UNIVERSIDAD NACIONAL DE EDUCACIÓN A DISTANCIA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

Trabajo Fin de Máster en Ingeniería y Ciencia de Datos

Aprendizaje federado aplicado al diagnóstico de tumores mamarios en imágenes de ultrasonido

Miguel Buenestado Cortés

Dirigido por: José Manuel Cuadra Troncoso / Mariano Rincón Zamorano

Curso: 2021-2022: 1ª Convocatoria

Agradecimientos

A mis tutores del Trabajo Fin de Máster, José Manuel Cuadra Troncoso y Mariano Rincón Zamorano, por el tiempo y el apoyo que me han dedicado guiándome en este proyecto.

A la Universidad Nacional de Educación a Distancia, por haberme cedido, a través de los propios tutores, los recursos virtuales necesarios sin los que me habría sido mucho más difícil completar los objetivos y resultados alcanzados en tiempo y forma.

Resumen

En el campo de la medicina, el aprendizaje automático se erige como una herramienta poderosa y eficaz en la automatización de la tarea del diagnóstico por imagen, mediante la creación de modelos informáticos predictivos.

Para lograr resultados con un alto grado de fiabilidad, dichos modelos requieren de grandes volúmenes de datos para su aprendizaje, característica poco frecuente en contextos reales, pues organismos e instituciones médicas, individualmente, no suelen disponer de tales cantidades de información. Idealmente, ésta podría ser compartida y cedida mutuamente en busca de un fin común, pero esto resulta altamente improbable, dadas las regulaciones imperantes en la actualidad en torno a la propiedad intelectual y la privacidad de datos médicos de pacientes.

A fin de superar tales escollos, surge el paradigma del aprendizaje federado (*federated learning*).

En este Trabajo Fin de Master, aplicamos dicho paradigma en un caso de aplicación concreto (detección de tumores mamarios en imágenes de ultrasonidos), empleando tecnología y técnicas específicas de federación (la librería *TensorFlow Federated*; la agregación federada estándar, *Federated Average*, y variantes de la misma), con las que entrenamos modelos predictivos sobre diferentes combinaciones de *datasets*. Evaluaremos la idoneidad y eficacia de dichos modelos federados, tanto por sí mismos como en comparación con resultados análogos derivados de un aprendizaje clásico. Exponemos que el aprendizaje federado puede alcanzar, bajo ciertas configuraciones, cotas de eficacia similares a las de un aprendizaje clásico.

El presente documento se articula, primero, con la introducción de los conceptos clave tocantes al propio caso de estudio: el problema a resolver en cuestión (segmentación de imágenes de ultrasonido), los aspectos concretos del aprendizaje automático que han ayudado a resolverlo (modelo de aprendizaje profundo, una red neuronal convolutiva *U-Net*) y profundizaremos en los fundamentos del paradigma de aprendizaje federado.

Posteriormente, se detallará la metodología aplicada durante el desarrollo del proyecto y se expondrán los resultados derivados del mismo.

Finalmente, abordaremos las conclusiones de dichos resultados, así como las limitaciones y posibles mejoras en el ámbito del presente trabajo y del aprendizaje federado, en particular.

Abstract

Machine Learning has emerged as a powerful and efficient tool for automating tasks such as diagnostic medical imaging (also known as *Computed-Aided Diagnosis*, *CAD*) via predictive computing models creation.

In order to reach high peaks of reliability, those intelligent creations may require large amounts of data, although such characteristic may be quite a challenge for a single institution under realistic contexts within the medical imaging domain.

Ideally, organizations could share their data to a centralized location, thus achieving a common goal. However, that is unlikely to happen due to legal regulations with regards to patient data privacy and data-ownership.

As a means to overcome those barriers, *federated learning* arises.

The present Master Final Project is intended to apply the *federated learning* paradigm for a specific case of study (breast tumour detection in ultrasound images) and by using specific federation technology and techniques (the *TensorFlow Federated* library by means of the standard federated aggregation algorithm, *Federated Average*, plus several modification branches).

We will analyse the suitability and efficiency of the resulting federated models and compare them with a classical *Machine Learning* model (i.e., trained on centralized data). We will outline that some federated learning trials have proven to yield a similar degree of efficiency as a classical learning model.

As a guide for this document, first, we will introduce the key concepts related to the present case: the task at hand (breast ultrasound image segmentation), the *Machine Learning* techniques implemented (a *Deep Learning*, convolutional neural network model: *U-Net*) and the principles of *federated learning*.

Then, we will detail the methodology applied in the project and results will be presented.

Finally, we will expose our conclusions and final thoughts derived from the project execution: its limitations and future improvements likely to be tackled both for the present case and, particularly, for progressing on further research around *federated learning* .

Palabras clave: *Deep Learning, federated learning, image segmentation, breast ultrasound imaging, breast cancer detection, TensorFlow*

Índice general

1. Introducción **1**

1.1. Planteamiento del problema . . . 1

1.1.1. Del Aprendizaje Automático al aprendizaje federado . . . 2

1.1.2. Qué es el aprendizaje federado . . . 2

1.1.3. Los datos: cuáles son y qué hacer con ellos . . . 3

1.1.4. Visión por computador . . . 5

1.2. Motivación . . . 7

1.2.1. Aportación . . . 8

1.3. Objetivos y tareas . . . 8

2. Fundamento teórico y estado de la cuestión **11**

2.1. Ultrasonido como técnica diagnóstica de tumores mamarios . . . 11

2.2. *Deep Learning* aplicado a *CAD*: segmentación de imágenes de ultrasonidos . . . 13

2.2.1. Breve introducción al *Deep Learning* y las *CNN's*: evolución e hitos . . . 13

2.2.2. Evaluación en segmentación de imágenes . . . 15

2.2.3. Casos de éxito . . . 17

2.3. Aprendizaje federado . . . 18

2.3.1. Cómo es el proceso de aprendizaje federado . . . 19

2.3.2. Agregación de aprendizajes locales . . . 19

2.3.3. Tipos de aprendizaje federado . . . 20

2.3.4. Retos en el proceso de aprendizaje . . . 23

2.3.5. Algoritmos de aprendizaje federado . . . 24

2.3.6. Recursos y herramientas . . . 26

2.3.7. Casos de éxito . . . 27

3. Materiales y métodos **29**

3.1. Descripción general . . . 29

3.2. Entorno computacional y tecnología . . . 30

3.2.1. Recursos hardware . . . 30

3.2.2. Recursos software . . . 30

3.3. *Datasets* . . . 30
3.4. Arquitectura de red neuronal . . . 32
3.5. Preprocesado . . . 34
3.5.1. Directorio de ficheros . . . 34
3.5.2. Fusión y despliegue de los *datasets* . . . 34
3.5.3. Transformaciones y aumento de datos . . . 35
3.6. Entrenamiento . . . 37
3.6.1. Subconjuntos de entrenamiento y validación . . . 37
3.6.2. Modelos federados . . . 37
3.7. Evaluación . . . 38
3.7.1. Función de pérdida . . . 38
3.7.2. Métricas de rendimiento . . . 38
3.7.3. Evaluación federada . . . 38

4. Experimentos y resultados **41**
4.1. Pruebas . . . 41
4.2. Resultados . . . 42
4.2.1. Métricas de evaluación . . . 42
4.2.2. Predicciones . . . 45
4.2.3. Fase de entrenamiento . . . 51
4.3. Discusión . . . 55
4.3.1. Análisis de los resultados . . . 57
4.3.2. Retos enfrentados . . . 61

5. Conclusiones **63**
5.1. Eficacia del aprendizaje federado . . . 63
5.2. Trabajos futuros . . . 64
5.2.1. Tarea a resolver . . . 64
5.2.2. *Datasets* . . . 64
5.2.3. Preprocesado . . . 64
5.2.4. Entrenamiento y evaluación . . . 65
5.2.5. Aprendizaje federado . . . 65

Bibliografía **67**

A. Apéndice A: gráficas de entrenamiento **77**
A.1. Entrenamiento de modelos *DL*: resultados completos . . . 77

B. Apéndice B: gráficas de entrenamiento (II) 85
B.1. Entrenamiento de modelo clásico prolongado 85

Nomenclatura

ANN Artificial Neuronal Networks - redes neuronales artificiales, página 13

BUSI Breast Ultrasound Image - imagen o ecografía mamaria de ultrasonido, página 3

CAD Computed-Aided Diagnosis, página 2

CLAHE Contrast Limited Adaptive Histogram Equalization, página 35

CNN Convolutional Neuronal Network - red neuronales convolucionales o convolutivas, página 13

DL Deep Learning - Aprendizaje Profundo, página 1

DP Differential Privacy - privacidad diferencial, página 22

FCN Fully Neuronal Network - red completamente convolucionales, página 14

FedAvg Federated Average, página 24

FL Federated Learning - aprendizaje federado, página 2

GT Ground Truth - máscara de etiquetado, página 6

IA Inteligencia Artificial, página 1

IoU Intersection over Union, página 16

ML Machine Learning - Aprendizaje Automático, página 1

MTF Falsas masas tumorales, página 45

MTO Masas tumorales no detectadas u obviadas, página 45

non-IID data non Identically nor Independently Distributed data - datos no independientes y/o no idénticamente distribuidos, página 22

SGD Stochastic Gradient Descent - descenso estocástico de gradiente, página 24

TFF TensorFlow Federated, página 26

TFM Trabajo Fin de Máster, página 8

UNED Universidad Nacional de Educación a Distancia, página 8

Índice de figuras

1.1. Topología simplificada del aprendizaje federado 4
1.2. Ejemplos de *BUSI's* originales 4
1.3. Ejemplos de clasificación de imágenes *BUS* 5
1.4. Localización de objetos en visión por computador 6
1.5. Ejemplos de imágenes *BUS* y máscaras *GT* 7

2.1. Descripción de las zonas de una lesión tumoral en una imagen *BUS* 12
2.2. Número de publicaciones por año sobre *IA, ML, DL y BUS* en *PubMed* 12
2.3. Representación gráfica de patrones aprendidos por una red neuronal 13
2.4. Representación gráfica de la arquitectura *U-Net* 15
2.5. Funciones de pérdida para segmentación semántica 16
2.6. Coeficientes para el cálculo de la función de pérdida en segmentación 17
2.7. Descripción básica del entrenamiento federado 20
2.8. Esquema básico del ciclo de entrenamiento federado participado por instituciones . . 21
2.9. Federación *cross-device* 21
2.10. Federación *cross-silo* 23
2.11. Ataques inferenciales en aprendizaje federado 24
2.12. Comparativa de librerías y *frameworks* de aprendizaje federado 27

3.1. Desbalanceo de clases que presentan los datasets unificados 32
3.2. Distribución de píxeles en los *datasets* 33
3.3. Preprocesado de *datasets* originales (sin división) 35
3.4. Preprocesado de *datasets* originales (con división) 36
3.5. Evaluación clásica y evaluación federada 39

4.1. Predicciones del modelo clásico 46
4.2. Predicciones de modelos federados 47
4.3. Predicciones del modelo clásico: fallos de detección 48
4.4. Predicciones de modelos federados: fallos de detección 48
4.5. Predicciones: falsas masas tumorales 49

4.6. Predicciones: masas tumorales no detectadas u obviadas 50
4.7. Función de pérdida durante el entrenamiento clásico 52
4.8. Función de pérdida durante el entrenamiento federado (3 clientes) 52
4.9. Función de pérdida durante el entrenamiento federado (5 clientes) 53
4.10. Función de pérdida durante el entrenamiento federado (7 clientes) 54
4.11. Exactitud durante el entrenamiento clásico 55
4.12. Exactitud durante el entrenamiento federado (3 clientes) 56
4.13. Exactitud durante el entrenamiento federado (5 clientes) 56
4.14. Exactitud durante el entrenamiento federado (7 clientes) 57
4.15. *Boxplot* de la exactitud 58
4.16. *Boxplot* de la función de pérdida 59
4.17. *Boxplot* del coeficiente *Jaccard / IoU*) 59
4.18. *Boxplot* de la sensibilidad (*recall*) 59

A.1. Función de pérdida durante el entrenamiento federado (3 clientes). 78
A.2. Función de pérdida durante el entrenamiento federado (5 clientes). 79
A.3. Función de pérdida durante el entrenamiento federado (7 clientes). 80
A.4. Exactitud (*accuracy*) durante el entrenamiento federado (3 clientes). 81
A.5. Exactitud (*accuracy*) durante el entrenamiento federado (5 clientes). 82
A.6. Exactitud (*accuracy*) durante el entrenamiento federado (7 clientes). 83

B.1. Función de pérdida durante el entrenamiento clásico prolongado (1000 *epochs*) . . . 85
B.2. Exactitud durante el entrenamiento clásico prolongado (1000 *epochs*) 86

Índice de tablas

2.1. Diferencias entre aprendizajes federados *cross-silo* y *cross-device* 22

3.1. Resumen descriptivo de los *datasets* 31
3.2. Construcción de *datasets* finales de entrenamiento a partir de los originales A, B y C 36

4.1. Pruebas ejecutadas para el proyecto 41
4.2. Pruebas federadas ejecutadas en cada bloque FED-x 42
4.3. Relación de clientes federados con los *datasets* disponibles en cada bloque de pruebas federadas 43
4.4. Puntuación (*coeficiente Dice*) de las pruebas de evaluación de test 43
4.5. Puntuación (exactitud, *accuracy*) de las pruebas de evaluación de test 44
4.6. Puntuación (*coeficiente IoU* o *Jaccard*) de las pruebas de evaluación de test 44
4.7. Puntuación (sensibilidad) de las pruebas de evaluación de test 45
4.8. Puntuación del índice de elaboración propia MTF sobre los modelos entrenados. . . 48
4.9. Puntuación del índice de elaboración propia MTO sobre los modelos entrenados. . . 48
4.10. Puntuación de los índices de elaboración propia MTF y MTO: promedios por bloque de entrenamiento. 49
4.11. Puntuación de los índices de elaboración propia MTF y MTO: promedios por test de entrenamiento federado. 51
4.12. Duración de la fase de entrenamiento para cada modelo objeto de estudio 51

Capítulo 1

Introducción

En este primer capítulo, introduciremos los conceptos básicos alrededor de los cuales girará la lectura del presente documento. Comenzaremos con una aproximación al Aprendizaje Automático que, aun resultando muy introductoria, nos va a permitir apuntar los aspectos clave que justifican la aplicación del aprendizaje federado. También explicaremos este paradigma federado y el problema concreto a abordar bajo el mismo.

1.1. Planteamiento del problema

El campo de la Inteligencia Artificial (IA), concretamente las áreas del Aprendizaje Automático (*Machine Learning*, *ML*) y el aprendizaje profundo (*Deep Learning*, *DL*, una rama de la anterior), han experimentado enormes progresos desde principios del siglo XXI hasta la actualidad. Estos avances están permitiendo la aparición de multitud de nuevas técnicas que tratan de ofrecer respuestas autónomas a problemas de toda índole. Problemas, por lo general, complejos, extensos y que históricamente han requerido de cierta o completa intervención humana; tareas que la tecnificación no había sido capaz de resolver con éxito, hasta hace relativamente poco tiempo. La medicina es una de esas áreas en las que la IA está aportando soluciones automatizadas, tiempo atrás inabordables, presentando unas cuotas de robustez y eficacia elevadas. Una de esas tareas es la ayuda al diagnóstico por imagen.

El diagnóstico por imagen consiste en la detección de patologías mediante observación por personal médico cualificado de una prueba visual (Yap et al. (2010)). Un ejemplo de patología es el cáncer (Stavros (2004)) y ejemplos de pruebas diagnósticas visuales son imágenes fotográficas, de ultrasonidos, microscópicas (tejidos, sangre...), radiológicas, etcétera. Además de la variedad en cuanto a la técnica visual empleada, hay que contar con la variedad en cuanto a la zona corporal u órgano vital objeto de diagnóstico. Esto multiplica el abanico de problemas concretos que pueden abordarse con el diagnóstico por imagen.

Desde el punto de vista del Aprendizaje Automático, éste puede aplicarse al diagnóstico por

imagen y es lo que denominamos la ayuda automatizada al diagnóstico por imagen o diagnóstico por imagen automatizado (*CAD*, *Computed-Aided Diagnosis*). Según la técnica y el órgano objeto de análisis, se concretan dominios específicos de tareas automatizables. El aprendizaje profundo y, en concreto, las redes neuronales, sobresalen en la asistencia a dichas tareas.

1.1.1. Del Aprendizaje Automático al aprendizaje federado

El *Aprendizaje Automático* es la ciencia de crear programación que aprenda por sí misma de los datos (Géron (2019). En palabras de Arthur Samuel (1959), es *"el campo de estudio que provee a las computadoras la habilidad de aprender sin ser programadas explícitamente"*.

En síntesis, los algoritmos de Aprendizaje Automático son la base de modelos inteligentes con capacidad de aprender (de los datos del entrenamiento) y con capacidad predictiva (inferir una respuesta sobre nuevas muestras no contempladas anteriormente). Con relación al mencionado caso del diagnóstico *CAD*, son estas capacidades las que hacen posible ejecutar tareas de manera automatizada, como localizar una masa tumoral en una imagen o determinar el tipo de tumor presente en la misma.

Hace años que los algoritmos *ML* comenzaron a producir resultados muy prometedores en diversas áreas de aplicación gracias, principalmente, al tremendo aumento en las capacidades de computación y a la disponibilidad de grandes volúmenes de datos [1] en las últimas décadas.

Pero no siempre se puede disponer de tanta información; concretamente, en el caso que nos ocupa, la dificultad estriba en que los datos clínicos se encuentran custodiados por organizaciones, tanto públicas como privadas, que velan por su integridad, privacidad y seguridad. Dicha custodia se rige por leyes que la sociedad misma nos damos. Esta legislación, conviene remarcarlo, es un cuerpo vivo en constante evolución, tanto a nivel nacional como internacional (estatal, europeo, global...). Por tanto, el acceso a la información (médica, en el caso que nos ocupa) necesaria para entrenar algoritmos *ML* no es abierto. Y esto supone un obstáculo crítico a la hora de aplicar en contextos de la vida real las técnicas surgidas en el ámbito de la comunidad investigadora. Para intentar superar este reto, en los últimos años ha surgido el paradigma del aprendizaje federado (*federated learning, FL*).

1.1.2. Qué es el aprendizaje federado

En un enfoque clásico o *estándar*, el entrenamiento de un modelo de Aprendizaje Automático se encuentra centralizado en una única máquina; el algoritmo o modelo es único y es entrenado con los datos de que se dispone (muchos, usualmente) bajo un único entorno computacional. Desde esta perspectiva, los datos, todos ellos, se encuentran unificados y es así como se ponen a disposición de dicho modelo de Aprendizaje Automático. Expresado de otra manera, los datos "viajan" al algoritmo.

[1] Cuando hablamos de "*datos*", en el contexto de la tarea *CAD*, nos referimos a aquellas imágenes de soporte al diagnóstico.

La dificultad viene cuando ese escenario no es real, es decir, cuando los datos se encuentran descentralizados, dispersos en distintos entes. Al hablar de entes, podemos referirnos a organizaciones, instituciones, particulares... En definitiva, entornos digitales individuales, generalmente estancos y aislados entre sí. Puede ser algo tan complejo como un centro de datos de una empresa multinacional, los servidores de un hospital, o algo tan simple como un dispositivo *smartphone*, como el que llevamos en el bolsillo. En el contexto del Aprendizaje Automático, cada ente posee su propio conjunto de datos, por lo general de volumen más reducido en comparación a los tamaños gestionados en un entorno teórico o estándar. Debido a la estanqueidad y privacidad que cada ente propio debe asegurarse, no es posible unificar los datos bajo una única localización. Por ello y de forma natural, dentro de la comunidad desarrolladora e investigadora han surgido iniciativas que tratan de afrontar esta problemática. El *aprendizaje federado* es un ejemplo de ello.

El término aprendizaje federado fue acuñado por primera vez en McMahan et al. (2016): *"[...] aprendizaje federado, ya que la tarea de aprender es resuelta mediante una federación libre entre dispositivos participantes (o clientes) coordinados por un servidor central."*.

En esencia, el aprendizaje federado permite procesos de Aprendizaje Automático en los contextos descritos, respetando las limitaciones en cuanto a acceso y descentralización de los datos, asegurando la preservación de la privacidad de estos y, a ser posible y como mínimo, ofreciendo resultados sin merma en la calidad.

