0.06	0.16	0.16

Tabla 3.46: Experimento 5: Distribución de probabilidades y desviación estándar por clase para cada paciente

Ponderación adaptativa y escalado de temperaturas

Ante la persistencia del problema de sobreconfianza, se probó una técnica de calibración conocida como escalado de temperatura (*temperature scaling*) con el objetivo de corregir esta sobreconfianza estructural. Esta estrategia ajusta las probabilidades predichas a partir de los logits originales, sin afectar a las clases predichas. Consiste en aplicar la transformación $\texttt{logits}/T$, donde si T>1 se suavizan las predicciones (se reduce la confianza) y T<1 las agudiza.

Para cada modelo, se recogieron sus logits sobre su respectivo conjunto de validación (*out-of-fold*). A continuación, se calculó la constante T que minimizara la entropía cruzada entre los logits ajustados y las etiquetas reales para cada modelo. Por último, se aplicó *softmax* a logits / T. En consecuencia, se suavizaron las probabilidades, ya que el valor de T calculado para cada modelo estaba en torno a 1,8.

Para su implementación en la inferencia del TestSet-1, se decidió combinar las dos estrategias, tanto ponderación adaptativa como escalado de temperaturas. Como se muestra en la Tabla 3.47, se redujo la sobreconfianza en las predicciones. Aunque esta estrategia no corregía el hecho de que algunos modelos eran menos fiables para algunos pacientes, se mejoró la coherencia entre las puntuaciones de la confianza y la certidumbre del modelo en la predicción. Al aplicar escalado de temperaturas, el modelo redistribuye la probabilidad entre las clases para reflejar mejor su incertidumbre, como se muestra en la Tabla 3.48. Esto puede hacer que aumente la puntuación de subtipos incorrectos, no porque el modelo esté más confundido, sino porque deja de mostrar una seguridad excesiva cuando en realidad no la tiene.

Paciente	Nº tiles	Predicción	Real	Confianza	Tiempo (s)
TCGA-DX-A2J0	31	DDLPS	DDLPS	0.61	24.76
TCGA-FX-A3NJ	64	DDLPS	LMS	0.53	48.62
TCGA-QQ-A5V9	233	UPS	UPS	0.87	162.51
TCGA-QQ-A5VB	289	LMS	LMS	0.76	122.90
TCGA-VT-A80J (DX2)	154	UPS	UPS	0.51	82.03

Tabla 3.47: Experimento 5. Ponderación adaptativa y escalado de temperaturas: Predicción por paciente en el conjunto de evaluación externa TestSet-1

Paciente	DDLPS	LMS	UPS	Std_DDLPS	Std_LMS	Std_UPS
TCGA-DX-A2J0	0.61	0.25	0.14	0.09	0.03	0.08
TCGA-FX-A3NJ	0.53	0.27	0.2	0.05	0.04	0.06
TCGA-QQ-A5V9	0.10	0.03	0.87	0.05	0.05	0.08
TCGA-QQ-A5VB	0.04	0.76	0.20	0.02	0.07	0.07
TCGA-VT-A80J (DX2)	0.26	0.24	0.51	0.05	0.13	0.12

Tabla 3.48: Experimento 5. TestSet-1. Ponderación adaptativa y escalado de temperaturas: Distribución de probabilidades y desviación estándar por clase para cada paciente

3.8. Prototipo de aplicación

Como objetivo final de este trabajo, se planteó el desarrollo de una aplicación para que pudiera ser utilizada por patólogos como herramienta de apoyo. Para ello, se implementó un prototipo de aplicación en Google Colab utilizando la biblioteca `Gradio`. Esta decisión responde a varios motivos:

- Las imágenes histológicas empleadas (formato `.svs`) son muy pesadas y requieren un preprocesamiento específico para convertilas en *tiles* en formato `.jpg`.
- Los patólogos pueden no estar familiarizados con entornos de desarrollo o instalación de dependencias.
- Google Colab permite ejecutar el modelo en GPU de forma gratuita, sin necesidad de infraestructura local.
- La interfaz generada con `Gradio` permite cargar imágenes, realizar la inferencia y visualizar los resultados de forma sencilla, sin necesidad de conocimientos técnicos previos.