Bajo estas premisas, el proceso de aprendizaje se *federa* o delega a los distintos entes que pueden participar del entrenamiento del modelo *ML*: es el algoritmo el que "viaja" a los datos. En cada uno de los entes intervinientes, el modelo aprenderá cuanto pueda. Tras ello, retornará a una matriz (un servidor) que coordina el aprendizaje y donde se consensua lo aprendido en los distintos entes participantes en el aprendizaje. Una topología simplificada de aprendizaje federado puede observarse en la figura 1.1.

Sirva un caso de éxito muy popular e ilustrativo para captar con claridad el concepto de aprendizaje federado y su utilidad real: la aplicación *Gboard* de ayuda a la auto compleción de texto para dispositivos móviles Kairouz et al. (2019). Esta herramienta, su facultad de ayudar a la inserción de texto por el usuario, fue desarrollada mediante aprendizaje federado.

1.1.3. Los datos: cuáles son y qué hacer con ellos

Volviendo a los datos, también referidos como conjuntos de datos o *datasets*, el objeto de análisis en el marco del presente proyecto serán imágenes de ultrasonidos de órganos mamarios.

Una imagen de ultrasonido o ecografía mamaria (*BUSI*, *Breast Ultrasound Image*, por sus siglas en inglés) es una representación digital gráfica, generalmente en blanco y negro, captada y procesada por un aparato de escáner de ultrasonidos, Al-Dhabyani et al. (2020); Cheng et al. (2010), como pueden verse en la figura 1.2. Estos equipos pueden tomar una secuencia de imágenes (vídeo), pero aquí tratamos con instantáneas del proceso de prueba diagnóstica.

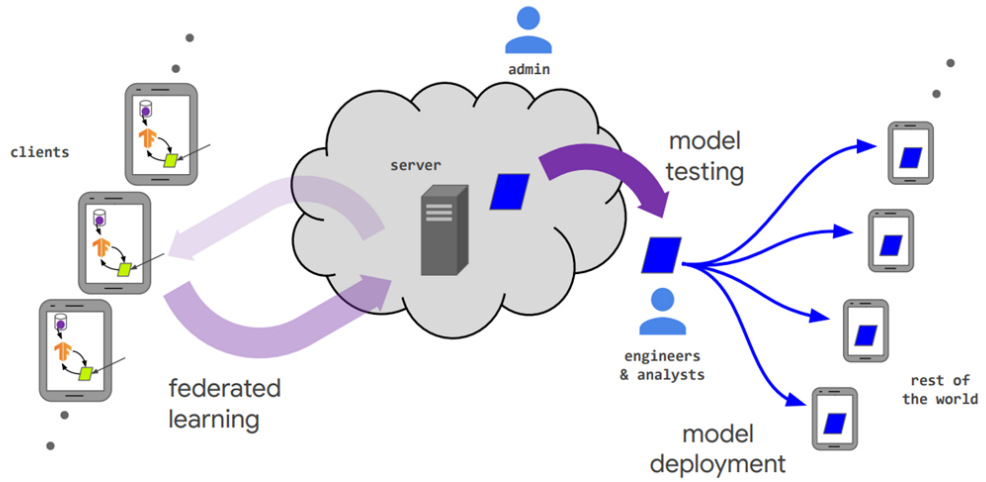

Figura 1.1: Topología simplificada del aprendizaje federado, tomada de Kairouz et al. (2019)

Figura 1.2: Ejemplos de *BUSI*'s originales, tomados de Al-Dhabyani et al. (2020)

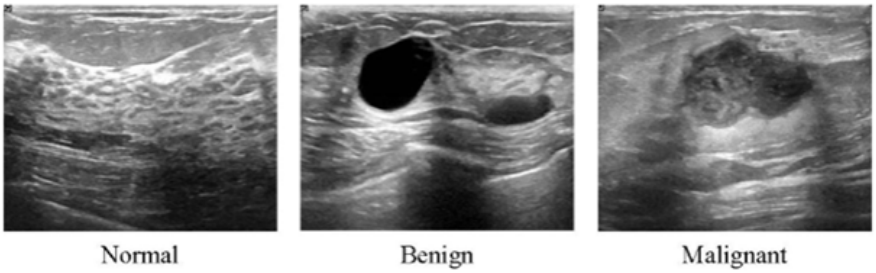

Figura 1.3: Ejemplos de clasificación de imágenes *BUS*, tomados de Al-Dhabyani et al. (2020)

Cada imagen o instantánea representa un elemento o muestra del conjunto de datos. Con esa imagen se pretende descubrir la existencia o no de masas tumorales en los órganos mamarios. Estas masas tumorales pueden ser benignas o malignas. Podemos distinguir dos vertientes concretas del diagnóstico *CAD*:

1. Localizar (*segmentar*) una masa tumoral en una imagen
2. Determinar (*clasificar*) o tipificar el tipo de tumor presente

El Aprendizaje Automático puede ser aplicado en ambos casos. Determinar el tipo de tumor suele asociarse a la tarea de *clasificación*: dada una imagen *BUS*, el objetivo es indicar la categoría de tumor presente en aquella (si lo hubiera). Resultados posibles de ese ejercicio de clasificación podrían ser palabras o etiquetas como "*benigno*", "*maligno*" o "*sin tumor*", por ejemplo. Nótese que esta etiqueta se asocia a la imagen en su conjunto; no se obtiene más información que esa, como puede apreciarse en la figura 1.3. La eficacia de los resultados obtenidos mediante esta técnica puede ser relativamente baja, debido mayormente a que la imagen analizada abarca una zona demasiado amplia y solamente una parte de la misma está relacionada con los signos que deben influir en la clasificación. De ahí que tome relevancia la otra etapa, la de localización de los signos de interés. Y, en este caso, las posibilidades son mayores.

1.1.4. Visión por computador

El objetivo es, pues, localizar con precisión la masa tumoral, esto es, determinar el correspondiente conjunto de píxeles dentro de la imagen. Esta es una tarea propia de la denominada *visión por computador* (*computer vision*). Dentro de la visión por computador, existen distintas maneras de abordar la tarea de *localización* dentro de una imagen. Es importante explicar estas diferencias, pues ayudarán a entender y delimitar los objetivos específicos del presente proyecto. La figura 1.4 presenta un ejemplo gráfico de resolución de cada una de esas maneras. En esencia, se diferencian en qué tipo de información permiten predecir Michelucci (2019):

1. *Detección de objetos*: determinar el encuadre de uno o varios objetos dentro de la imagen mediante cajas rectangulares que lo acoten.

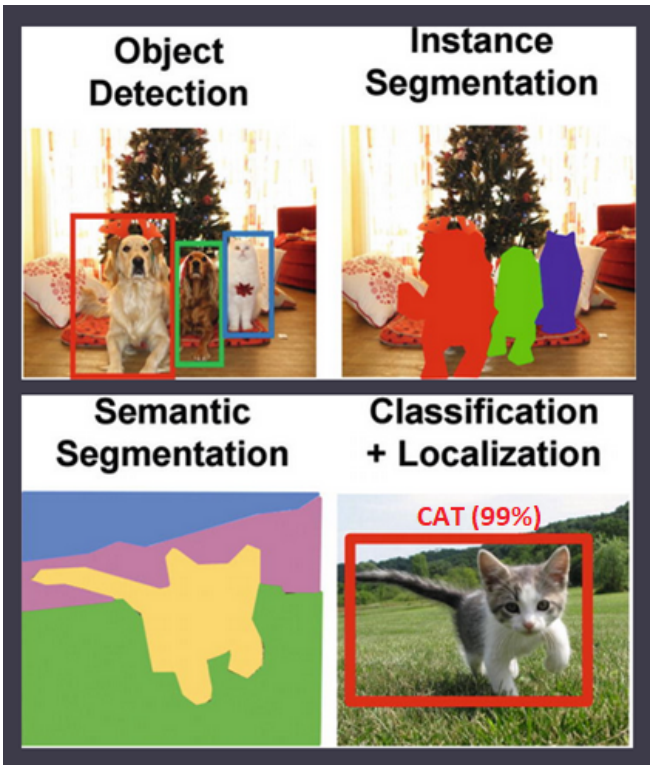

Figura 1.4: Ejemplos visuales de localización de objetos en visión por computador, en Michelucci (2019)

2. *Clasificación + localización*: combinación de la clasificación y la detección de objetos. Este caso sería aplicable solo con el objetivo de localizar un único objeto en la imagen.

3. *Segmentación semántica*: cada píxel de la imagen será asociado a una categoría o etiqueta a la que pertenece. Siguiendo un ejemplo con el presente proyecto, un píxel podría asociarse a dos categorías: "*tumor*" o "*no tumor*". En este caso, si apareciesen dos tumores en la misma imagen, la tarea no distingue entre ambas instancias; ambos pertenecerían a la misma clase o categoría.

4. *Segmentación de instancias*: una suerte de combinación de la detección de objetos y la segmentación semántica; cada píxel se asocia a una clase y, además, permite la diferenciación de objetos de la misma clase.

La tarea que se resolverá en este proyecto es la *segmentación semántica*. Para ello, cada imagen *BUS* debe acompañarse de una imagen adicional, fruto del diagnóstico manual realizado por un profesional técnico. Concretamente, esta imagen adicional representa la superficie ocupada por la masa tumoral, si la hubiera, respecto de la imagen original. Esta imagen-diagnóstico, conocida como máscara o *ground truth (GT)*, es una representación de imagen binaria. Nótense los ejemplos de la figura 1.5.

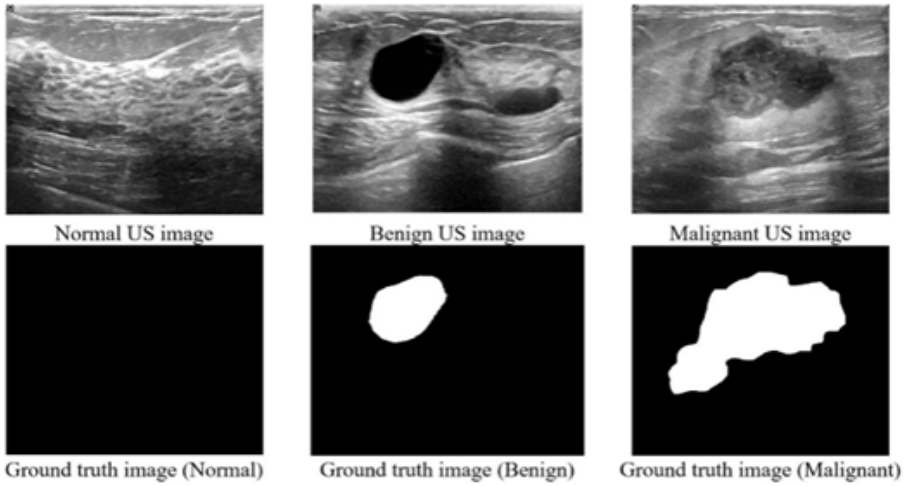

Figura 1.5: Ejemplos de imágenes *BUS* y máscaras *ground truth*, tomados de Al-Dhabyani et al. (2020)

1.2. Motivación

Dos motivaciones principales fundamentan este proyecto fin de máster: abordar una tarea de diagnóstico por imagen de relevancia y crear una solución para la ayuda al diagnóstico por imagen lo más realista posible.

El primero, desde la perspectiva médica, es el diagnóstico por imagen concreto escogido: la detección de masas tumorales en órganos mamarios de mujeres. En 2010, el cáncer de mama ya suponía uno de los tumores más diagnosticados en mujeres Cheng et al. (2010). En 2016, se convertía en el más común de los cánceres (sobrepasando al cáncer de pulmón) y la segunda causa de muerte entre la población femenina, a nivel mundial Siegel et al. (2016). Así continuaba en 2019 Siegel et al. (2019) y 2020 Sung et al. (2021), con un número estimado de nuevos casos al año de 2.3 millones (11,7 % del total de tumores diagnosticados). El diagnóstico temprano juega un papel clave en la prevención de muertes por cáncer de mama, ya que aumenta la proporción de curación y recuperación Rodrigues (2017). Trabajar en soluciones que ayuden al diagnóstico temprano de esta enfermedad es una tarea, desde luego, alentadora.

La segunda motivación fundamental es testar una solución de Aprendizaje Automático, aplicado a *CAD*, que sea extrapolable, en términos factibles, al mundo real. Esto es, evaluar y contrastar los resultados del aprendizaje federado, simulando ciertas características propias de un contexto médico-profesional más realista. Como se ha explicado anteriormente, la privacidad de los datos clínicos y la estanqueidad de las organizaciones médicas lastran la proliferación de *datasets* públicos y de gran volumen que ayuden a la comunidad investigadora en el desarrollo de soluciones *Machine Learning* a la manera clásica, esto es, pudiendo centralizar los *datasets* y computar un único entrenamiento unificado.

1.2.1. Aportación

El aprendizaje federado es la materia más novedosa que aportará este TFM en el ámbito del Máster en Ingeniería y Ciencia de Datos de la UNED y, por ello, será objeto de mayor énfasis en el presente documento.

Como veremos en el repaso del estado de la cuestión, capítulo 2, existen muchas referencias en la literatura que contribuyen en alguna o varias áreas de las implicadas en el presente proyecto (diagnóstico por imagen, técnicas de aprendizaje profundo, aprendizaje federado. . .). Pero, hasta la fecha, no se ha encontrado ninguna aportación de relevancia que englobe los tres pilares de este trabajo: 1) *Deep Learning* (redes neuronales) aplicado a 2) *CAD* de imágenes *BUS* mediante 3) aprendizaje federado.

En este sentido, el presente trabajo puede también tomarse como una aportación iniciática y transversal al caso de uso abordado.

1.3. Objetivos y tareas

Los objetivos del proyecto son los siguientes:

1. Crear modelos *Deep Learning* que permitan predecir la presencia de masas tumorales en órganos mamarios de mujeres mediante el análisis de imágenes de ultrasonido.

2. Para ello, resolveremos la tarea de segmentación semántica (binaria) de imágenes: la tarea de localización de la masa tumoral no distinguirá si ésta es benigna o maligna, es decir, la clasificación se deja para un posterior sistema, fuera del alcance del presente proyecto.

3. Los modelos *DL* serán entrenados mediante las técnicas de aprendizaje clásico y aprendizaje federado.

4. Los modelos, esto es, sus resultados, serán contrastados unos con otros. Pondremos el foco en el desempeño del aprendizaje federado, tanto relativo (respecto del aprendizaje clásico) como absoluto, es decir, si realmente suponen herramientas efectivas como tal para el diagnóstico *CAD.* En concreto, analizaremos el impacto en términos de generalidad y bondad que presentan los modelos federados.

5. Finalmente, se expondrán las oportunas valoraciones sobre los posibles beneficios y/o riesgos del aprendizaje federado derivadas en el contexto de este trabajo.

Estos objetivos se materializarán a través de las siguientes tareas:

1. Entrenaremos un modelo mediante *aprendizaje clásico*. Este modelo hará las labores de *gold standard* o referencia para la comparativa con el aprendizaje federado.

2. Entrenaremos varios modelos mediante *aprendizaje federado*. Los distintos modelos federados se diferenciarán fundamentalmente en:

   - El número de *datasets* participantes de la federación.
   - La técnica de agregación federada implementada.

3. Emplearemos la librería *TensorFlow Federated* como vehículo para la federación. Se explorarán las técnicas por defecto disponibles en dicha librería y su desempeño en un entorno computacional potente.

4. Se dispone de tres conjuntos de datos *BUSI* principales.

   - Para la creación del modelo con aprendizaje clásico, todos los *datasets* serán fusionados en un único conjunto.
   - Para las creación de los modelos con aprendizaje federado, proveeremos de varias combinaciones que enriquezcan el análisis posterior. Dispondremos de tres, cinco y siete *datasets* o clientes federados. En los dos últimos casos, producto de particionar varios de los *datasets* originales.

5. Implementaremos una única arquitectura y parametrización de red neuronal, concretamente, *U-Net* Ronneberger et al. (2015), para todos los modelos a entrenar.

Capítulo 2

Fundamento teórico y estado de la cuestión

En este capítulo, haremos un recorrido por el estado del arte que fundamenta el presente proyecto. En concreto, sus tres pilares principales: el objeto médico de análisis (imágenes de ultrasonidos), el área de ciencia de datos que lo resuelve (Aprendizaje Automático, *Deep Learning*, redes neuronales artificiales) y, en mayor detalle, el paradigma novedoso para cómo resolverlo (aprendizaje federado).

2.1. Ultrasonido como técnica diagnóstica de tumores mamarios

Las ecografías o imágenes de ultrasonidos destacan en la detección de tumores mamarios por sus bondades sobre otras técnicas de diagnóstico, como son la biopsia celular Siegel et al. (2016); Liu et al. (2010) o las radiografías Cheng et al. (2010). La técnica ecográfica no es radiactiva, es indolora para el paciente, económicamente eficiente y altamente disponible Cheng et al. (2010). Detalles en profundidad sobre la labor técnica profesional pueden encontrase en Stavros (2004). Baste un ejemplo descriptivo de la morfología de una masa tumoral (figura 2.1) para tomar conciencia de la tarea.

Las imágenes de ultrasonido son comúnmente usadas para el diagnóstico como apoyo a los radiólogos en la detección de masas tumorales: *"[...] en la práctica actual, el médico escanea el órgano mamario y toma imágenes estáticas. El radiólogo evaluará y anotará las imágenes BUS. Los sistemas de diagnóstico asistido por computador (CAD) se pueden utilizar como un "segundo lector" para el análisis computarizado de imágenes médicas."* Yap et al. (2018).

Para comenzar a abordar el estado de la cuestión desde una perspectiva global y, al tiempo, clara y concisa, puede resultar muy práctico comenzar con Cheng et al. (2010). Aunque ciertos temas han quedado superados, sobre todo lo referente a algoritmos, siguen siendo remarcables casi todas las partes del proceso completo (*pipeline*): la definición de ayuda al diagnóstico por computador, *CAD*, la descripción de la muestra (imágenes *BUS*), los retos que presenta, la visión detallada de la fase

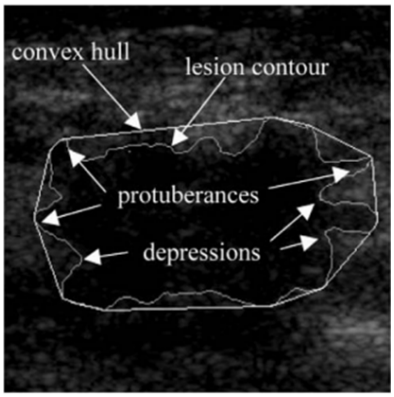

Figura 2.1: Descripción de las zonas de una lesión tumoral en una imagen *BUS*, tomada de Cheng et al. (2010)

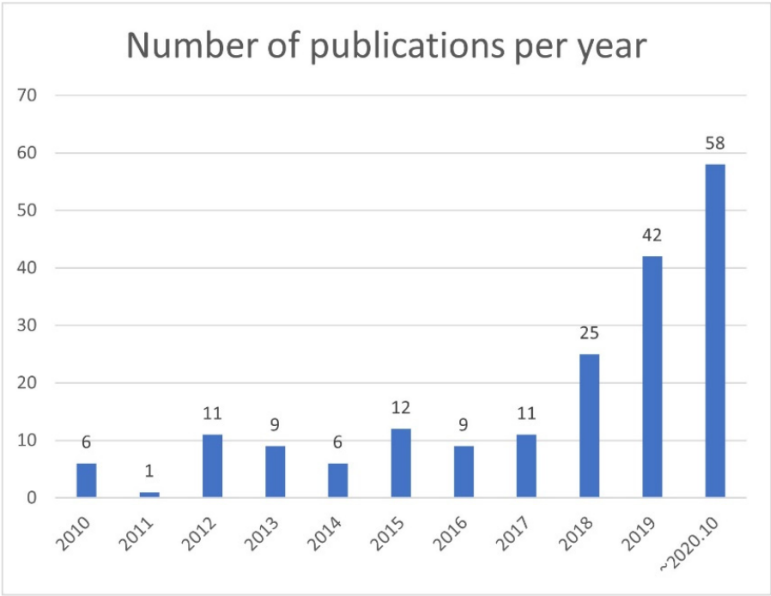

Figura 2.2: Número de publicaciones por año sobre *IA, ML, DL y BUS* en la base de datos bibliográfica *PubMed* - tomada de Fujioka et al. (2020), figura 3

de preprocesamiento o las herramientas de evaluación (métricas) son algunas de sus características destacables.

Posteriormente, es muy recomendable leer Fujioka et al. (2020), el cual resume cómo el *Deep Learning* está produciendo avances significativos en la ayuda al diagnóstico de tumores mamarios mediante imágenes de ultrasonidos, con su amplio abanico de algoritmos disponibles. Prueba de ello es la proliferación de contribuciones por parte de la comunidad investigadora ha sido y es ingente, como muestra la figura 2.2 tomada del citado Fujioka et al. (2020).