Como modelo final se eligió el ensamble de este quinto experimento, concretamente la versión final que integra ponderación adaptativa y escalado de temperaturas. Las versiones de los paquetes principales son las siguientes:

- python 3.11
- fastai 2.7.19
- matplotlib 3.10.0
- pandas 2.2.2
- torch 2.6.0
- torchvision 0.21.0
- Pillow 11.2.1
- sklearn 1.6.1

El cuaderno `Jupyter Notebook` donde se implementa la aplicación está dividido en tres secciones, de forma que integre todas las fases necesarias:

1. **Configuración previa**. Se monta la unidad de Google Drive para acceder a las carpetas que contienen los modelos y los archivos `.svs`. Además, se instalan todas las dependencias necesarias y se fijan los parámetros generales, como `batch_size`, estableciendo la GPU como dispositivo por defecto.
2. **Definición de funciones**. Bloque que incluye las funciones de generación de *tiles* a partir de los archivos `.svs` y de inferencia usando ponderación adaptativa y escalado de temperaturas
3. **Aplicación**. El código correspondiente a la aplicación `Gradio`, que llama a las funciones necesarias para realizar la inferencia y muestra los resultados.

De esta forma, el patólogo únicamente tiene que asegurarse de tener el directorio de carpetas esperado: una carpeta raíz y dentro la carpeta de los modelos y la de las imágenes `.svs`. Para ejecutar la aplicación solo tendrá que darle al botón «Conectar GPU» de Google Colab y, después, «Ejecutar todas las celdas».

Finalmente, el usuario ve la interfaz de la aplicación que se muestra en la Figura 51, que permite seleccionar la acción deseada (generación de *tiles* o inferencia) y muestra progresivamente los resultados, incluyendo una galería de imágenes y la exportación de los resultados en formato `.csv`.

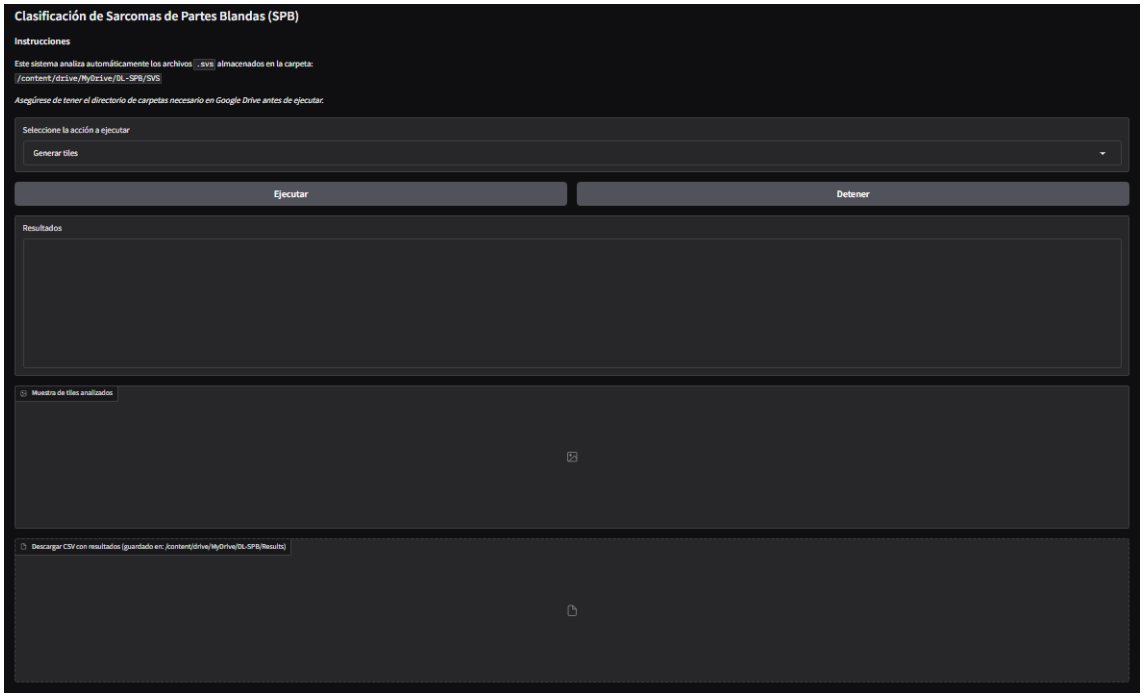


Figura 51: Interfaz de la aplicación

En la Figura 52 y la Figura 53 se muestra un ejemplo de inferencia. Como se observa, se muestran las puntuaciones de confianza para cada subtipo. Se tomó esta decisión porque, como se ha visto anteriormente, el modelo comete errores, y es importante que la información sea lo más transparente y completa posible. Mostrar los porcentajes de los tres subtipos permite que el patólogo interprete con mayor contexto el nivel de certeza del modelo. Por ejemplo, si dos subtipos presentan puntuaciones similares, puede ser indicativo de ambigüedad en el diagnóstico.