Si ampliamos la búsqueda hacia aspectos como el preprocesado de las imágenes, la parametrización específica o la combinación múltiple de modelos de red neuronal, comprobaremos que las aportaciones disponibles en la literatura son aún más numerosas Gupta et al. (2018).

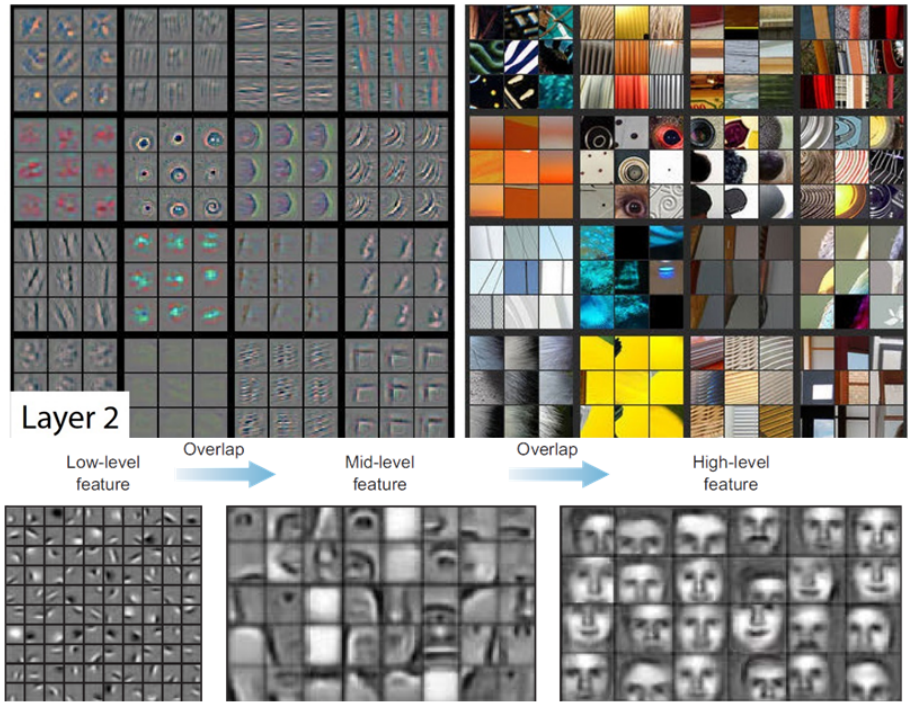

Figura 2.3: Ejemplos de representación gráfica de patrones aprendidos por una red neuronal, arriba Zeiler and Fergus (2014) y abajo Ivan Vasilev (2019).

2.2. *Deep Learning* aplicado a *CAD*: segmentación de imágenes de ultrasonidos

2.2.1. Breve introducción al *Deep Learning* y las *CNN's*: evolución e hitos

Los modelos de redes neuronales artificiales (*Artificial Neuronal Networks*, ANN's) tienen la capacidad de aprender representaciones y patrones a través de los datos (imágenes) que usan durante el aprendizaje. Estos métodos de aprendizaje extraen o descubren automáticamente las características (*features*) inherentes a los datos usados en el aprendizaje, sin requerir de unas reglas predefinidas. Cada tipología de red *ANN* presenta múltiples niveles y modos de conexión neuronal; en cada nivel, las características aprendidas representarán una pieza del aprendizaje en su conjunto Yap et al. (2018).

De todos los tipos de *ANN's*, hay un grupo que destaca: las redes neuronales convolucionales (*CNN*'s, *Convolutional Neuronal Networks*). Las *CNN*'s se han erigido como una técnica importantísima en la visión por computador y el análisis de imágenes, particularmente, en la detección de objetos Yap et al. (2018). La figura 2.3 muestra la capacidad de estas redes de aprender patrones gráficos. Los modelos *CNN* también permiten clasificar objetos presentes en una imagen *BUS* (por ejemplo, una masa tumoral) o determinar qué clase o categoría corresponde a una imagen dada (esto es, determinar si un tumor es benigno o maligno).

Desde luego, en el área del *Deep Learning*, las *CNN*'s presentan un alto grado de eficacia,

demostrada a lo largo de los años, fruto del desarrollo de innovaciones continuas que han mejorado y superado resultados respecto de sus predecesoras. Una prueba de ello es la evolución demostrada en la competición *ILSVRC* del *ImageNet challenge* ILSVRC (2022). En esta competición y en el curso de tan solo seis años, la ratio de error de los cinco modelos participantes más eficaces se redujo drásticamente, del 26 % al 2,3 % Géron (2019).

Las arquitecturas *CNN* que han marcado hitos destacables en esta vertiginosa evolución son *LeNet-5* (1998) Lecun et al. (1998), *AlexNet* (2012) Krizhevsky et al. (2017), *GoogLeNet* (2014) Szegedy et al. (2015), *VGGNet* (2014) Simonyan and Zisserman (2014), *ResNet* (2015) He et al. (2015), *XCeption* (2016) Chollet (2016) y *SeNet* (2017) Hu et al. (2017). Todas estas arquitecturas, en su forma original, permiten abordar la tarea de clasificación y, en algunos casos, la detección de objetos de relativa complejidad.

En esta carrera, surgió el concepto de redes totalmente convolucionales (*FCN's*, *Fully Convolutional Networks*) Long et al. (2014). Básicamente, se trata de una modificación de las *CNN*'s mediante la que se sustituyen las últimas capas "densas" por capas convolutivas, permitiendo 1) mayor capacidad predictiva y 2) mayor flexibilidad en cuanto a las dimensiones de las imágenes usadas. De ahí surgen las arquitecturas realmente capaces de resolver la detección de objetos complejos (más clasificación) y, además en algunos casos, ser extensibles a la tarea de segmentación de instancias. Las arquitecturas más destacadas son *Region-Based CNN*, o *R-CNN* (2014) Girshick et al. (2013), y sus subsecuentes versiones mejoradas *Fast R-CNN* (2015) Girshick (2015), *Faster R-CNN* (2015) Ren et al. (2015) y *Mask R-CNN* (2018) He et al. (2017); también sobresalen *YOLO* (2015) Redmon et al. (2015) y sus actualizaciones, *YOLOv2* (2016) Redmon and Farhadi (2016) y *YOLOv3* (2018) Redmon and Farhadi (2018); y *Single-Shot MultiBox Detector (SSD)* (2015) Liu et al. (2015).

Por último, en segmentación de imágenes, debemos destacar la arquitectura concreta implementada en el presente proyecto: *U-Net* (2015), propuesta por Ronneberger et al. (2015). El modelo *U-Net* o *UNet* es, probablemente, la arquitectura de red neuronal más popular a la hora de abordar la segmentación de imágenes Byra et al. (2020). Se trata de una modificación extendida de una arquitectura *FCN*, un *codificador-decodificador* con el añadido de conexiones puente que ayudan en la optimización. El nombre de *U-Net* viene de la forma de *U* que adopta la red en su representación gráfica, como muestra la figura 2.4. La arquitectura *U-Net* también puede incluir capas de *batch normalization* o *dropout* después de las transformaciones de *pooling* y convolución transpuesta, a fin de aliviar los posibles problemas de explosión o desvanecimiento de gradientes. Una de las grandes virtudes de la arquitectura *U-Net* es que no requiere ser entrenada con *datasets* excesivamente grandes para obtener resultados significativos. Este hecho ha resultado diferencial en su elección para el presente proyecto. *U-Net*, en combinación con técnicas de aumento de datos sobre imágenes *BUS* y sus correspondientes anotaciones o máscaras (aprendizaje supervisado), demostrará un alto grado de eficacia en el reto del presente proyecto.

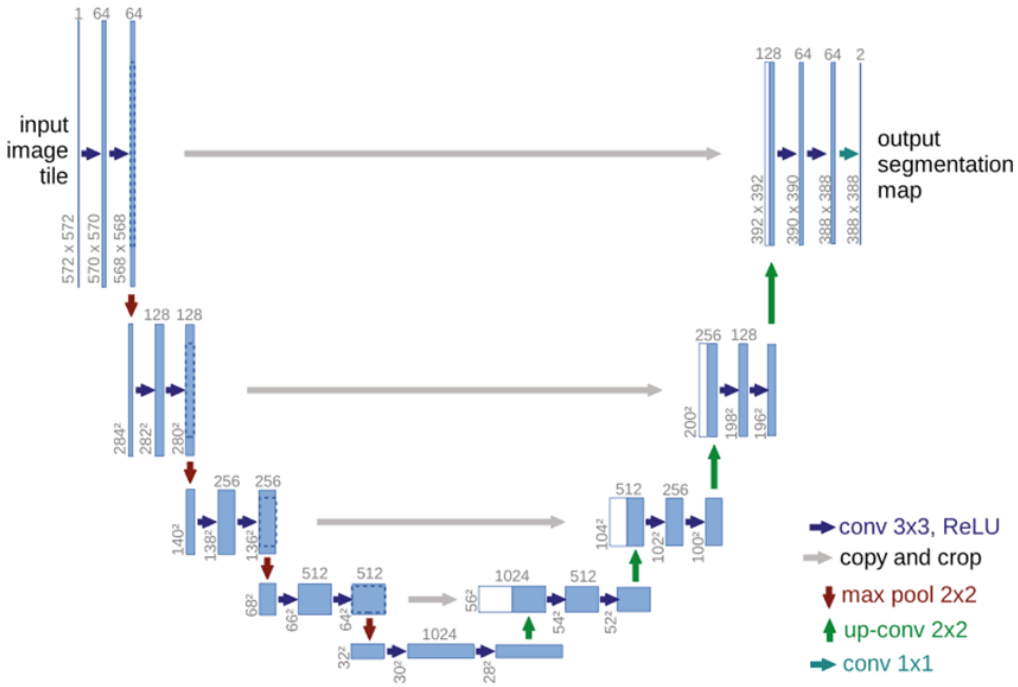

Figura 2.4: Representación gráfica de la arquitectura U-Net, tomada de Ronneberger et al. (2015)

2.2.2. Evaluación en segmentación de imágenes

Al igual que otros aspectos del aprendizaje profundo, la evaluación depende del tipo de problema que se esté abordando. Para el caso que nos ocupa, la segmentación binaria de imágenes, se pueden aplicar las métricas típicas empleadas en clasificación, entre las que destaca la exactitud u otras también derivadas de una matriz de confusión, como son precisión, cobertura, curva *ROC* o el área bajo la curva (*AUC*). Esto es así porque la segmentación de imágenes puede entenderse, y es que en verdad así es, como un ejercicio de clasificación de píxeles. Cada píxel en una imagen toma un valor que corresponde a una categoría o clase a predecir. En el caso presente, un píxel puede clasificarse de dos maneras: "*tumor*" o "*no tumor*"; numéricamente, convendremos uno (1) y cero (0), respectivamente.

Para cada imagen, la evaluación consiste en confrontar la predicción de cada píxel con su correspondiente etiquetado o píxel *ground truth*, considerándose acierto si sendos valores coinciden y fallo, en caso contrario. Así, es posible calcular las métricas y generar las herramientas gráficas descritas anteriormente.

Ahora bien, el caso de uso que nos ocupa presenta desbalanceo de clases: 92 % masa no tumoral, 8 % masa tumoral. Por ello, debemos emplear funciones de pérdida que encaucen la optimización del modelo a penalizar los fallos de las clases minoritarias en mayor medida. En nuestro caso, cuando el píxel pertenece a la clase "*tumor*".

Loss Function	Use cases
Binary Cross-Entropy	Works best in equal data distribution among classes scenarios Bernoulli distribution based loss function
Weighted Cross-Entropy	Widely used with skewed dataset Weighs positive examples by β coefficient
Balanced Cross-Entropy	Similar to weighted-cross entropy, used widely with skewed dataset weighs both positive as well as negative examples by β and $1-\beta$ respectively
Focal Loss	works best with highly-imbalanced dataset down-weight the contribution of easy examples, enabling model to learn hard examples
Distance map derived loss penalty term	Variant of Cross-Entropy Used for hard-to-segment boundaries
Dice Loss	Inspired from Dice Coefficient, a metric to evaluate segmentation results. As Dice Coefficient is non-convex in nature, it has been modified to make it more tractable.
Sensitivity-Specificity Loss	Inspired from Sensitivity and Specificity metrics Used for cases where there is more focus on True Positives.
Tversky Loss	Variant of Dice Coefficient Add weight to False positives and False negatives.
Focal Tversky Loss	Variant of Tversky loss with focus on hard examples
Log-Cosh Dice Loss(ours)	Variant of Dice Loss and inspired regression log-cosh approach for smoothing Variations can be used for skewed dataset
Hausdorff Distance loss	Inspired by Hausdorff Distance metric used for evaluation of segmentation Loss tackle the non-convex nature of Distance metric by adding some variations
Shape aware loss	Variation of cross-entropy loss by adding a shape based coefficient used in cases of hard-to-segment boundaries.
Combo Loss	Combination of Dice Loss and Binary Cross-Entropy used for lightly class imbalanced by leveraging benefits of BCE and Dice Loss
Exponential Logarithmic Loss	Combined function of Dice Loss and Binary Cross-Entropy Focuses on less accurately predicted cases
Correlation Maximized Structural Similarity Loss	Focuses on Segmentation Structure. Used in cases of structural importance such as medical images.

Figura 2.5: Funciones de pérdida para segmentación semántica - Tabla II en Jadon (2020)

2.2.2.1. Taxonomía de las funciones de pérdida en segmentación semántica

Existe una diversidad de métricas que resultan convenientes para resolver la segmentación semántica. Un buen resumen puede encontrase en Jadon (2020), que tipifica las funciones de pérdida más comunes e idóneas en segmentación de imágenes. La figura 2.5 es un compendio de las funciones de pérdida más destacables a la hora de abordar la segmentación.

Como puede apreciarse en dicha figura, varias de estas funciones se basan en modificaciones o combinaciones de otras también presentes en el listado. Ma (2021) ofrece una visualización práctica de tales relaciones, resumida en la figura 2.6. Sus diferencias son sutiles pero resultarán más o menos indicadas dependiendo del problema concreto a resolver y de las posibles características del conjunto de datos abordado.

Dos de las funciones de pérdida más populares presentes en la literatura son el *coeficiente de similaridad Dice* o *coeficiente Sørensen–Dice* Dice (1945); SORENSEN (1948) y el *coeficiente Jaccard* o *coeficiente IoU* (*Intersection over Union*) *Jaccard (1912)*. Ambos permiten medir cuan exacto se superponen dos conjuntos. Véanse sendas fórmulas 2.1 y 2.2, Benjamin Planche (2019).

$$\mathbf{Dice}\,(A,B) = \frac{2\,|A \cap B|}{|A| + |B|} \tag{2.1}$$

$$\mathbf{IoU}\,(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.2}$$

En el contexto de la segmentación, lo que se mide es la superposición de la predicción (imagen *BUS*) sobre el etiquetado real (imagen-máscara *ground truth*). Más concretamente, la superposición de los píxeles uno a uno.

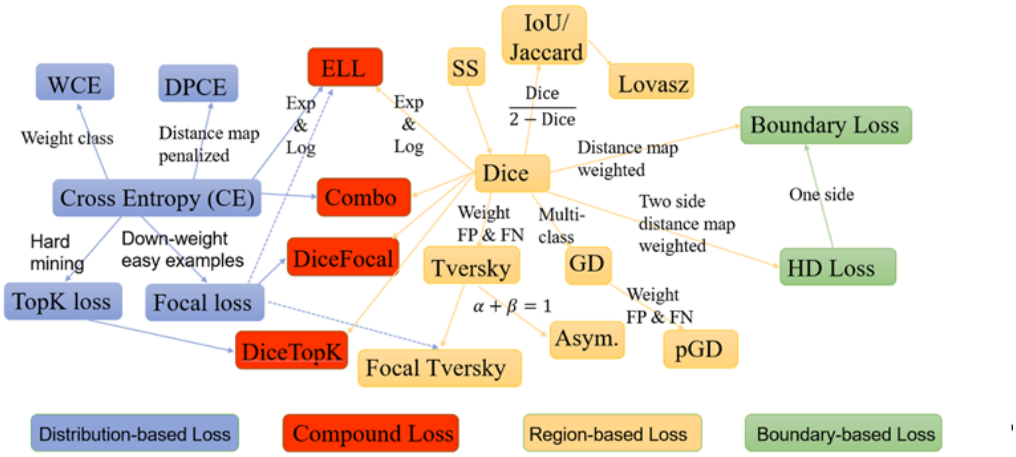

Figura 2.6: Coeficientes para el cálculo de la función de pérdida en segmentación y sus interrelaciones, grafo tomado de Ma (2021)

El *coeficiente Dice* no depende del número relativo de píxeles de cada clase, lo cual es muy indicado para problemas con clases desbalanceadas, como es el caso del presente proyecto. De hecho, el coeficiente *Dice* es equivalente al *f1-score*, más popular como herramienta en problemas de clasificación.

Respecto al *coeficiente Jaccard* o *coeficiente IoU*, no se considera idóneo para guiar el aprendizaje con clases desbalanceadas, pero sí se pondera como una medida realística de la bondad de un modelo de segmentación: Lo cual es lógico, ya que este *coeficiente IoU* mantiene cierta equivalencia con el *coeficiente Dice*, en tanto que crece y decrece en su misma dirección, como puede comprobarse en la fórmula 2.3.

$$\mathbf{IoU}\,(A,B) = \frac{\mathbf{Dice}\,(A,B)}{2-\mathbf{Dice}\,(A,B)}\;;\;\mathbf{Dice}\,(A,B) = \frac{2\times\mathbf{IoU}\,(A,B)}{1+\mathbf{IoU}\,(A,B)} \tag{2.3}$$

2.2.3. Casos de éxito

Centrándonos en la segmentación de imágenes *BUS*, destacan varias soluciones presentes en la literatura, además del ya comentado modelo original de red neuronal convolucional *U-Net Ronneberger et al. (2015)*. Los primeros a destacar serían aquéllos basados en modificaciones de *U-Net* con cotas de evaluación superiores. Ejemplos de ello son:

- Yu et al. (2021) propone un estudio comparativo entre el modelo *U-Net* original, *U-Net* con *Res-Path* (inspirado en *ResNet)* y su propia propuesta, *RCU-NET*.
- En una línea parecida, Byra et al. (2020) presenta un modelo de *U-Net* modificada: *SK-UNet*.
- Otro ejemplo similar sería Punn and Agarwal (2021) y su propuesta *RCA-IUNet*, una modificación mediante una técnica denominada "*residual cross spatial attention guided inception*", capas de *pooling* híbrido y conexiones puente (*skip connections*) adicionales.

- En Ning et al. (2021), proponen *SMU-Net*, con fondo saliente.

Otras propuestas ponen el foco en apuntalar los centroides de la masa tumoral en la imagen o, por otro lado, crear las denominadas regiones de interés (*ROI's, Regions Of Interest*), un elemento intermedio que ayuda a la detección de la masa tumoral; se trata de cajas, o regiones, en la imagen que delimitan la masa tumoral y complementan en la decisión final de segmentación. Un ejemplo de ello puede leerse en Yap et al. (2018) que, además, incluye una evaluación y comparativa de varios modelos (*UNet, LeNet, FCN-AlexNet*). Otro caso de estudio se encuentran en Ragab et al. (2019), en el que se combina la extracción de características, mediante una red neuronal profunda, con un clasificador *SVM*.

Mención adicional, en lo tocante a la sola tarea de clasificación de imágenes *BUS*, merecen ciertos casos de éxito. *Shi et al. (2016)* detalla una implementación de red neuronal polinomial apilada con buenos resultados incluso usando pequeños *datasets*; variantes en el preprocesado y el uso de *transfer learning* también han demostrado una alta eficacia en Byra et al. (2019); en la línea del *transfer learning*, Eroğlu et al. (2021) propone extracción de características mediante una arquitectura híbrida que combina modelos populares de redes neuronales (*AlexNet, MobileNetV2, ResNet50*) con algoritmos *ML* para clasificación (*SVM*'s y *KNN*'s).

Y respecto de aquellos trabajos que han sido capaces de abordar ambas tareas, clasificación y segmentación, al mismo tiempo, también existen contribuciones reseñables, como Mejri et al. (2021), que propone un interesante *pipeline* a modo de detector 2-etapas y evaluación y comparación de varios modelos populares (*VGG, U-Net, Mask-RCNN*...); también Yap et al. (2020), con una propuesta de arquitectura *Faster R-CNN*, *transfer learning* y un preprocesamiento y conversión de las *BUSI's* a formato RGB (3 canales de input) para lograr mejores resultados.