Por otra parte, además de las predicciones finales por paciente, se exportan también las predicciones por clase y por modelo, así como los indicadores de desacuerdo entre modelos, lo que permite un análisis más detallado y transparente del comportamiento del modelo.

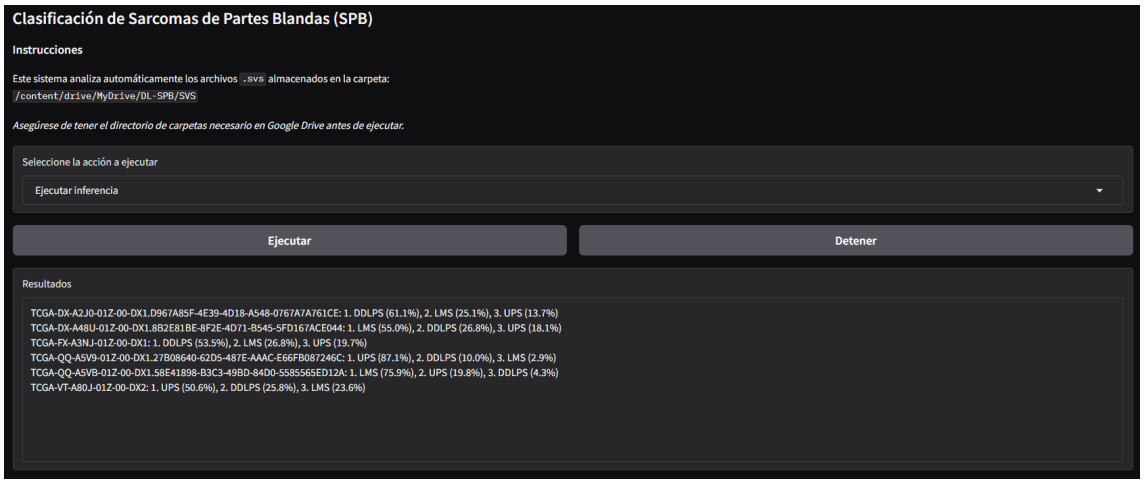


Figura 52: Interfaz de la aplicación. Ejemplo de inferencia: Resultados de predicción