Otra aportación es Dong et al. (2015), aunque éste hace uso de algoritmos de bosques aleatorios, *SVM*'s y la muestra de datos son mamografías (rayos X) en lugar de *BUSI's*.

2.3. Aprendizaje federado

El aprendizaje federado es una metodología que surge como respuesta a la descentralización, la dispersión, la estanqueidad y la privacidad de los datos. Una solución que el Aprendizaje Automático clásico, o estándar, no es capaz de ofrecer. Sobre todo, en lo que respecta a la fase de entrenamiento.

Aunque de recién alumbramiento, el aprendizaje federado realmente arrastra décadas de desarrollo e investigación en tareas como el análisis o el aprendizaje de datos masivamente distribuidos, en campos tan dispares como la criptografía, las bases de datos o el Aprendizaje Automático que nos ocupa Kairouz et al. (2019). Pero cuando realmente el aprendizaje federado toma entidad propia como tal es en contextos con características y retos concretos, como detallaremos en profundidad.

Aquí es importante incidir en una característica del aprendizaje federado y es que, como tal, no representa una implementación concreta, ni siquiera un tipo concreto de algoritmo de Apren-

dizaje Automático. Hablamos de un paradigma que trata de resolver los mismos problemas que el Aprendizaje Automático clásico, solo que en un contexto bien distinto.

Evidentemente, sigue tratándose de Aprendizaje Automático y, por ello, comparte todos los fundamentos teóricos de éste. Las diferencias estribarán en 1) las implementaciones concretas y, quizás lo más importante, 2) los retos y dificultades que emanan en contextos federados Kairouz et al. (2019), como veremos en la subsección 2.3.4.

2.3.1. Cómo es el proceso de aprendizaje federado

El aprendizaje federado es un proceso iterativo, compuesto por rondas de entrenamiento. En el estado inicial del entrenamiento, el modelo (sus parámetros) es inicializado y enviado a algunos o todos los posibles participantes en el entrenamiento; un único modelo (el del servidor) se replica y se copia a todos los entes que vayan a colaborar en esta ronda de entrenamiento. Una vez allí, en cada cliente, cada modelo ejecutará una ronda de aprendizaje clásico. Trazando un paralelismo con el entrenamiento estándar de Aprendizaje Automático, esta ronda *federada* viene a corresponder con una iteración o etapa clásica, en tanto en cuanto el modelo (local, en cada cliente) trata de minimizar una función de pérdida.

Cuando finaliza la ronda federada, cada ente participante devolverá su versión entrenada del modelo (los pesos) a la matriz. Posteriormente, en la matriz, se agregarán los entrenamientos individuales recibidos de manera que se actualice el modelo único. Lo descrito supondría la ronda inicial de entrenamiento.

Posteriormente, este modelo mejorado se trasladará de nuevo a los participantes pertinentes para comenzar una siguiente ronda de aprendizaje federado. Este ciclo se repetirá tantas veces se requiera con el fin de obtener un modelo capaz y eficaz.

Este sería el ciclo básico de aprendizaje federado. Y decimos básico porque existen multitud de variantes y modificaciones al proceso descrito que tratan de mejorar los resultados aplicados a situaciones concretas, incluso ideados para modelos ML concretos. La figura 2.7 representa, en síntesis, el proceso de aprendizaje federado.

2.3.2. Agregación de aprendizajes locales

La consolidación en el servidor de los aprendizajes surgidos en cada ronda de comunicación en los clientes hacia un único modelo matriz es un aspecto fundamental del entrenamiento federado.

Y es que, efectivamente, es altamente probable que los aprendizajes que ocurren en cada ente participante sean distintos unos de otros, ya que cada cual presenta datos en forma, volumen y características propias. Dicho de otro modo, el modelo (sus parámetros) evolucionará de manera particular y acorde a los datos con los que participa en cada ente de la federación, especialmente en situaciones con datos no independientes ni idénticamente distribuidos (*non-IID data*).

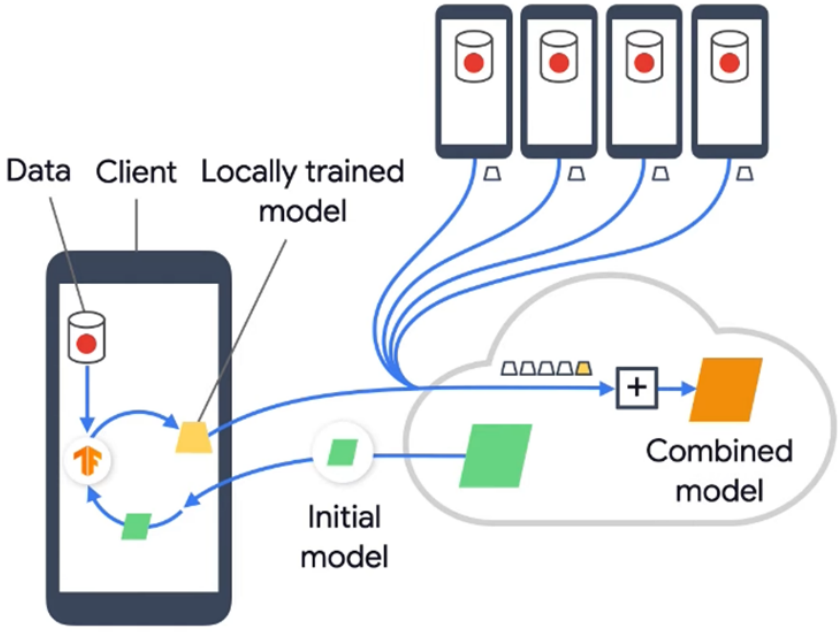

Figura 2.7: Descripción básica del entrenamiento federado Liu (2021)

Por tanto, es necesaria esa tarea de consolidación, de crear consenso entre los aprendizajes y unificarlos en un único modelo que pueda ofrecer la mejor respuesta posible a todos ellos. En el argot federado, esta tarea se conoce como *agregación*. Existen multitud de variantes y estrategias para acometer la agregación, como veremos en la subsección 2.3.5. La figura 2.8 destaca la importancia de la agregación en el proceso federado.

2.3.3. Tipos de aprendizaje federado

Identificamos dos tipos principales de federación Kairouz et al. (2019): *cross-device* y *cross-silo*. Sus similitudes y diferencias Kairouz et al. (2019) se recogen en la tabla 2.1.

A modo de resumen, diremos que en la federación *cross-device* los entes participantes son, comúnmente, dispositivos móviles (*smartphones*, por ejemplo), a los que también se suele referir como "clientes". Un ejemplo real sería la aplicación *Gboard* de Google comentada en la subsección 1.1.2. En la figura 2.9 puede visualizarse este método.

Por otro lado, en la federación *cross-silo*, los entes representarán organizaciones o instituciones; en definitiva, organismos más complejos y grandes que un dispositivo portátil personal. La figura 2.10 muestra el esquema típico de este contexto federado.

Este último es el caso que más se relaciona con el presente trabajo pues, en la práctica, los datos de que disponemos residirían al amparo de instituciones médicas, las cuales participan a modo de *clientes*.

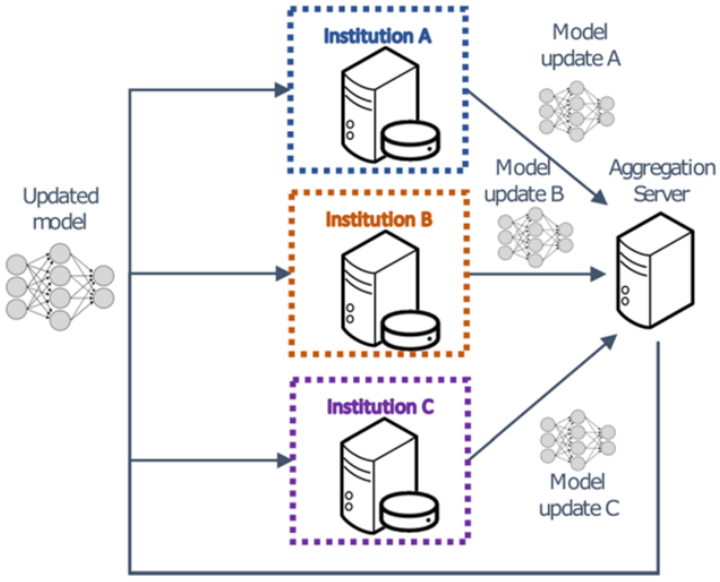

Figura 2.8: Esquema básico del ciclo de entrenamiento federado participado por instituciones Sheller et al. (2020)

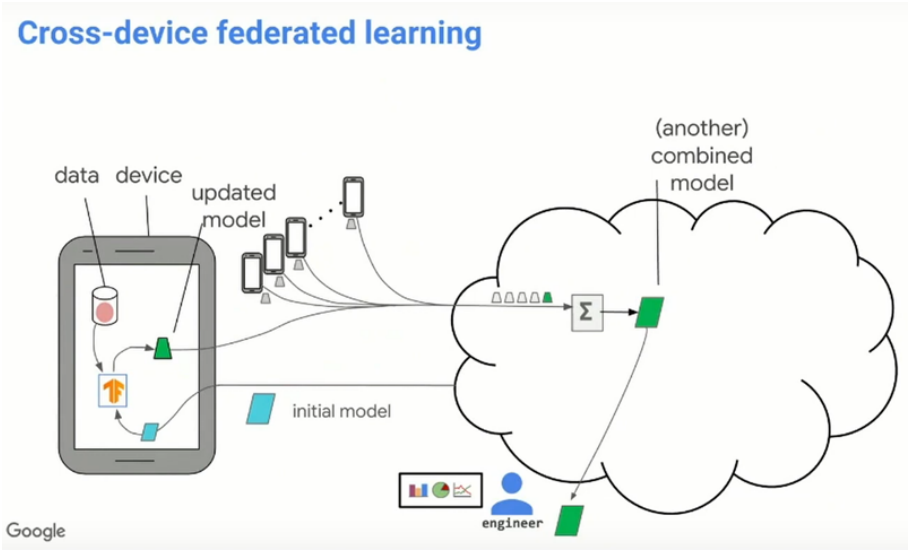

Figura 2.9: Esquema de federación *cross-device* OpenMined (2020)

	Aprendizaje federado	
	Cross-silo	***Cross-device***
Descripción general	Entrenamiento de datos en silos. Los clientes son organizaciones o centros de datos distribuidos.	Entrenamiento con un gran número de clientes, lo cuales son dispositivos móviles o *IoT*.
Coordinación	Un servidor o servicio central controla el proceso de aprendizaje (pero nunca tiene visibilidad sobre los datos).	
Comunicación	Topología cliente-servidor, con un hub central que coordina el aprendizaje y comunicación múltiple con clientes.	
Distribución de los datos	Datos no independientes o idénticamente distribuidos. Los datos se generan a nivel local y permanecen descentralizados y aislados: cada cliente solo dispone de sus datos, sin acceso a los de otros clientes	
Escala de la distribución	2 o más clientes.	Masivo, del orden de miles de millones.
Disponibilidad de los datos	Todos los clientes suelen estar disponibles siempre, durante todo el proceso de aprendizaje.	Normalmente, solo una parte de los clientes estará disponible de entrenamiento.
Identificación de los clientes	Cada cliente posee un identificador que el servicio coordinador reconoce y usa en consecuencia.	Los clientes no poseen identificador.
Cuello de botella principal	Puede ser computacional o de comunicación.	Depende de la tarea, pero suele ser la comunicación: tipos de conexiones (inalámbricas), de baja calidad, etcétera.
Trazabilidad del aprendizaje (statefulness)	Cada cliente puede acumular el estado o progreso de su aprendizaje entre rondas de entrenamiento.	*Stateless*: cada cliente puede participar escasamente al proceso global de aprendizaje (por ejemplo, en una sola ronda), por lo que en cada ronda se asume participantes completamente nuevos.
Ratio de fallos de los clientes	Los clientes suelen presentar una tasa relativamente baja de fallos.	Credibilidad de los clientes altamente comprometida. Se espera una tasa de fallo de 5 % o más de los clientes participantes por ronda (problemas de conexión, batería, capacidad de procesamiento local...).

Tabla 2.1: Diferencias entre aprendizajes federados *cross-silo* y *cross-device*

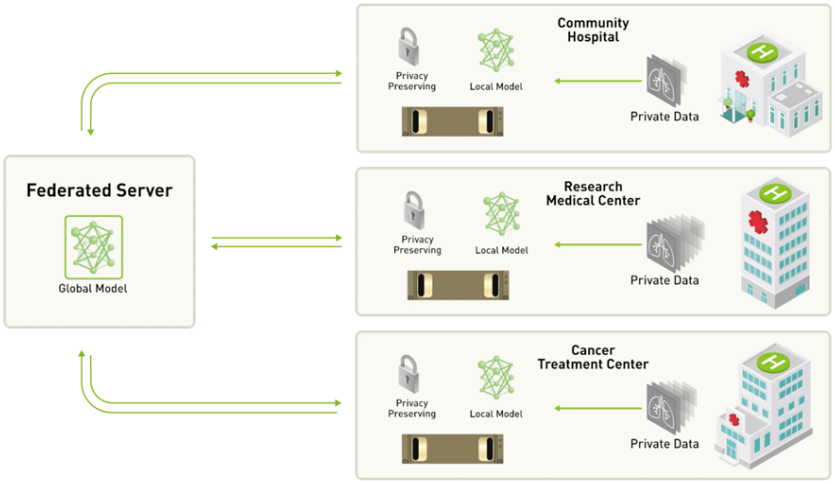

Figura 2.10: Esquema de federación *cross-silo* NVidia (2019)

2.3.4. Retos en el proceso de aprendizaje

El aprendizaje federado se postula como solución a los condicionantes de estanqueidad, dispersión y privacidad de los datos, sí, pero esos mismos condicionantes van a generar nuevos retos con los que lidiar, propios de la naturaleza del contexto al que se pretende aplicar. A destacar Kairouz et al. (2019):

- **Datos no independientes y/o no idénticamente distribuidos (*non-IID data*)**: los datos pueden ser heterogéneos, pues cada cliente puede contener información particular, no representativa de la distribución general. Un ejemplo de ello se daría al abordar un aprendizaje federado de reconocimiento de texto escrito, en el que cada cliente fuera una persona, cada cual con su expresión caligráfica única. Datos *non-IID* pueden presentar también diferencias en cuanto al criterio de etiquetado de los datos; la cantidad de datos alojados en cada cliente también puede variar significativamente. En suma, modelos entrenados en estos contextos pueden carecer de generalización suficiente al aplicarse a dominios nuevos.

- **Datos no balanceados**: en los datos locales pueden predominar ciertas clases de datos, mientras otras apenas tengan escasa o nula representación. Algo bastante frecuente en el ámbito oncológico, por ejemplo.

- **Datos distribuidos masivamente**: el número de participantes de la federación puede ser abrumador, del orden de millones, en contextos *cross-device*.

- **Comunicación limitada**: también en contextos *cross-device*, la participación de los clientes (ej.: dispositivos móviles) estará sujeta al acceso a una conexión a Internet y dicha participación podría o no ser voluntaria; por ejemplo, cuando la participación federada solo sea posible si el dispositivo se encuentra en modo inactivo o *idle*. La comunicación tampoco será simultánea

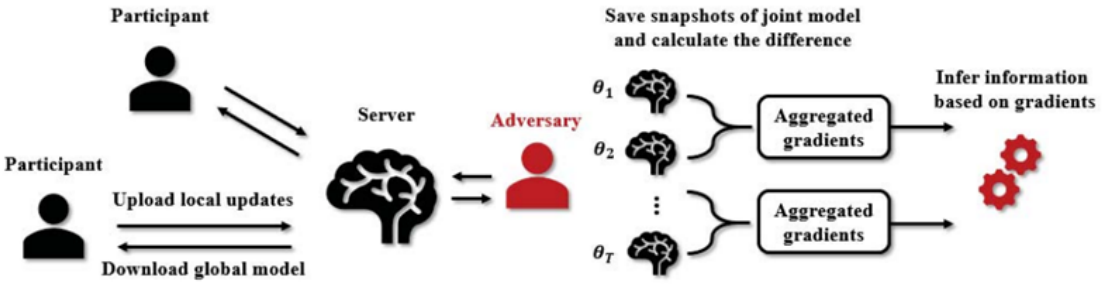

Figura 2.11: Ataques inferenciales en aprendizaje federado, aparece en Melis et al. (2019)

cuando los dispositivos disten largas distancias geográficas o se encuentren en zonas horarias muy alejadas entre sí.

Estos son los retos propios más destacables del aprendizaje federado, pero también se enfrenta a otros relacionados con la seguridad y la privacidad de los datos Kaissis et al. (2020); Liu et al. (2021); Dwork and Roth (2013); Dwork (2011), lo cual es de especial interés en el caso médico que nos ocupa:

- **Privacidad diferencial** (o la no trazabilidad de los datos) Abadi et al. (2016); Wang et al. (2020a): nada de lo aprendido por un modelo de aprendizaje federado puede servir como traza o pista para la identificación de un caso o conjunto de casos clínicos concretos. La privacidad diferencial (*differential privacy, DP*) es la técnica que procura proteger los datos de ataques inferenciales, como describe la figura 2.11.
- **Anonimización**: todos los registros contenidos en los conjuntos de datos deben preservar el anonimato. No debe existir manera de identificar un elemento concreto del conjunto de datos.
- **Encriptación**: de manera más genérica, se debe proteger el aprendizaje de posibles ataques mediante el cifrado de las comunicaciones y los sistemas de almacenamiento de los datos.
- **Escalabilidad**: el rendimiento de todo sistema de aprendizaje federado debe de mantenerse cuando se produzca un crecimiento de este, ya sea al aumentar el número de clientes, el tamaño de los datasets, la complejidad de los *pipelines* cliente/servidor, la capacidad computacional de los entes implicados, etcétera..

2.3.5. Algoritmos de aprendizaje federado

En la introducción del presente capítulo, subsección 2.3.1, describimos el ciclo básico del aprendizaje federado. El algoritmo que implementa tal proceso originario fue propuesto por McMahan et al. (2016) y acuñado con el título de *Federated Average (FedAvg)*. *FedAvg* arranca con una inicialización aleatoria del modelo (sus pesos) y aplica el descenso estocástico de gradiente (*SGD*, *stochastic gradient descent*) en cada ronda de entrenamiento local, en cada cliente. Para la agregación, el cálculo

se basa en una media de pesos proporcional a la cantidad de datos disponible en cada participante (*weighted sum*). Este algoritmo ha demostrado ser una opción robusta frente a ciertos conjuntos de datos *non-IID*, así como ser capaz de reducir el tiempo de entrenamiento en contextos altamente descentralizados, habiendo demostrado convergencias más rápidas, entre 10 y 100 veces, respecto de implementaciones sencillas de *SGD*'s Google (2017).

Sin embargo, dada su naturaleza simple o ingenua, puede no ofrecer un desempeño igual de idóneo con todo tipo de distribuciones de datos *non-IID*, ya que su forma de agregación de pesos para valores muy dispares puede resultar en agregaciones con poca generalidad. Aún así, tal es su relevancia que *FedAvg* es la opción de agregación por defecto en la librería *TensorFlow Federated* (y la única, hasta la versión 0.19 inclusive).

Tras *FedAvg*, han surgido y continúan apareciendo nuevos algoritmos y técnicas que introducen variaciones en uno o varios de los aspectos clave del ciclo federado: el cálculo de agregación de aprendizajes locales, el proceso concreto de aprendizaje local en los clientes, la selección de los clientes participantes en cada ronda de comunicación, la inicialización del modelo a entrenar, cambios dinámicos durante la fase de entrenamiento, etcétera. A continuación, se enumeran algunos de los más destacables. Todos ellos han sido propuestos a través de publicaciones y testados empíricamente.

El algoritmo *FedProx,* Li et al. (2018), introduce una regularización que penaliza grandes cambios en los pesos del modelo provenientes de clientes *outliers*, lo cual limita la posibilidad de que el algoritmo deambule innecesariamente durante la fase de entrenamiento.

Otro algoritmo similar a *FedAvg* es *FedPer*, propuesto en Arivazhagan et al. (2019), en cuanto que computa de manera similar la agregación de pesos locales, pero con matices; en el caso de *FedPer*, la agregación no incluye las capas superiores de la red neuronal. El objetivo es consensuar únicamente aquellas capas que, teóricamente, configuran las representaciones inherentes a los datos. Puede entenderse como un *FedAvg* con *transfer learning*.