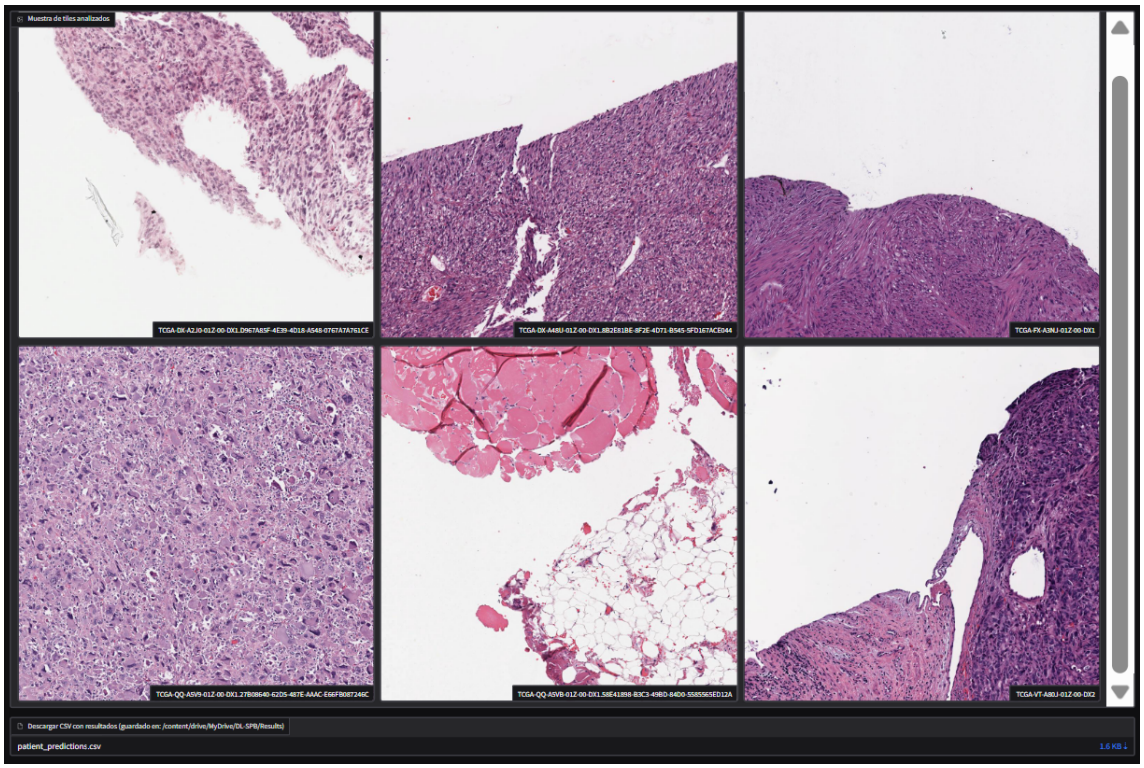


Figura 53: Interfaz de la aplicación. Ejemplo de inferencia: Galería de imágenes de las muestras analizadas

Capítulo 4

Presupuesto

4.1. Presupuesto real asumido

Para la realización de este proyecto, el único coste ha sido la suscripción a `Google Colab Pro`, con un importe mensual de 11,19 € durante un total de 4 meses, lo que supuso un gasto final de 44,76 €.

4.2. Presupuesto estimado en un entorno profesional

En un entorno profesional, el presupuesto estimado es de 13.758,34 €, como se detalla en la Tabla 4.1. Los principales costes corresponden a la contratación del personal necesario: un ingeniero con experiencia en aprendizaje profundo y un patólogo especializado en sarcomas para la selección de muestras y la interpretación histopatológica de los resultados. El coste laboral se obtiene de la Encuesta Trimestral de Coste Laboral (ETCL) del Instituto Nacional de Estadística (INE) [38].

Además de la suscripción a `Google Colab Pro`, se recomienda un plan de pago de almacenamiento en `Google Drive` de 200 GB, lo que facilita y agiliza las tareas de preprocesamiento y organización del dataset.

Por último, dado que el desarrollo del proyecto se ha realizado en `Google Colab`, no se requiere un ordenador de altas prestaciones. Por ello, se recomienda un equipo de gama media como el portátil HP 250 G10, con procesador Intel Core i7-1355U, 16 GB de RAM y 512 GB de almacenamiento SSD.

Concepto	**Coste unitario**	**Unidades**	**Coste total**
Suscripción `Google Colab Pro`	11,19 €/mes	4 meses	44,76 €
Suscripción `Google Drive 200 GB`	2,99 €/mes	2 meses	5,98 €
Contratación ingeniero	34,93 €/hora	350 horas	12.225,50 €
Contratación patólogo	29,77 €/hora	30 horas	893,10 €
Portátil HP 250 G10	589,00 €	1 unidad	589,00 €
Total			**13.758,34 €**

Tabla 4.1: Presupuesto estimado del proyecto

Capítulo 5

Impacto del proyecto

Como se ha explicado a lo largo de este trabajo, los sarcomas de partes blandas constituyen un grupo heterogéneo de tumores poco frecuentes, caracterizados por su alta agresividad y complejidad diagnóstica. Esta complejidad se traduce en un alto porcentaje de diagnósticos erróneos, lo que puede afectar negativamente al pronóstico del paciente y condicionar las decisiones terapéuticas. Esto, sumado a la escasez de especialistas con experiencia específica en sarcomas, hace fundamental que los pacientes sean evaluados en un CSUR por un equipo multidisciplinar experto en esta patología.

En este contexto, la implementación propuesta no pretende sustituir el juicio del patólogo, sino ser una herramienta de apoyo en el diagnóstico histopatológico, especialmente en casos complejos, donde pueden surgir discrepancias incluso entre patólogos con experiencia en sarcomas. Además de servir de apoyo en la toma de decisiones clínicas, también puede ser útil como herramienta de investigación para identificar patrones morfológicos o características intermedias entre subtipos.