Otras variantes populares son *q-FedAvg* o *q-FFL* (*q-Fair Federated*), Li et al. (2019a), y *per-FedAvg,* Fallah et al. (2020). El primero introduce una regularización que otorga mayor importancia a los participantes con peor desempeño, incentivando al modelo a centrar la atención en esos aprendizajes. En términos de cómputo de la función de pérdida global, los clientes con valores de función de pérdida local altos serán asignados un peso mayor. Un término q de calibración controlará ese aporte de peso en el cálculo de la función de pérdida del servidor. De hecho, este algoritmo puede considerarse la generalización de otro, *AFL* (*Agnostic Federated Learning*) en Mohri et al. (2019), cuando el parámetro $q = 0$. En el caso de *per-FedAvg*, éste permite un aprendizaje personalizado para cada participante una vez cumplidas varias iteraciones iniciales del entrenamiento, lo que deriva en una suerte de aprendizaje federado multitarea.

También destaca *FedMA* (*federated matched averaging), Wang et al. (2020b)*, que aplica dinámicamente modificaciones a la estructura del modelo de red neuronal, fusionando elementos internos de la red que pueden ser considerados redundantes. Para ello, aplica la técnica probabilística bayesiana *PFNM* (*Probabilistic Federated Neural Matching*, Yurochkin et al. (2019)). Dada su especificidad,

FedMA está concebido para usarse únicamente con arquitecturas convolucionales sencillas. También permite reducir el tiempo del entrenamiento federado al minimizar la comunicación cliente-servidor.

Una técnica parecida a *FedMA* se encuentra en *FedDist* (*federated distance*), Ek et al. (2021). *FedDist* también aplica modificaciones dinámicas al modelo de red neuronal, pero, en su caso, basado en un cálculo de distancia euclídea de los pesos locales; aquellas neuronas (elementos de la red neuronal) muy distantes, es decir, con pesos muy dispares, no serán consolidadas y serán añadidas al modelo agregado bajo una capa específica. Esto favorece la convergencia y la generalización de las sub-estructuras comunes dentro del modelo.

También existen combinaciones de algoritmos, como la representada por *FedVC* y *FedIR* de Hsu et al. (2020). Dos técnicas que trabajan conjuntas, ideadas mediante simulaciones en problemas de clasificación con grandes conjuntos de datos *non-IID*, dispersos en multitud de clientes, del orden de miles. Entre sus modificaciones al algoritmo *FedAvg*, destacan la selección de clientes en cada ronda de federación (la probabilidad depende del tamaño del *dataset* en cada cliente), el número de iteraciones de entrenamiento locales, la creación de clientes "virtuales" (partición dinámica de un cliente grande en varios más pequeños) y un recálculo de pesos proporcional al tamaño del *dataset* local. Un algoritmo parecido a la combinación anterior, en cierto modo, es *FedMed, Wu et al. (2020)*, que implementa una agregación voluble y dinámica durante el entrenamiento, vía un *mediador*, junto con un proceso selectivo concreto de clientes.

En cuanto a opciones que exploran o están muy en línea con la privacidad diferencial, podemos destacar a *NbAFL* (*noising before model aggregation FL*), propuesto en Wei et al. (2020). Por último, entre los algoritmos que ponen el foco en asegurar la comunicación efectiva cliente-servidor, mencionamos *RFA* (*Robust Federated Averaging*) de Pillutla et al. (2019).

2.3.6. Recursos y herramientas

Existen librerías de programación que implementan el paradigma del aprendizaje federado. Una comparativa entre varias de ellas puede verse en la figura 2.12. Una de las más importantes y objeto de análisis en el presente trabajo es *TensorFlow Federated*, *TFF*, de Google (2022a). *TensorFlow Federate*d nació en el año 2018 y continúa su desarrollo dentro de la compañía Google y es apoyada por la comunidad investigadora. Actualmente, el último *release* estable corresponde a la versión *0.19.0*. Una de las grandes virtudes de *TFF* es que permite realizar simulaciones federadas bajo un único entorno computacional, lo cual resulta de gran utilidad para pruebas conceptuales, ya que no existe necesidad de desplegar o tener acceso a distintas máquinas, configurar entornos de red, gestionar accesos, etcétera. A tales fines, la versión actual de *TFF* ofrece, *out-of-the-box*, funciones encapsuladas que facilitan enormemente la tarea de desarrollo de código. Ajustándose a las funcionalidades básicas, se puede construir un proceso federado en pocas líneas de código. Otras librerías destacables son *PySyft* de OpenMined (2022), Flower (2022), FedML (2022), *FedLab* de Zeng et al. (2021), FedJAX (2022) e *EasyFL*, propuesto en Zhuang et al. (2021).

TABLE I
COMPARE OUR PROPOSED EASYFL WITH EXISTING FL LIBRARIES, PLATFORMS, AND FRAMEWORKS. EASYFL REQUIRES LITTLE CODING AND PROVIDES MORE OUT-OF-THE-BOX FUNCTIONALITIES TO IMPROVE THE PRODUCTIVITY OF RESEARCHERS. ◯ MEANS LIMITED SUPPORT.

	LEAF [1]	PySyft [2]	PaddleFL [3]	TFF [4]	FATE [5]	EasyFL (Ours)
Lines of Code (Vanilla FL App)	∼400	∼190	∼190	∼30	∼100	3
Heterogeneity Simulation	◯	×	◯	◯	◯	√
Training Flow Abstraction	×	×	×	×	×	√
Distributed Training Optimization	×	×	×	×	×	√
Tracking	×	×	×	◯	√	√
Deployability	×	×	√	◯	√	√

Figura 2.12: Comparativa de librerías y *frameworks* de aprendizaje federado - Table I en Zhuang et al. (2021)

Algunas de estas librerías, tales como *TensorFlow Federated* y *FedML*, representan *frameworks* completos que también incluyen utilidades de *benchmarking* como son *datasets* específicamente preparados para entrenar y evaluar modelos federados. El primero se apoya principalmente en la librería primaria *tensorflow*, mientras que *FedML* se fundamenta en la librería *PyTorch*. Otro *framework* destacable es Leaf (2022), aunque éste no cuenta con librería específica que implemente la federación. Un ejemplo de plataforma que provee específicamente servicios web para entrenamiento federado es *Acuratio Acuratio (2022)*. El paradigma también puede implementarse *ad hoc*; usando meramente como librería base *TensorFlow* o *PyTorch* puede resultar suficiente para realizar simulaciones locales o conceptuales.

El aprendizaje federado es un área de investigación activa que en el presente genera numerosas contribuciones entre la comunidad investigadora, y no solo aplicado al ámbito médico, sino también a la economía, al *blockchain*, etcétera. La web comienza a recoger y organizar esos trabajos bajo puntos de acceso centralizados, lo cual facilita la propia labor de investigación, así como su conocimiento y difusión. Dos ejemplos de ello son los repositorios *github* "*Awesome Federated Learning*", He (2021), y "*TensorFlow Federated Research*", en Google (2022b). El primero representa un glosario completísimo en todas las áreas que está abarcando el aprendizaje federado: publicaciones, código de programación, ámbitos de aplicación, etcétera; el segundo recoge desarrollos en lenguaje Python con la librería *TFF* de Google.

Asimismo, empiezan a aparecer iniciativas para canalizar esfuerzos en cuanto a formalizar el aprendizaje federado, como "*The Open Brain Consent*", Bannier et al. (2021), y otras directamente encaminadas a promocionar el entrenamiento federado y la disponibilidad de *datasets* públicos como *The Federated Tumor Segmentation (FeTS) initiative, FeTS (2022),* y su pionera competición *Federated Tumor Segmentation Challenge 2021*, FeTSChallenge (2022).

2.3.7. Casos de éxito

Es imperativo volver a destacar que, durante la elaboración del presente proyecto, no se encontraron trabajos que englobasen la segmentación y/o clasificación de imágenes *BUS* mediante Aprendizaje Automático federado en su totalidad. Aunque sí son de reseñar ciertas publicaciones que, de una manera u otra, aportan en la vía ilustrada en el presente proyecto sobre el aprendizaje

federado.

Sheller et al. (2019) propone un enfoque integral federado para la segmentación de tumores cerebrales resolviendo con un modelo *U-Net*. *Li et al. (2019b)* también aborda la tarea federada de segmentación de tumores cerebrales; sin especificar la arquitectura del modelo, implementa un algoritmo con preservación de la privacidad de los datos. En Siddhesh (2021) también se opta por *U-Net* en un entorno federado simulado con la librería *PySyft*, en este caso, aplicado a imágenes neuro-oncológicas RGB. Li et al. (2020) propone una técnica más novedosa, implementando una red *GAN* (*Generative Adversarial Network*), para la segmentación de imágenes volumétricas (3D) de rayos X en un contexto federado. En *Hsu et al. (2020)* también se propone un ejemplo integral. En este caso, se hace explícito el uso de *TensorFlow Federated*, aunque no aplicado al campo médico (grandes *datasets* genéricos) y se simulan condiciones de datos *non-IID* y se testan distintos algoritmos de agregación federada (existentes y propios). Vepakomma et al. (2018) va un paso más allá, proponiendo un modelo de federación modificado (*SplitNN*), pensado para aplicar en contextos médicos y ofrece una comparativa de resultados, incluidos de rendimiento computacional, con implementaciones clásica y federada pura.

Capítulo 3

Materiales y métodos

En este capítulo, expondremos las fases a través de las que se completó el presente proyecto y detallaremos los recursos y técnicas concretas empleadas en la consecución del mismo.

3.1. Descripción general

El proyecto ha constado de las siguientes etapas:

1. Aproximación al problema a abordar y resolver desde la perspectiva del Aprendizaje Automático: clasificación, segmentación de imágenes, detección de objetos, combinación de las anteriores.
2. Revisión del estado de la cuestión.
3. Exploración de las fuentes de datos disponibles.
4. *Pipeline* clásico: creación de un modelo *Deep Learning* bajo la metodología clásica (datos unificados y centralizados). Tres bloques dentro del *pipelines* han sido objeto de exhaustiva investigación: el preprocesado, la implementación concreta de *U-Net* y la parametrización del entrenamiento. En las secciones 3.3, 3.4 y 3.5 se encontrarán los detalles concretos testados.
5. *Pipeline* federado: creación de modelos *Deep Learning* simulando un entorno de características federadas en el que se replican (y adaptan, cuando ha sido necesario) todos los procesos del *pipeline* clásico posibles. El objetivo es poder comparar modelos fundamentalmente similares en cuanto a la consecución de resultados.
6. Preparación y comparación de resultados de los modelos producidos en los *pipelines* anteriores.
7. Elaboración de conclusiones, autocrítica al proyecto y futuras mejoras.

Las fases 4) y 5), de definición de los *pipelines*, han sido objeto de constantes iteraciones en busca de una mejora de los resultados finales y de la calidad del proceso en sí mismo.

3.2. Entorno computacional y tecnología

El proyecto ha sido desarrollado enteramente en lenguaje Python en notebooks Jupyter. Las principales librerías empleadas para la creación de las redes neuronales y los aprendizajes clásico y federado han sido *TensorFlow* y *TensorFlow Federated*.

En el caso del aprendizaje clásico, se ha hecho uso de *TensorFlow* en versión 2.7.0; para el aprendizaje federado, se ha empleado la última versión estable hasta la fecha de *TensorFlow Federated* (0.19.0), la cual requiere y depende de versiones anteriores de otras librerías, como *TensorFlow* 2.5.0.

El proyecto ha sido posible gracias a la cesión por parte de la UNED de acceso y uso de una máquina virtual con alta capacidad de procesamiento. A continuación, se exponen sus características más destacables.

3.2.1. Recursos hardware

- Servidor acelerador de alto rendimiento: ASUS ESC4000 G4
- Procesadores (CPUs): 40 cores a 2.2GHz (= 2x Intel Xeon Silver 4210)
- Memoria RAM 96GB (= 12x Samsung DDR4 2666 8GB ECC Reg RDIMM)
- GPUs: 2x NVIDIA V100 16GB HBM2 PCIe (640 Tensor Cores + 5120 CUDA Cores)
  2x NVIDIA V100 32GB HBM2 PCIe (640 Tensor Cores + 5120 CUDA Cores)
- Discos: 1x SSD 1TB (Lect. 3400MB/s, Escr. 2500MB/s) 1x HDD 4TB (Lect./Escr. 194MB/s)
- Acceso por red vía VPN de la UNED y sesión SSH.

3.2.2. Recursos software

- Sistema operativo GNU/Linux: Debian-11 "bullseye" (stable)
- Drivers GPU Nvidia versión 460, librerías Nvidia CUDA versión 11.2, librerías cuDNN versión 8.1.1, OpenCL versión 2.2, LibTorch versión 1.8.0, GCC/G++: versión 9 (la última por omisión)
- Python: versión 3.9.2
- R: versión 4.0.4

3.3. *Datasets*

Se ha trabajado con tres conjuntos de imágenes *BUS* distintos. Dada la dificultad de encontrar datasets anotados y en abierto, solo hemos podido disponer de tres datasets: Al-Dhabyani et al.

	DATASETS			
Característica	**A**	**B**	**C**	**unificados**
Fuente	***Al-Dhabyani et al. (2020)***	***Yap et al. (2018)***	***Xian et al. (2018)***	
Formato imagen BUS	.png	.png	.png	-
Formato imagen máscara (GT)	.png	.png	.bmp	-
Número de imágenes BUS	780	163	562	1505
Tamaño imágenes uniforme	NO	NO	NO	NO
Tamaño imágenes promedio (ancho x alto)	616 x 501	538 x 455	510 x 402	568 x 549
Ratio pixeles tumor / no-tumor	0.03 / 0.97	0.05 / 0.95	0.11 / 0.89	0.08 / 0.92
Clasificación de tumores disponibles	SI	SI	NO	NO
Etiquetas/Máscaras (nº imágenes por clase)	benign (437) normal (133) malignant (210)	benign (109) malignant (54)	unclassified (562)	-

Tabla 3.1: Resumen descriptivo de los *datasets*

(2020), Yap et al. (2018) y *Xian et al. (2018)*. La tabla 3.1 recoge sus características más relavantes. Para facilitar la comprensión, se han denominado datasets "A", "B" y "C", respectivamente.

Es necesario remarcar que los *datasets* A y C fueron particionados de diversas maneras con el fin de ampliar y enriquecer la evaluación federada. Recordemos que el aprendizaje federado se basa en entrenar con datos descentralizados y aislados en entes (clientes/silos) estancos entre sí. Por tanto, cada *dataset* será asociado a un cliente. Cuantos más datasets disponibles, mayor número de clientes federados podremos configurar. En nuestro caso, se ha optado por particiones en mitades (50/50) o tercios (˜33/33/33), cuando así ha sido posible. Detalles del proceso divisivo se encuentran en la sección que describe el preprocesamiento, 3.5 y en la tabla 3.2.

A destacar, sobre todo, el alto desbalanceo en la ratio píxeles *tumor/no-tumor* (0,08 / 0,92), y que presenta uno de los retos del problema a abordar. Estas cifras provienen de la proporción de píxeles en las máscaras *ground-truth*. Como ejemplo, la figura 3.1.

Respecto de las imágenes *BUS*, los histogramas de la figura 3.2 muestran gran cantidad de píxeles a valor 0 (en la escala de grises, corresponde a negro puro). Ese valor no presenta una biyección contra la máscara *ground-truth*. Dicho de otro modo, un píxel *BUS* negro puro no puede asociarse directamente a una clasificación, bien sea de masa tumoral o de tejido sano o normal.

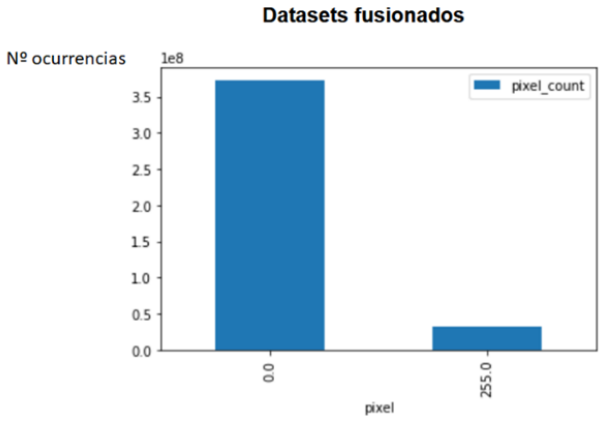

Figura 3.1: Desbalanceo de clases que presentan los *datasets* (unificados). Píxeles *no-tumor (0) / tumor (255)*

3.4. Arquitectura de red neuronal

Como ya se ha comentado, la arquitectura escogida es *U-Net* Ronneberger et al. (2015). La implementación específica se inspira en varios ejemplos concretos de implementación aplicados a segmentación de imágenes (véase Bnsreenu (2021); Kaggle (2021); StackOverflow (2017)). Sus aspectos más destacables son los siguientes:

- Tamaños de entrada/salida: 128 x 128 píxeles.
- Cuatro (4) niveles de covolución o contracción: comenzando con sesenta y cuatro (64) filtros en la primera capa convolutiva, doblando el número hasta la cuarta capa, con quinientos doce (512) filtros. En todas, se aplica un filtro convolutivo de tamaño 3x3, con desplazamiento unitario 1x1, función de activación *ReLu* y *padding* (*same*) que no altera el tamaño de la imagen, a diferencia de la propuesta original.
- Capas de normalización *dropout* (con ratio = 0,2) y agrupación *max-pooling* 2x2 entre capas convolutivas.
- Una (1) última de convolución con mil veinticuatro (1024) filtros, con capa de normalización dropout pero sin agrupación posterior.
- Cuatro (4) capas de convolución transpuesta o expansión, con las que se reduce gradualmente el número de filtros, hasta los sesenta y cuatro (64) originales. Además, cada capa se concatena con parte de la capa homólogas convolutiva y se aplica un filtro de convolución traspuesta de tamaño 2x2 y desplazamiento 2x2*;* la función de activación es *ReLu* y *padding* con valor *same.*
- Finalmente, la salida se consigue con una capa convolutiva de dos (2) filtros, con desplazamiento unitario, 1x1, y función de activación sigmoide o *sigmoid*.

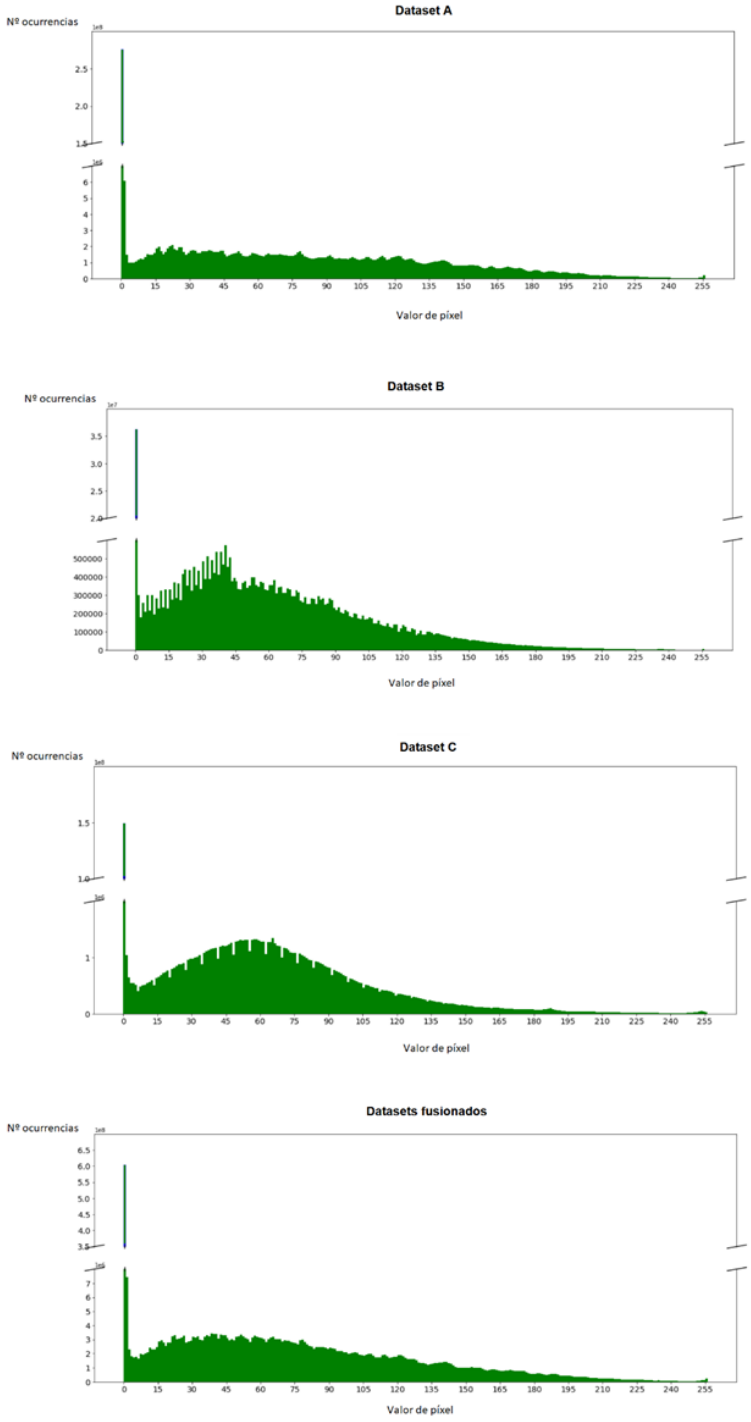

Figura 3.2: Distribución de píxeles en los *datasets*: A (primera figura), B (segunda), C (tercera) y unificados (última)

3.5. Preprocesado

3.5.1. Directorio de ficheros

Primeramente, cada *dataset* fue reestructurado en directorios de trabajo separados y organizados. Los ficheros de imagen (tanto *BUS* como *ground truth*) fueron renombrados cuando fue conveniente de cara a facilitar la ingesta de datos por los *pipelines* de entrenamiento.

La primera fase de preprocesado consiste en crear una segunda estructura de directorios en la que:

1. Se escoge uno o varios *datasets* objetivo aprocesar.
2. Opcionalmente, se hace una partición del *dataset* escogido: en mitades (50/50) o tercios (˜33/33/33).
3. Se realiza una división aleatoria y proporcional sobre el conjunto de datos seleccionado, creando tres (3) subdirectorios nuevos, a razón de:
   - 80 % del *dataset* para entrenamiento (*training subset*)
   - 10 % del *dataset* para validación (*validation subset*)
   - 10 % del *dataset* para test *(test subset*)
4. Se clonan los ficheros *BUS* y *GT* bajo los directorios correspondientes.

Este ejercicio de reestructuración (división, clonación y reorganización) también ayudará en la posterior ingesta de datos en el *pipeline* de aprendizaje.

3.5.2. Fusión y despliegue de los *datasets*

En el caso del modelo de aprendizaje *clásico*, necesitamos que todos los *datasets* (3) sean procesados a la vez. Mediante el proceso descrito anteriormente en la subsección 3.5.1, primero se hace una selección de los tres conjuntos (A, B, C) a la vez y se fusionan o combinan bajo un único directorio final que contiene todas las imágenes disponibles para el proyecto. En definitiva, de cara al entrenamiento clásico, se dispone de un único *dataset*.

Por otro lado, para el aprendizaje federado, llevamos a cabo más operaciones. Las figuras 3.3 y 3.4 muestran ejemplos esquemáticos de ello.

Primeramente, cada *dataset* (*A, B, C*) fue tratado individualmente a través del proceso descrito, resultando en sendos directorios individuales con su estructura interna correspondiente, recordemos, *train-valid-test (80-10-10)*.

Posteriormente, para los *datasets A* y *C*, repetimos la operación pero, previo a ello, acometimos divisiones de cada conjunto en proporciones idénticas:

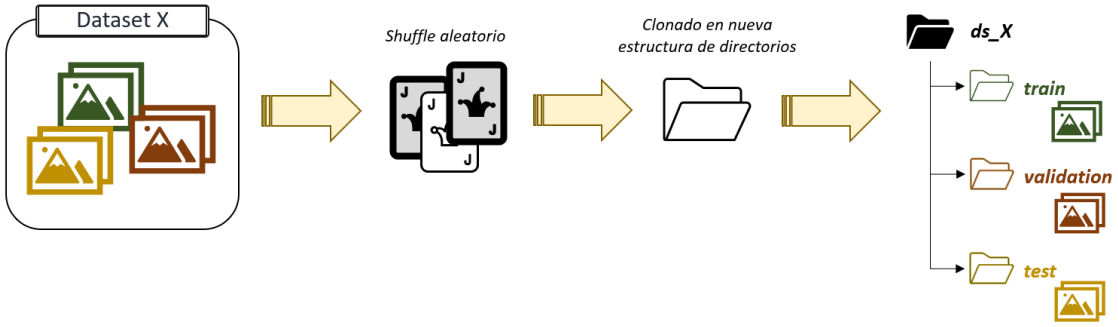

Figura 3.3: Diagrama representativo de las tareas de preprocesado (organización y distribución) de los datasets implicados en los entrenamientos de red neuronal.

- Una división en dos partes, esto es, 50/50, resultando en dos subconjuntos por cada *dataset* principal. Con ello, generamos cuatro (4) nuevos directorios, nombrados A21, A22, C21 y C22.
- Una división en tres partes, tres tercios, resultando en tres subconjuntos por cada *dataset* principalello, generamos seis (6) nuevos directorios, nombrados A31, A32, A33, C31, C32 y C33.

Todos estos subconjuntos replican la estructura de directorios *train-valid-test*, e igualmente con las proporciones descritas (*80-10-10*).

La elección de los datasets *A* y *C* no fue aleatoria; se trata de los conjuntos de mayor volumen en cuanto a número de imágenes. Ambos permitían particionados que produjeran divisiones que, individualmente y en el peor de los casos, fueran de volumen similar al *dataset B*, el de menor tamaño de todos los disponibles.

Los catorce (14) conjuntos finales resultantes (ver tabla 3.2) se usarán acorde a las pruebas programadas (ver sección 4.1).

3.5.3. Transformaciones y aumento de datos

El preprocesado común a todos los conjuntos incluye redimensionamiento de imágenes (tanto imágenes *BUS* como *ground truth*). Para las imágenes *BUS*, se ha aplicado reescalado de los valores de píxel al rango [0, 1] y filtro *CLAHE, Contrast Limited Adaptive Histogram Equalization* Ponraj et al. (2011). Las máscaras no han requerido preprocesado específico. El aumento de datos se ha construido con volteo horizontal (*horizontal flip*).

Además, conviene mencionar pruebas exploratorias, descartadas finalmente para la evaluación final, al no alcanzar tan buenos resultados como el filtro *CLAHE*. Dichas pruebas pretendían ampliar

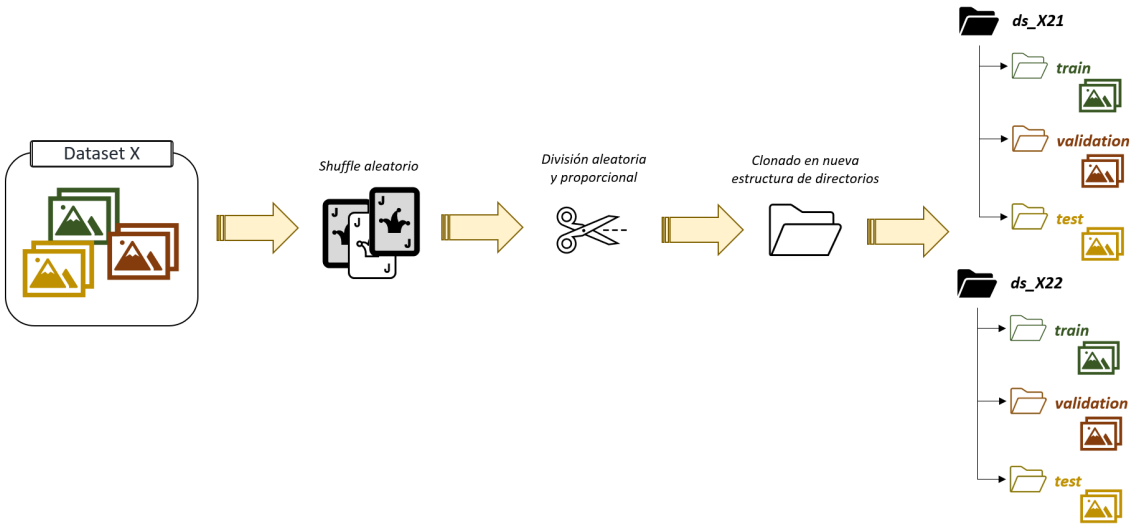

Figura 3.4: Diagrama representativo de las tareas de preprocesado (organización y distribución) de los datasets implicados en los entrenamientos de red neuronal, cuando se particiona un *dataset* original en varios.

Dataset/directorio final	**Dataset original**	**Descripción del dataset / directorio final**
ds_ALL	A + B + C	*Datasets* A, B y C completos
ds_A	A	*Dataset* A completo
ds_B	B	*Dataset* B completo
ds_C	C	*Dataset* C completo
ds_A21	A	Mitades (50 %) aleatorias del *dataset* A
ds_A22	A	
ds_C21	C	Mitades (50 %) aleatorias del *dataset* C
ds_C22	C	
ds_A31	A	Terceras partes (˜33 %) aleatorias del *dataset* A
ds_A32	A	
ds_A33	A	
ds_C31	C	Terceras partes (˜33 %) aleatorias del *dataset* C
ds_C32	C	
ds_C33	C	

Tabla 3.2: Construcción de *datasets* finales de entrenamiento a partir de los originales A, B y C

el aumento de datos (técnicas de *shear*) y reducir el ruido de las imágenes *BUS*, mediante filtros gaussiano y de mediana, detección de bordes y *thresholding* Ponraj et al. (2011).

Es importante destacar que nos vimos forzados a programar el redimensionamiento como primera acción del preprocesado, antes de cualquier otra transformación, debido a una limitación de la librería *TensorFlow Federated*.

3.6. Entrenamiento

3.6.1. Subconjuntos de entrenamiento y validación

En el caso del aprendizaje federado, el conjunto de *training* se usó para el entrenamiento y el *validation subset* ha servido para guiar la optimización de la función de pérdida.

En el caso federado, con la librería *TensorFlow Federated*, el conjunto de *training* guía directamente la optimización de la función de pérdida en cada cliente durante la fase de entrenamiento. Es decir, no hay uso directo de ningún *validation subset*; la librería *TensorFlow Federated* no presenta esa funcionalidad. En nuestro caso, hemos aprovechado este subconjunto de validación meramente como indicativo de si la fase de entrenamiento federado incurría o no en sobreajuste del modelo (*overfitting*). Dicho de otro modo, se trata de una evaluación "al vuelo" tras cada ronda federada de entrenamiento, usando los *validation subsets* de los clientes.

Ambos subconjuntos, *training* y *validation*, se ponen a disposición de la fase de entrenamiento con *shuffling* aleatorio y en lotes de 32 pares de imágenes *BUS-GT*.

El entrenamiento clásico constó de 500 *epochs*; el entrenamiento federado se realizó con 500 rondas.

3.6.2. Modelos federados

Se realizó un total de veintisiete (27) entrenamientos de modelos federados distintos con éxito. Todos comparten el mismo número de rondas de federación (500) pero difieren en el algoritmo o proceso de agregación en el servidor.

La mayoría de ellos comparten el algoritmo de agregación de base (*FedAvg*) pero incorporan diferencias sensibles respecto del proceso agregativo. Los detalles concretos pueden encontrarse en la sección 4.1. Una de ellas implementa el algoritmo *RFA* (*Robust Federated Aggregation*), una variante de *FedAvg* citada en la subsección 2.3.5.

3.7. Evaluación

3.7.1. Función de pérdida

La función de pérdida se basa en el *coeficiente Dice* descrito en la subsección 2.2.2, fórmula 2.1.

Es de destacar que se realizaron pruebas con distintas funciones de pérdida, pero que no resultaron en mejores puntuaciones que las obtenidas con el *coeficiente Dice*. Fueron testados, a modo de función de pérdida, el *coeficiente Jaccard*, la entropía cruzada (*cross entropy*) y *focal loss*, así como dos optimizadores (*SGD* y *Adam* con múltiple parametrización).

3.7.2. Métricas de rendimiento

Para completar la valoración de la bondad del modelo resultante, se hace uso de varias métricas para valorar la bondad del modelo: la exactitud (*accuracy*), el *coeficiente Jaccard* o *Intersection over Union (IoU)* y la sensibilidad (*recall*).

3.7.3. Evaluación federada

Algo fundamental a destacar es que la librería *TensorFlow Federated*, en la versión empleada (0.19.0), solo permite evaluar a través de métricas; no permite generar predicciones al detalle de muestras nuevas.

Remarcamos este aspecto importante y que influye en los resultados cuantitativos mostrados en la sección 4.2: debido a las características (o limitaciones) de la versión utilizada de la librería *TFF*, no es posible obtener una probabilidad a nivel píxel con la que componer imágenes respuesta a fin de localizar la masa tumoral y, con ello, obtener valores para las métricas de evaluación.

Esta falta de capacidad de predicción de la librería *TFF* nos fuerza, en cierto sentido, a llevar a cabo la evaluación de los modelos federados resultantes mediante dos vías paralelas.

1. Evaluación federada (con *TFF*): llevamos a cabo una evaluación de las métricas directamente con la librería federada *TFF*, usando directamente el modelo federado, que es lo máximo que puede hacerse. Esta agregación se calcula como un promedio de las evaluaciones en cada cliente, sin aplicar proporcionalidad ni peso alguno por cliente [1], como se muestra en la fórmula 3.1.

2. Evaluación clásica (sin *TFF*): recuperamos los pesos del modelo federado al modo clásico (directamente con la librería *TensorFlow*), unificamos los *test subsets* de cada cliente federado en un único conjunto de test (vía conversión entre librerías *TFF* a *TensorFlow*) y realizamos la predicción del subconjunto *test* y la evaluación de métricas.

[1] Evaluaciones empíricas propias contradicen el procedimiento descrito por la librería TFF. No ha podido confirmarse el motivo de la divergencia con Google, la desarrolladora de la librería.

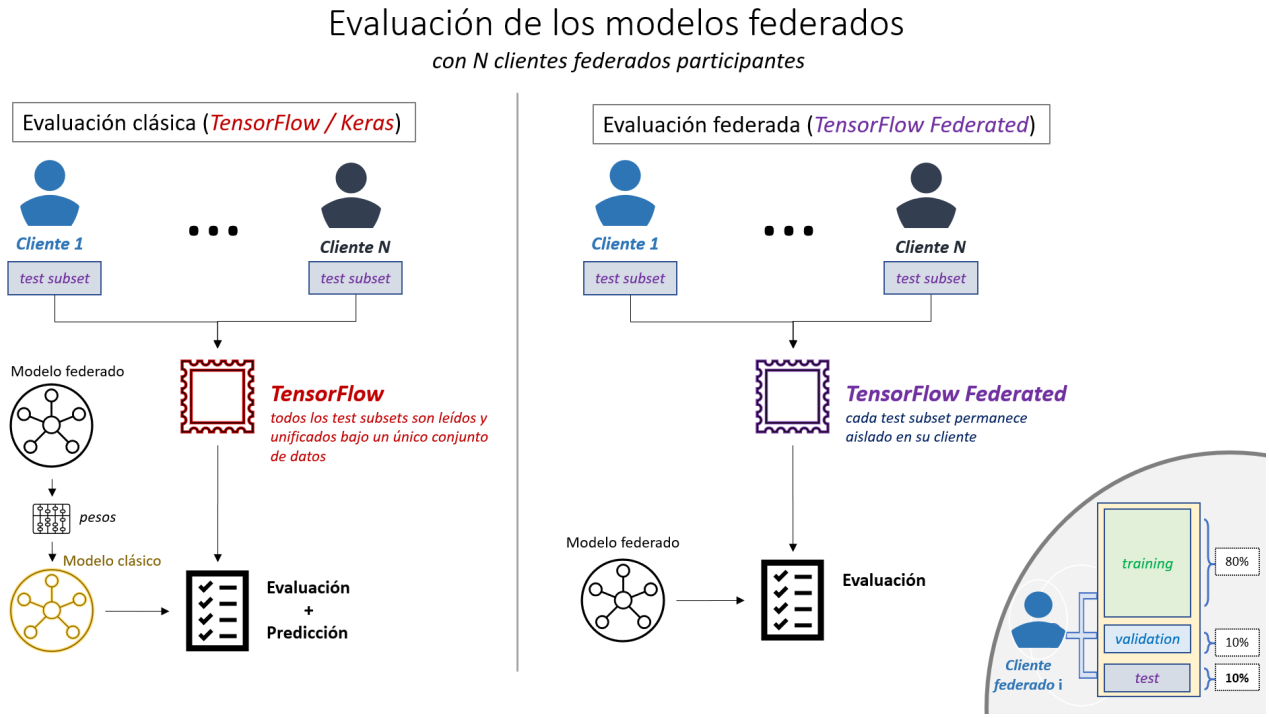

Figura 3.5: Diagrama representativo de los procesos de evaluación clásica (con *TensorFlow/Keras*) y federada (con *TFF*).

$$\textbf{Métrica}_{\textbf{test}} = \frac{1}{k}\sum_{i=1}^{k} metrica(C_k)$$

$$\text{donde } \begin{matrix} k \text{ es el número de clientes participantes en la evaluación federada y} \\ C_k \text{ es el } k\text{-cliente sobre el que se ejecuta una evaluación individual} \end{matrix} \tag{3.1}$$

La clave para que ambos procesos evaluadores sean análogos es que sean ejecutados sobre las mismas muestras, esto es, una muestra idéntica, como se reitera en la figura 3.5. Es por este motivo que los resultados recogidos en la subsección 4.2 incluirán dos puntuaciones: una obtenida con las librerías *TensorFlow/keras* y otra obtenida con la librería *TFF*.

Capítulo 4

Experimentos y resultados

En este capítulo, detallaremos las pruebas realizadas y los resultados obtenidos en las mismas. También comentaremos las dificultades enfrentadas en cuanto a la aplicación del aprendizaje federado, a fin de enriquecer el análisis de los propios resultados.

4.1. Pruebas

Se han acometido cuatro conjuntos de pruebas distintas recogidas en la tabla 4.1.

Todos los modelos, veintiocho (28) en total, siendo uno (1) clásico y veintisete (27) federados, comparten la misma arquitectura y parametrización de red neuronal *U-Net*, definida en la sección 3.4.

En el bloque CLÁSICO, se ha entrenado un único modelo mediante aprendizaje clásico, esto es, con un único conjunto de datos unificado y centralizado, fusión de los tres *datasets* A, B y C disponibles. Sus resultados servirán como punto de referencia para los siguientes modelos federados.

Cada bloque de pruebas federadas (FED-3, FED-5 y FED-7) incluye nueve (9) procesos federados

	BLOQUE DE PRUEBAS			
Característica	**CLÁSICO**	**FED-3**	**FED-5**	**FED-7**
Descripción general	Entrenamiento clásico	Entrenamiento federado		
Número de modelos entrenados	1	9	9	9
Número de clientes	1	3	5	7
Conformación de los *datasets*	A+B+C	A | B | C	A21 | A22 | B | C21 | C22	A31 | A32 | A33 | B | C31 | C32 | C33

Tabla 4.1: Pruebas ejecutadas para el proyecto

	Técnicas de agregación federadas por bloque de pruebas (etiquetas FED-x)		
N# test federado	**Técnica de agregación**	**Característica especial**	**Detalles**
0	*FedAvg*	-	Implementación por defecto, única disponible con la versión *vanilla* de *TFF*
1	*FedAvg* explícito (*MeanFactory*)	-	Llamada explícita a la factoría que implementa *FedAvg*
2	*RFA*	*Robust Federated Averaging*	Implementación con *TFF* del algoritmo *RFA* de Pillutla et al. (2019)
3	*FedAvg* + *Zeroing*	Control de casos extremos	Técnica para reemplazar valores inusualmente grandes por valor cero,Google (2022c)
4	*FedAvg* + *Clipping* adaptativo	Control de casos extremos	Técnica para controlar *outliers*, reescalado de los pesos bajo una norma dinámica, Google (2022c)
5	*FedAvg* + *Differential Privacy*	Privacidad diferencial	Técnica de introducción de ruido para contener ataques inferenciales, Google (2022c)
6	*FedAvg* + *Lossy Compression*	Compresión de datos	Técnica para comprimir el volumen de datos intercambiados entre servidor y clientes federados, Google (2022c)
7	*FedAvg* + *Secure Aggregation*	Protocolo de seguridad	Técnica de encriptación en la comunicación servidor-clientes federados, Google (2022c)
8	*FedAvg* + *Secure Aggreation (II)*		

Tabla 4.2: Pruebas federadas ejecutadas en cada bloque FED-x

distintos. Difieren en el método de agregación empleado, basándose en las pautas de *tuning* para entrenamiento federado con *TFF,* recomendadas por Google (2022c) (ver tabla 4.2).

En cada bloque de pruebas, intervienen distinto número de clientes (o *datasets*). La correspondencia entre cliente federado y dataset se recoge en la tabla 4.3.