Además de implicaciones médicas, este proyecto presenta implicaciones tecnológicas y sociales derivadas del uso del aprendizaje profundo para diagnosticar una enfermedad poco frecuente y compleja como el sarcoma de partes blandas. Aunque existen múltiples aplicaciones del aprendizaje profundo en oncología, su uso en sarcomas es limitado. En este trabajo se ha actualizado y mejorado una implementación obsoleta, y se ha generado un dataset propio que podría servir como base para futuras versiones más amplias y refinadas. Por otra parte, el desarrollo de una aplicación en `Google Colab` mediante `Gradio` permite que el modelo sea accesible para profesionales médicos sin necesidad de conocimientos técnicos avanzados ni infraestructura local.

En relación con los Objetivos de Desarrollo Sostenible (ODS), este trabajo se relaciona principalmente con el número 3: salud y bienestar, y con el número 9: industria, innovación e infraestructuras, al contribuir al desarrollo de herramientas tecnológicas que podrían mejorar el diagnóstico de los sarcomas de partes blandas. Además, se relaciona de forma indirecta con el ODS número 10: reducción de las desigualdades. En primer lugar, puede contribuir a homogeneizar criterios diagnósticos entre profesionales y reducir tiempos de decisión en casos complejos. También puede ser una herramienta útil en formación médica, facilitando la formación especializada en sarcomas o el aprendizaje sobre subtipos histológicos poco frecuentes. Por otro lado, podría ser útil en países que no cuentan con redes especializadas como los CSUR existentes en España.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

A lo largo de este trabajo se ha abordado la complejidad del diagnóstico histopatológico del sarcoma de partes blandas, especialmente desafiante por su alta heterogeneidad y baja prevalencia. Con el objetivo de contribuir a la mejora de esta situación, se ha desarrollado un sistema basado en aprendizaje profundo para la clasificación de los tres subtipos más frecuentes a partir de imágenes histológicas digitalizadas.

La implementación final consiste en un ensamble de cinco modelos entrenados mediante validación cruzada estratificada para la clasificación de los subtipos leiomiosarcoma, liposarcoma desdiferenciado y sarcoma pleomórfico indiferenciado. Para el entrenamiento se ha generado un dataset compuesto por imágenes en formato JPG (*tiles*) extraídas de WSI correspondientes a 33 pacientes, obtenidas del dataset público TCGA-SARC.

Los distintos experimentos permitieron identificar algunas conclusiones importantes. En el primer experimento se empleó una única división del dataset en entrenamiento (90 %) y validación (10 %), estratificada por paciente. Se observó que el modelo presentaba signos de sobreajuste *overfitting*, debido principalmente al reducido tamaño del dataset y a la escasa representatividad de los pacientes incluidos en validación, lo cual es especialmente crítico en el contexto de los sarcomas, caracterizados por una amplia variabilidad. Estos fueron los principales desafíos del proyecto: la disponibilidad limitada del dataset y la amplia heterogeneidad de los sarcomas, tanto intrapaciente como interpaciente.

Frente a estos desafíos, la validación cruzada se presentó como una estrategia útil, obteniendo mejoras significativas en la exactitud y en la pérdida de validación en el conjunto de evaluación externa. Concretamente, la validación cruzada estratificada por paciente, ya que es fundamental mantener la agrupación por paciente para evitar la fuga de datos, es decir, que los *tiles* de un paciente estén repartidos en entrenamiento y validación, lo que llevaría al modelo a memorizar patrones específicos de cada caso en lugar de aprender representaciones generalizables.

Aunque se lograron buenos resultados con muchos de los pacientes evaluados, el modelo no fue capaz de clasificar correctamente todos los casos. Además, a pesar de la mejora en la coherencia entre la incertidumbre y la precisión del modelo, en algunos casos se observaba una alta variabilidad de predicciones a nivel de *tile*,

poniendo de manifiesto las limitaciones del dataset. Además de ser reducido, en la validación cruzada se entrena sobre particiones distintas, y es importante que estas estén equilibradas no solo por subtipo —como se ha hecho—, sino también por la complejidad histológica de cada caso. De hecho, se observó que había pacientes que los modelos de los distintos experimentos tendían a clasificar de forma errónea. Para ello, resulta imprescindible la colaboración de un patólogo especializado en sarcomas. Esta es una de las conclusiones más relevantes del proyecto: la implementación desarrollada podría tener valor no solo como herramienta de apoyo diagnóstico, sino también como herramienta de investigación en situaciones de alta incertidumbre o para estudiar patrones de confusión entre subtipos.