4.2. Resultados

4.2.1. Métricas de evaluación

Los valores obtenidos en las métricas de evaluación durante los procesos de aprendizaje (clásico y federados) se recogen en las tablas 4.4, 4.5, 4.6 y 4.7. Se observarán dos valores para cada prueba unitaria, esto es, para cada modelo federado entrenado, como se explicó en la subsección 3.7.3. El primer valor de celda (a la izquierda) refleja la puntuación obtenida mediante la evaluación clásica (*Tensorflow* y *Keras*); el segundo valor de celda (a la derecha, entre paréntesis, en color gris y cursiva) corresponde a la evaluación federada, esto es, usando la librería *TFF*.

Bloque de pruebas	**Cliente**	***Datasets***
FED-3	ds_A	A completo
	ds_B	B completo
	ds_C	C completo
FED-5	ds_A21	Mitades (50 %) aleatorias del *dataset* A
	ds_A22	
	ds_B	B completo
	ds_C21	Mitades (50 %) aleatorias del *dataset* C
	ds_C22	
FED-7	ds_A31	Terceras partes (˜33 %) aleatorias del *dataset* A
	ds_A32	
	ds_A33	
	ds_B	B completo
	ds_C31	Terceras partes (˜33 %) aleatorias del *dataset* C
	ds_C32	
	ds_C33	

Tabla 4.3: Relación de clientes federados con los *datasets* disponibles en cada bloque de pruebas federadas

	Coeficiente Dice (%)			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	78,52 *(76,60)*	79,42 *(80,40)*	80,67 *(80,28)*	**79,54**
1	78,32 ***(77,94)***	80,83 *(80,61)*	79,42 *(77,97)*	
2	78,03 *(77,86)*	**81,83** ***(82,12)***	79,92 *(78,76)*	
3	**78,99** *(77,86)*	80,02 *(81,12)*	79,38 *(78,69)*	
4	77,75 *(76,09)*	80,85 *(81,41)*	77,69 *(75,51)*	
5	78,98 *(74,81)*	79,60 *(81,05)*	**81,28** ***(84,29)***	
6	76,02 *(76,66)*	80,66 *(80,93)*	78,78 *(77,29)*	
7	78,68 *(77,48)*	78,70 *(80,39)*	76,77 *(76,95)*	
8	76,64 *(77,73)*	81,04 *(81,74)*	80,11 *(78,11)*	

Tabla 4.4: Puntuación (*coeficiente Dice*) de las pruebas de evaluación de test. En negrita, la mejor puntuación en cada bloque de prueba federada con n-clientes. Entre paréntesis, los resultados obtenidos con la función de evaluación federada de la librería *TFF*.

	Exactitud (%)			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	96,12 *(96,09)*	96,38 *(96,84)*	96,52 *(97,03)*	**96,54**
1	96,02 *(96,12)*	96,37 *(96,78)*	96,55 *(96,81)*	
2	96,02 ***(96,31)***	**96,66** ***(97,12)***	96,57 *(97,15)*	
3	**96,11** *(96,16)*	96,38 *(96,91)*	96,40 *(96,83)*	
4	95,91 *(95,99)*	96,47 *(96,90)*	96,35 *(96,57)*	
5	96,07 *(95,91)*	96,37 *(96,89)*	**96,85** ***(97,52)***	
6	95,76 *(95,98)*	96,46 *(96,82)*	96,52 *(96,73)*	
7	96,08 *(96,24)*	96,35 *(96,87)*	96,16 *(96,63)*	
8	95,88 *(96,09)*	96,49 *(96,93)*	96,53 *(96,88)*	

Tabla 4.5: Puntuación (exactitud, *accuracy*) de las pruebas de evaluación de test. En negrita, la mejor puntuación en cada bloque de prueba federada con n-clientes. Entre paréntesis, los resultados obtenidos con la función de evaluación federada de la librería *TFF*.

	Coeficiente IoU / Jaccard (%)			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	78,27 *(75,21)*	77,94 *(76,15)*	79,35 *(77,81)*	**80,27**
1	79,01 *(77,34)*	79,02 *(77,63)*	78,44 *(75,25)*	
2	79,01 ***(77,96)***	**80,97** ***(79,82)***	79,14 *(76,49)*	
3	78,17 *(76,01)*	79,42 *(78,94)*	78,45 *(76,17)*	
4	77,50 *(75,63)*	80,11 *(79,25)*	76,23 *(72,67)*	
5	77,06 *(73,67)*	79,09 *(78,90)*	**80,36** ***(80,46)***	
6	76,03 *(74,36)*	80,35 *(79,22)*	77,36 *(73,57)*	
7	**79,02** *(77,48)*	78,17 *(76,54)*	76,18 *(73,09)*	
8	77,11 *(75,64)*	80,10 *(79,27)*	78,60 *(76,10)*	

Tabla 4.6: Puntuación (*coeficiente IoU* o *Jaccard*) de las pruebas de evaluación de test. En negrita, la mejor puntuación en cada bloque de prueba federada con n-clientes. Entre paréntesis, los resultados obtenidos con la función de evaluación federada de la librería *TFF*.

	Sensibilidad (*recall*) (%)			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	71,94 *(70,95)*	70,92 *(75,03)*	**75,61** ***(77,42)***	**78,26**
1	72,89 *(73,93)*	73,83 *(76,71)*	72,14 *(71,98)*	
2	72,89 *(72,57)*	**75,50** ***(77,97)***	72,98 *(72,91)*	
3	71,56 *(73,64)*	74,15 *(78,50)*	73,70 *(75,11)*	
4	70,35 *(70,98)*	74,39 *(77,92)*	68,26 *(67,90)*	
5	70,65 *(68,27)*	70,98 *(74,52)*	74,23 *(78,09)*	
6	68,06 *(70,53)*	74,67 ***(77,97)***	70,26 *(69,19)*	
7	**73,24** ***(73,94)***	69,13 *(73,47)*	69,44 *(71,40)*	
8	69,71 *(72,46)*	74,83 *(78,07)*	72,65 *(72,79)*	

Tabla 4.7: Puntuación (sensibilidad) de las pruebas de evaluación de test. En negrita, la mejor puntuación en cada bloque de prueba federada con n-clientes. Entre paréntesis, los resultados obtenidos con la función de evaluación federada de la librería *TFF*.

4.2.2. Predicciones

A continuación, se exponen algunos ejemplos de la capacidad predictiva de los modelos entrenados. Las virtudes y los defectos mostrados son generalmente comunes a todos los modelos, tanto clásico como federados.

En la figura 4.1, pueden observarse ejemplos del desempeño del modelo clásico; la figura 4.2 recoge las predicciones de varios modelos federados del bloque FED-3. Por otro lado, las figuras 4.3 y 4.4 muestran casos en los que la masa tumoral no es siempre detectada.

Respecto de esa falta parcial de capacidad predictiva, y a fin de enriquecer el análisis, se han elaborado dos índices propios con más vocación práctica y que se centran en los fallos de predicción:

1. Falsas masas tumorales (MTF): se da cuando la imagen predictiva identifica una masa tumoral claramente separada de la masa real (si la hubiera, en la máscara GT) o cercana a esta y de un volumen similar a la real. Se calcula como el procentaje de imágenes que cumplen esta condición entre todas las imágenes testadas. La figura 4.5 muestra ejemplos de estos falsos positivos.

2. Masas tumorales no detectadas u obviadas (MTO): se da cuando la imagen predictiva no es capaz de identificar una masa tumoral claramente, bien porque no la situa con ningún conjunto de píxeles o bien porque la predicción solo identifica una parte muy minoritaria de la masa tumoral. Se calcula como el procentaje de imágenes que cumplen esta condición entre todas las imágenes testadas. La figura 4.6 muestra ejemplos de estos falsos negativos.

Los resultados de ambas métricas, MTF y MTO, se recogen en las tablas 4.8 y 4.9, respectivamente.

Si agrupamos los resultados de estos índices por 1) bloques de entrenamiento y 2) test o técnica de federación, obtenemos los promedios recogidos en las tablas 4.10 y 4.11.

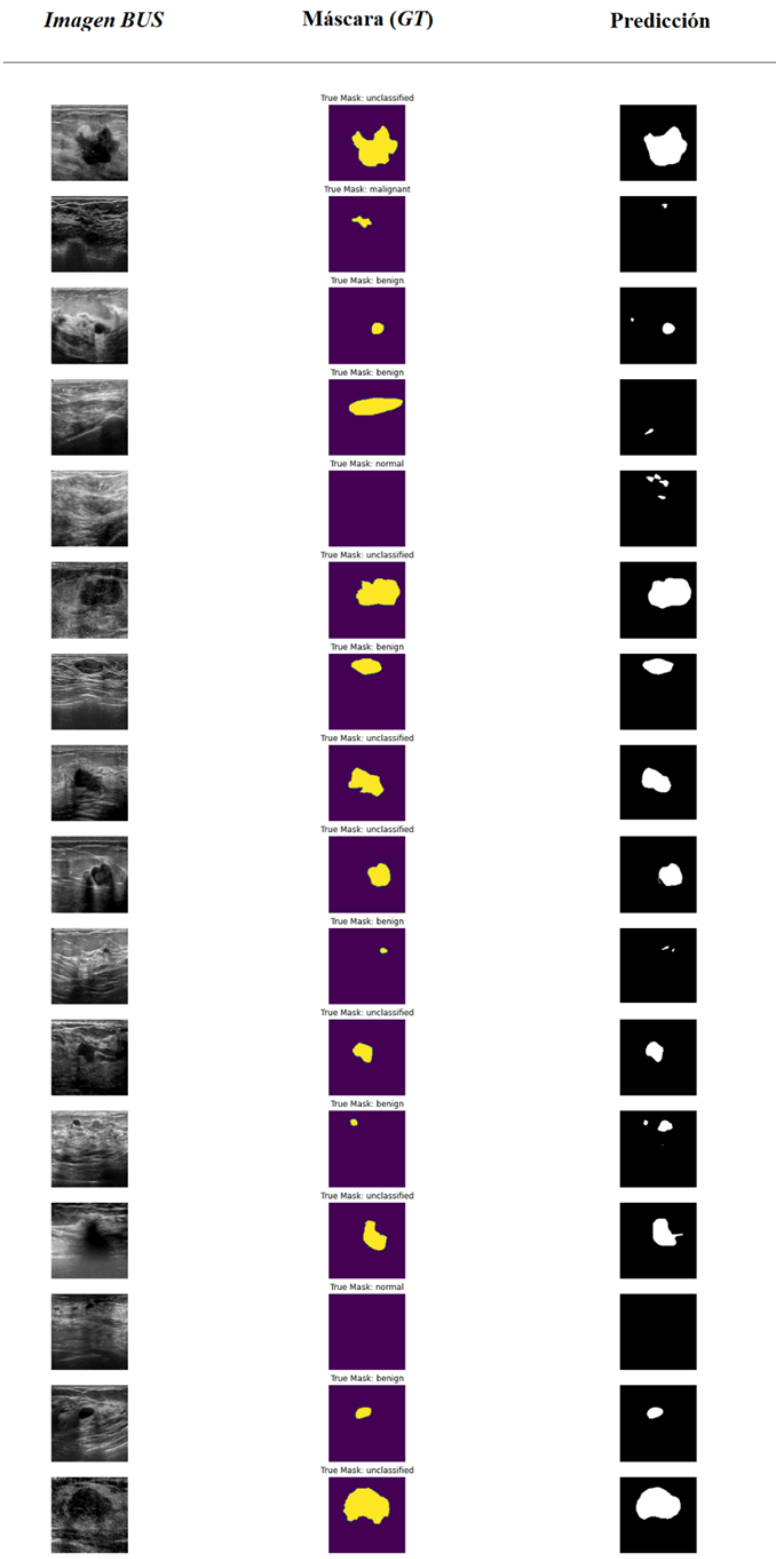

Figura 4.1: Predicciones realizadas por el modelo entrenado mediante aprendizaje clásico: imágenes *BUS* (columna izquierda) y sus respectivas máscaras (columna centro), más las predicciones (columna derecha).

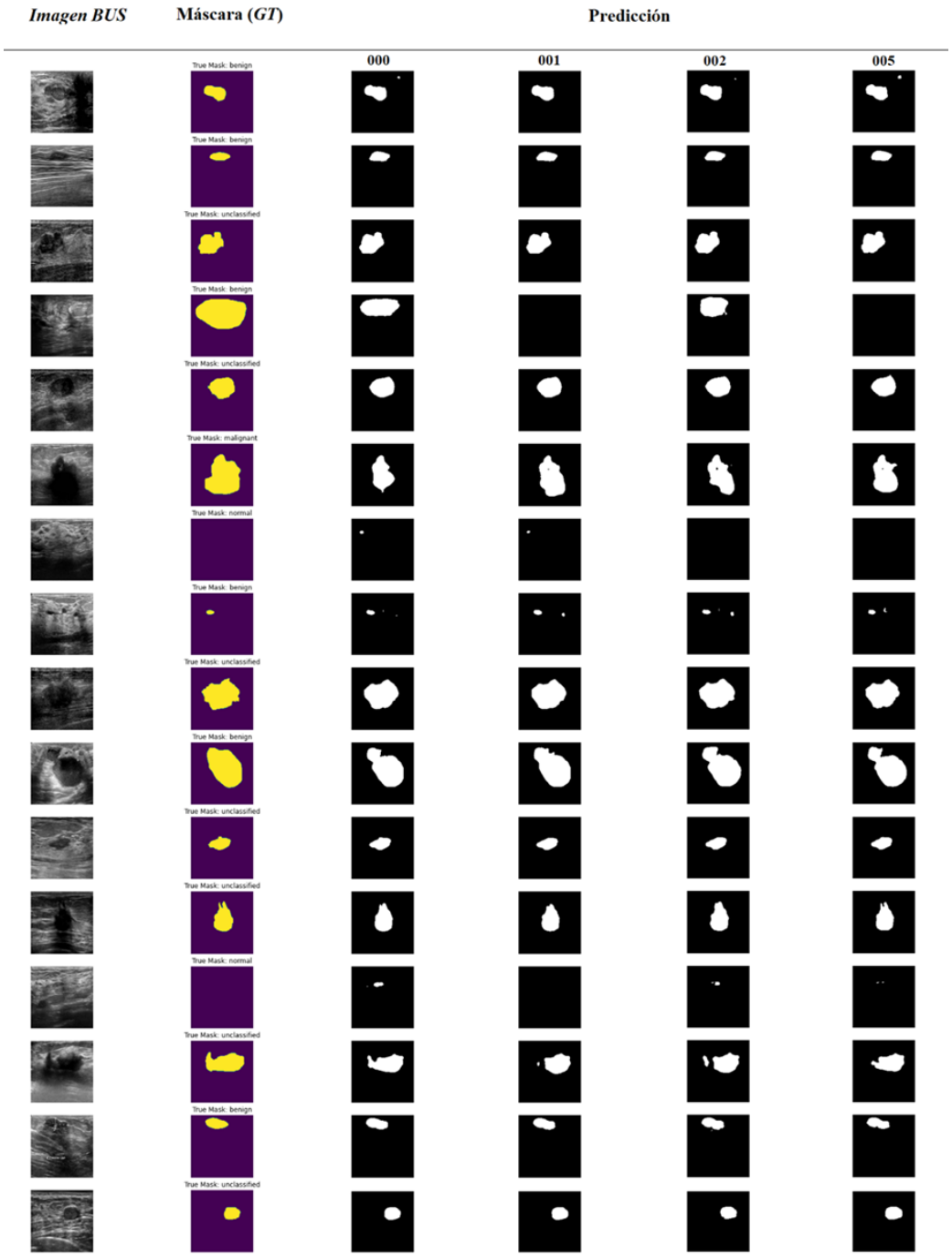

Figura 4.2: Predicciones realizadas por los modelos federados. De izquierda a derecha, imágenes *BUS*, sus respectivas máscaras, más las predicciones realizadas por los modelos #0, #1, #2 y #5.

Figura 4.3: Fallos de detección de masa tumoral por el modelo entrenado mediante aprendizaje clásico. Imagen *BUS* (izquierda) y su respectiva máscara (centro), más la predicción errónea (derecha).

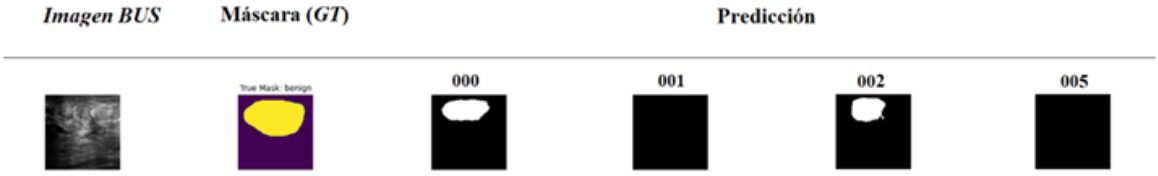

Figura 4.4: Fallos de detección de masa tumoral por los modelos federados. De izquierda a derecha, imagen BUS, su respectiva máscara y las predicciones realizadas por los modelos federados #0, #1, #2 y #5.

	Índice de falsas masas tumorales, MTF (%)			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	11,84	10,07	20,14	15,89
1	11,84	12,08	14,39	
2	9,21	11,41	14,39	
3	9,21	14,09	15,83	
4	8,55	14,77	11,51	
5	9,21	12,75	13,67	
6	11,18	16,11	10,07	
7	10,53	13,42	15,83	
8	12,50	15,44	12,23	

Tabla 4.8: Puntuación del índice de elaboración propia MTF sobre los modelos entrenados.

	Índice de masas tumorales no detectadas, MTO (%)			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	6,58	10,74	3,60	5,30
1	7,89	7,38	5,76	
2	9,21	8,72	6,47	
3	9,21	9,40	5,76	
4	9,87	8,05	6,47	
5	7,89	10,07	5,76	
6	11,18	8,72	5,76	
7	9,21	9,40	8,63	
8	12,5	7,38	8,63	

Tabla 4.9: Puntuación del índice de elaboración propia MTO sobre los modelos entrenados.

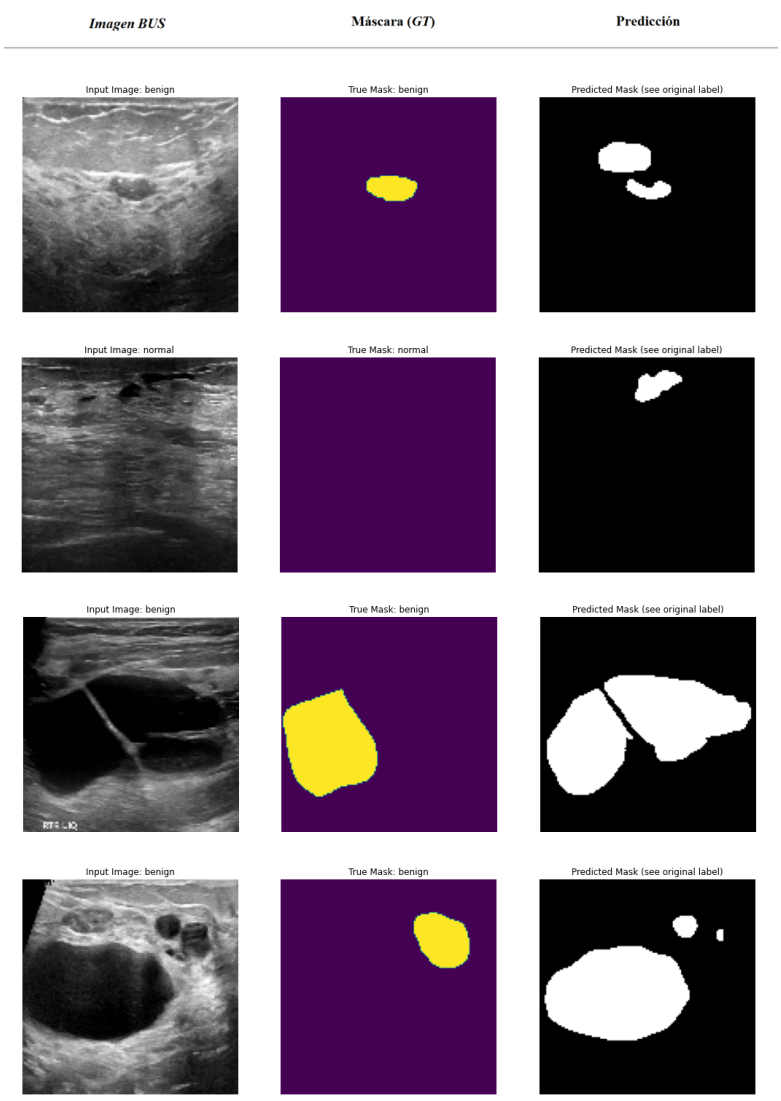

Figura 4.5: Ejemplos de falsas masas tumorales (MTF) en las predicciones de los modelos entrenados. De izquierda a derecha, imágenes *BUS*, sus respectivas máscaras y las predicciones.