6.2. Trabajos futuros

El proyecto deja abiertas múltiples vías de mejora y exploración futura, tanto por las limitaciones identificadas, como por las ideas surgidas durante su desarrollo. Algunas de las más relevantes son las siguientes:

- Incorporar técnicas de visualización como *Class Activation Maps* (CAMs), que permitan localizar qué regiones del tile han contribuido a la predicción, facilitando así la interpretación del modelo.
- Refinar el conjunto de datos mediante segmentación de regiones tumorales con la ayuda de un patólogo, o empleando modelos de segmentación automática, con el objetivo de generar *tiles* más representativos.
- Mejorar la estrategia de partición de los pacientes en los distintos *folds*, incorporando el conocimiento de un patólogo para equilibrar la complejidad de los casos y evitar sesgos en la validación cruzada.
- Analizar con mayor profundidad los errores del modelo y los casos de mayor incertidumbre en colaboración con un patólogo experto, revisando los *tiles* con mayor pérdida o con clasificaciones dudosas.
- Explorar distintos umbrales de descarte de *tiles* con alto contenido de fondo blanco con la colaboración de un patólogo, ya que zonas con escaso tejido podrían contener información relevante.
- Evaluar el comportamiento del modelo en pacientes con múltiples WSI, analizando las diferencias entre las predicciones.
- Probar otras resoluciones de entrada (por ejemplo, 512×512 px) y explorar con mayor profundidad diferentes tamaños de *batch* y *learning rates*, así como aumentar el número de *epochs*, utilizando más recursos computacionales.
- Integrar datos genómicos disponibles en portales como cBioPortal, con el objetivo de extender las capacidades del modelo hacia tareas de predicción de pronóstico o respuesta a tratamientos.
- Investigar la correlación entre patrones morfológicos de imágenes histológicas y mutaciones específicas.
- Explorar el enfoque de *Multiple Instance Learning* (MIL), donde el modelo aprende a partir de grupos de *tiles* (o *bag*) con una única etiqueta global, en lugar de asignar etiquetas a cada *tile*, lo que puede ser más oportuno para aprender las características a nivel de tumor.

Bibliografía

[1] M. A. Salgado, S. Resano, C. Saavedra e I. Pérez-Muñoz, «Epidemiología y estudio de extensión de los sarcomas de partes blandas y de los huesos,» *Revisiones en Cancer*, vol. 32, págs. 9-16, 2018. dirección: https://www.revistarevisionesencancer.com/articles/H0116/show.

[2] Grupo Español de Investigación en Sarcomas (GEIS), *¿Qué son los sarcomas?* 2025. dirección: https://grupogeis.org/es/que-son-los-sarcomas.

[3] Sarcoma UK, *The Loneliest Cancer - Sarcoma UK 2019 Report*, 2019. dirección: https://sarcoma.org.uk/about-us/stay-connected/the-loneliest-cancer/.

[4] I. Ray-Coquard, M. Montesco, J. Coindre et al., «Sarcoma: concordance between initial diagnosis and centralized expert review in a population-based study within three European regions,» *Annals of Oncology*, vol. 23, n.º 9, págs. 2442-2449, 2012. DOI: 10.1093/annonc/mdr610. dirección: https://pmc.ncbi.nlm.nih.gov/articles/PMC3425368/.

[5] GEIS, *Los sarcomas de partes blandas - Grupo GEIS*. dirección: https://grupogeis.org/es/que-son-los-sarcomas/los-sarcomas-de-partes-blandas.

[6] S. Shakya, E. L. Banneyake, S. Cholekho, J. Singh y X. Zhou, «Soft tissue sarcoma: clinical recognition and approach to the loneliest cancer,» *Exploration of Musculoskeletal Diseases*, vol. 2, n.º 1, págs. 56-68, 2024. DOI: 10.37349/emd.2024.00034. dirección: https://www.explorationpub.com/Journals/emd/Article/100734.

[7] S. E. de Oncología Médica (SEOM), *Sarcomas partes blandas - SEOM: Sociedad Española de Oncología Médica © 2019*, 2022. dirección: https://www.seom.org/info-sobre-el-cancer/sarcomas-partes-blandas?showall=1%5C&showall=1.

[8] A. Gronchi, A. Basiru, A. Tos et al., «Soft tissue and visceral sarcomas: ESMO–EURACAN–GENTURIS Clinical Practice Guidelines for diagnosis, treatment and follow-up,» *Annals of Oncology*, vol. 32, 2021. DOI: 10.1016/j.annonc.2021.07.006.

[9] J. Martin-Broto, N. Hindi, J. Cruz et al., «Relevance of Reference Centers in Sarcoma Care and Quality Item Evaluation: Results from the Prospective Registry of the Spanish Group for Research in Sarcoma (GEIS),» *The Oncologist*, vol. 24, n.º 6, e338-e346, 2018, ISSN: 1083-7159. DOI: 10.1634/theoncologist.2018-0121. dirección: https://doi.org/10.1634/theoncologist.2018-0121.

[10] A. C. Gamboa, A. Gronchi y K. Cardona, «Soft-tissue sarcoma in adults: An update on the current state of histiotype-specific management in an era of personalized medicine,» *CA: A Cancer Journal for Clinicians*, vol. 70, n.º 3, págs. 200-229, 2020. DOI: https://doi.org/10.3322/caac.21605. dirección: https://acsjournals.onlinelibrary.wiley.com/doi/abs/10.3322/caac.21605.

[11] K. Lacuna, S. Bose, M. Ingham y G. Schwartz, «Therapeutic advances in leiomyosarcoma,» *Frontiers in Oncology*, vol. 13, pág. 1 149 106, 2023. DOI: 10.3389/fonc.2023.1149106.

[12] R. Zafar e Y. Wheeler, *Liposarcoma*, StatPearls, ed. Treasure Island, FL: StatPearls Publishing, 2025. dirección: https://pubmed.ncbi.nlm.nih.gov/30855853/.

[13] B. C. Widemann y A. Italiano, «Biology and Management of Undifferentiated Pleomorphic Sarcoma, Myxofibrosarcoma, and Malignant Peripheral Nerve Sheath Tumors: State of the Art and Perspectives,» *Journal of Clinical Oncology*, vol. 36, n.º 2, págs. 160-167, 2018. DOI: 10.1200/JCO.2017.75.3467. dirección: https://ascopubs.org/doi/abs/10.1200/JCO.2017.75.3467.

[14] R. J. Canter, S. Beal, D. Borys, S. R. Martinez, R. J. Bold y A. S. Robbins, «Interaction of Histologic Subtype and Histologic Grade in Predicting Survival for Soft-Tissue Sarcomas,» *Journal of the American College of Surgeons*, vol. 210, n.º 2, 2010, ISSN: 1072-7515. DOI: https://doi.org/10.1016/j.jamcollsurg.2009.10.007. dirección: https://www.sciencedirect.com/science/article/pii/S1072751509014896.

[15] J. Cruz, S. Navarro, M. Guerra et al., «Valor de la inmunohistoquímica en la tipificación de los sarcomas de partes blandas y su discordancia con el análisis histopatológico convencional: un estudio de casos procedentes del INO (Cuba),» *Revista Española de Patología*, vol. 38, n.º 3, págs. 181-192, 2005. dirección: http://www.patologia.es/volumen38/vol38-num3/38-3n04.htm.

[16] E. d. Álava, «Patología Molecular de los sarcomas,» *Oncología (Barcelona)*, vol. 28, págs. 22-38, 2005, ISSN: 0378-4835. dirección: http://scielo.isciii.es/scielo.php?script=sci_arttext%5C&pid=S0378-48352005000900003%5C&nrm=iso.

[17] GEIS, *CSUR - Grupo GEIS*. dirección: https://grupogeis.org/es/quienes-somos/centros-de-referencia.

[18] J. Á. Fernández, B. Gómez Pérez, S. Cantín, J. M. Asencio y V. Artigas, «Encuesta nacional sobre el tratamiento de los sarcomas en España,» *Cirugía Española*, vol. 100, n.º 4, págs. 193-201, 2022, ISSN: 0009-739X. DOI: https://doi.org/10.1016/j.ciresp.2021.05.009. dirección: https://www.sciencedirect.com/science/article/pii/S0009739X21002049.

[19] N. Geographic, *Alan Turing y el test que diseñó para saber si una máquina es inteligente*, 2023. dirección: https://www.nationalgeographic.es/ciencia/2023/03/alan-turing-test-inteligencia-artificial.

[20] M. Cobo Cano y L. Lloret Iglesias, *¿Qué sabemos de la inteligencia artificial y la medicina?* (¿Qué sabemos de?). Madrid: Los Libros de la Catarata / CSIC, 2023. dirección: https://www.csic.es/es/ciencia-y-sociedad/libros-de-divulgacion/coleccion-que-sabemos-de/inteligencia-artificial-y-medicina.

[21] IBM, *Tipos de algoritmos de aprendizaje automático*, 2023. dirección: https://www.ibm.com/es-es/think/topics/machine-learning-algorithms.

[22] S. Dridi, *Supervised Learning - A Systematic Literature Review*, Preprint en OSF, 2024. DOI: 10.31219/osf.io/qtmcs.

[23] A. An, M. S. Rahman, J. Zhou y J. J. Kang, «A Comprehensive Review on Machine Learning in Healthcare Industry: Classification, Restrictions, Opportunities and Challenges,» *Sensors*, vol. 23, pág. 4178, abr. de 2023. DOI: 10.3390/s23094178.

[24] INTEF - Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado, *Aula en abierto: Inteligencia Artificial*. dirección: https://formacion.intef.es/aulaenabierto/mod/book/tool/print/index.php?id=5077.

[25] InteractiveChaos, *Tutorial de Deep Learning*. dirección: https://interactivechaos.com/es/manual/tutorial-de-deep-learning/tutorial-de-deep-learning.

[26] Khan Academy, *Estructura y función de las neuronas*. dirección: https://es.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/overview-of-neuron-structure-and-function.

[27] Pseudo1ntellectual, *Perceptron Research from the 50's & 60's, clip*, 2008. dirección: https://www.youtube.com/watch?v=cNxadbrN_aI.

[28] J. Pereira. «Perceptrón Multicapa y Algoritmo de Retropropagación,» MQL5 Articles. (2021), dirección: https://www.mql5.com/es/articles/8908.

[29] MathWorks, *Introducción a Deep Learning*. dirección: https://la.mathworks.com/discovery/deep-learning.html.

[30] P. Sabeghi, K. K. Kinkar, G. d. R. Castaneda et al., «Artificial intelligence and machine learning applications for the imaging of bone and soft tissue tumors,» *Frontiers in Radiology*, vol. Volume 4 - 2024, 2024, ISSN: 2673-8740. DOI: 10.3389/fradi.2024.1332535. dirección: https://www.frontiersin.org/journals/radiology/articles/10.3389/fradi.2024.1332535.

[31] R. Xu, J. Tang, C. Li et al., «Deep Learning-based Artificial Intelligence for Assisting Diagnosis, Assessment and Treatment in Soft Tissue Sarcomas,» *Meta-Radiology*, vol. 2, pág. 100 069, 2024. DOI: 10.1016/j.metrad.2024.100069.

[32] T. Hagi, T. Nakamura, H. Yuasa et al., «Prediction of prognosis using artificial intelligence-based histopathological image analysis in patients with soft tissue sarcomas,» *Cancer Medicine*, vol. 13, 2024. DOI: 10.1002/cam4.7252.

[33] S. Foersch, M. Eckstein, D.-C. Wagner et al., «Deep learning for diagnosis and survival prediction in soft tissue sarcoma,» *Annals of Oncology*, vol. 32, n.º 9, págs. 1178-1187, 2021, ISSN: 0923-7534. DOI: https://doi.org/10.1016/j.annonc.2021.06.007. dirección: https://www.sciencedirect.com/science/article/pii/S092375342102055X.

[34] S. Foersch, *DeepLearningSarcoma*, Repositorio GitHub, 2021. dirección: https://github.com/AGFoersch/DeepLearningSarcoma.

[35] The Cancer Imaging Archive, *TCGA-SARC Collection - The Cancer Imaging Archive*. dirección: https://www.cancerimagingarchive.net/collection/tcga-sarc/.

[36] National Cancer Institute, *TCGA-SARC Project - Genomic Data Commons*. dirección: https://portal.gdc.cancer.gov/projects/TCGA-SARC.

[37] cBioPortal for Cancer Genomics, *TCGA-SARC, Cell 2017*. dirección: https://www.cbioportal.org/study/summary?id=sarc_tcga_pub.

[38] Instituto Nacional de Estadística (INE), *Encuesta Trimestral de Coste Laboral (ETCL). Coste laboral por grupo de ocupación (CNO-11)*, 2024. dirección: https://www.ine.es/jaxiT3/Datos.htm?t=6037#_tabs-tabla.