	Promedios de los índices de generación propia (%)	
	MTF	**MTO**
Clásico	15,89	5,30
Federado (3C)	10,45	9,28
Federado (5C)	13,35	8,87
Federado (7C)	14,23	6,31

Tabla 4.10: Puntuación de los índices de elaboración propia MTF y MTO: promedios por bloque de entrenamiento.

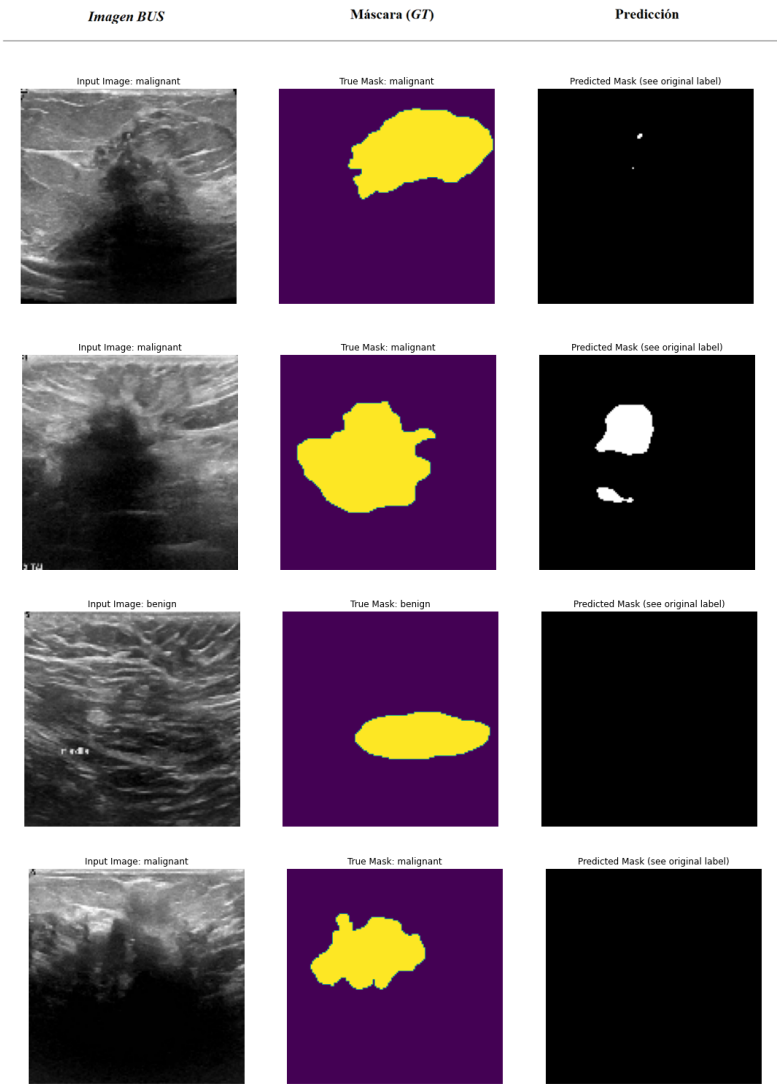

Figura 4.6: Ejemplos de masas tumorales no detectadas u obviadas (MTO) en las predicciones de los modelos entrenados. De izquierda a derecha, imágenes *BUS*, sus respectivas máscaras y las predicciones.

	Promedios de los índices de generación propia (%)	
N# test federado	**MTF**	**MTO**
0	14,02	6,97
1	12,77	7,01
2	11,67	8,14
3	13,04	8,12
4	11,61	8,13
5	11,88	7,91
6	12,45	8,55
7	13,26	9,08
8	13,39	9,51

Tabla 4.11: Puntuación de los índices de elaboración propia MTF y MTO: promedios por test de entrenamiento federado.

	Duración del entrenamiento			
N# test federado	**Federado (3C)**	**Federado (5C)**	**Federado (7C)**	**Clásico**
0	3h 51min 52s	3h 47min 28s	3h 49min 52s	56min 41s
1	3h 42min 58s	3h 39min 52s	3h 41min 10s	
2	4h 53min 56s	5h 30min 20s	6h 42s	
3	3h 47min 46s	3h 47min 56s	3h 46min 12s	
4	3h 46min 24s	3h 47min 56s	3h 46min 46s	
5	3h 44min 6s	3h 43min 34s	3h 41min 44s	
6	4h 18s	4h 17min 22s	4h 13min 54s	
7	4h 25min 38s	4h 33min 28s	5h 5min 48s	
8	4h 32min 6s	4h 41min 58s	5h 52s	

Tabla 4.12: Duración de la fase de entrenamiento para cada modelo objeto de estudio

4.2.3. Fase de entrenamiento

4.2.3.1. Duración

La tabla 4.12 recoge el tiempo de duración de cada uno de los entrenamientos de los modelos producidos.

4.2.3.2. Evolución de la función de pérdida y la exactitud

Esta sección recoge la evolución de la fase de entrenamiento del modelo clásico y de tres de las configuraciones federadas que han sido testadas. Las gráficas completas de todos los modelos generados pueden consultarse en el anexo A.1.

La evolución de la función de pérdida (*1 – Dice*) durante el entrenamiento puede observarse en las gráficas 4.7, 4.8, 4.9 y 4.10.

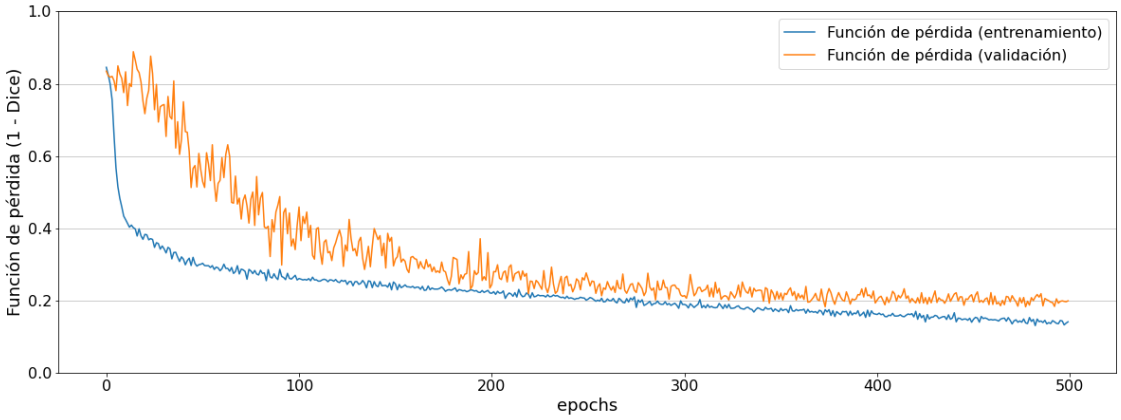

Figura 4.7: Función de pérdida (*1 – Dice*) durante el entrenamiento clásico

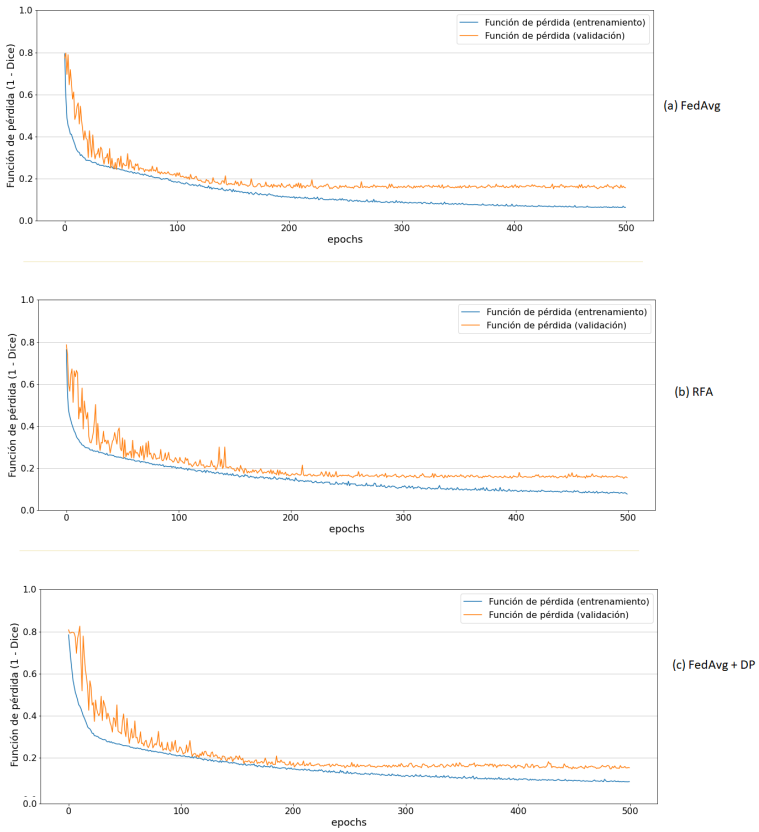

Figura 4.8: Función de pérdida (*1 – Dice*) durante el entrenamiento federado (3 clientes): tests #0 (a), #2 (b) y #5 (c).

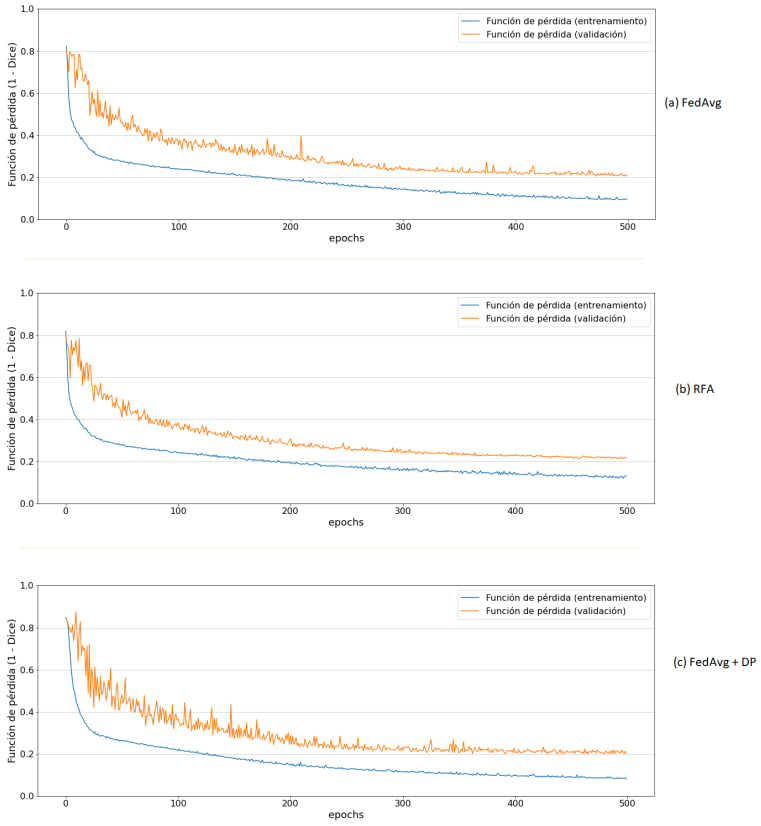

Figura 4.9: Función de pérdida (*1 – Dice*) durante el entrenamiento federado (5 clientes): tests #0 (a), #2 (b) y #5 (c).

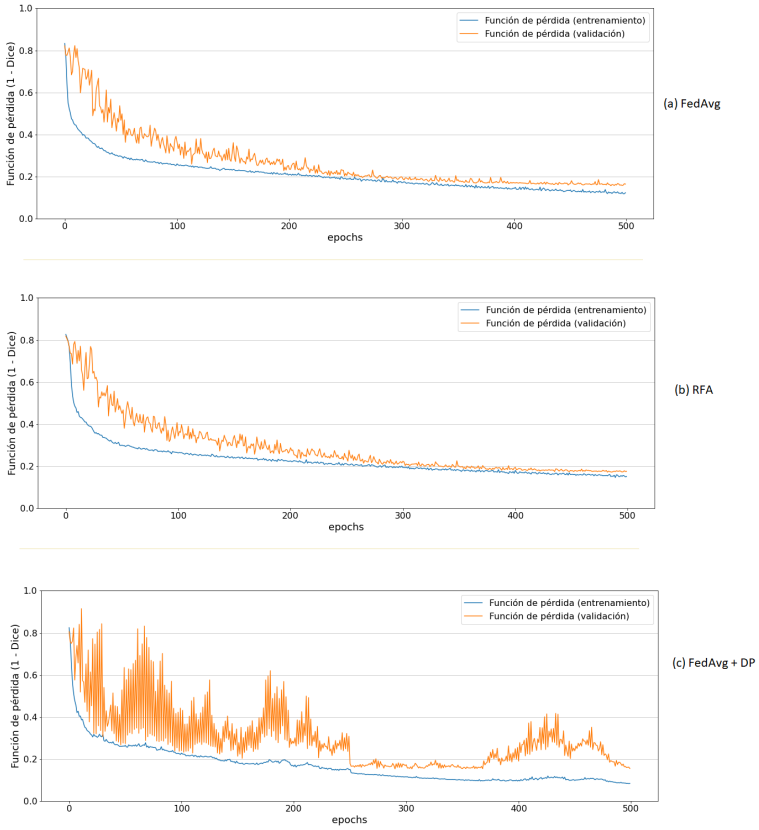

Figura 4.10: Función de pérdida (*1 – Dice*) durante el entrenamiento federado (7 clientes): tests #0 (a), #2 (b) y #5 (c).

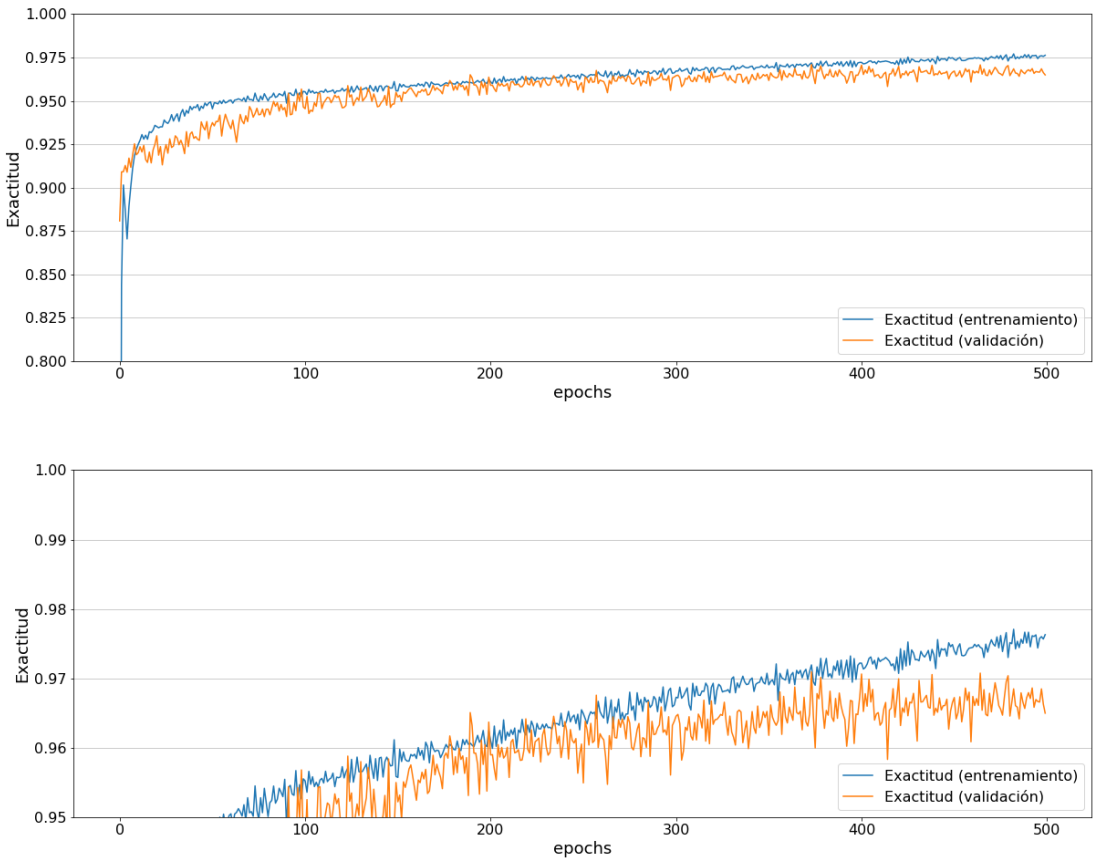

Figura 4.11: Exactitud (*accuracy*) durante el entrenamiento clásico. De arriba abajo: gráfica general y gráfica aumentada en el rango >95 %.

Y la evolución de la exactitud (*accuracy*) durante el entrenamiento se recoge en las gráficas 4.11, 4.12, 4.13, y 4.14.

4.3. Discusión

A continuación, discutiremos sobre la experiencia y los resultados obtenidos en el proyecto.

Dado el alto número de modelos federados entrenados (27), en lo sucesivo, las figuras de ejemplo referentes a tests federados unitarios muestras los tests #0 (*FedAvg*), #2 (*RFA*) y #5 (privacidad diferencial). El primero, por ser la opción por defecto con la librería *TFF*; el segundo, por representar una alternativa a *FedAvg* y, el tercero, por su posible aplicación a contextos médicos.

No ahondaremos en las opciones de compresión (test #6) ni de encriptación (tests #7 y #8), más allá de destacar un desempeño ligeramente inferior, en líneas generales, en todas las configuraciones de entrenamiento federado. Han sido probadas con éxito siguiendo las pautas básicas de *tuning* Google (2022c), lo cual era el objetivo principal, y para ellas se encontrará una mención especial en la sección 5.2.

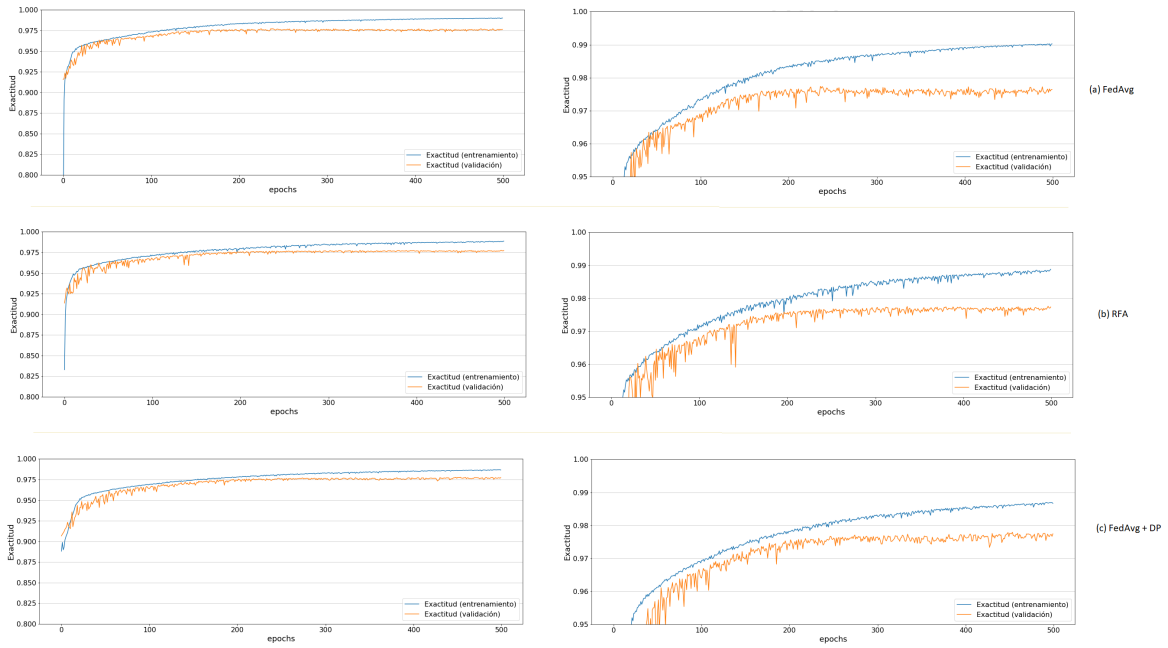

Figura 4.12: Exactitud (*accuracy*) durante el entrenamiento federado (3 clientes), de izquierda a derecha y de arriba abajo: tests #0 (a), #2 (b) y #5 (c), gráficas generales y gráfica aumentada en el rango >95 %.

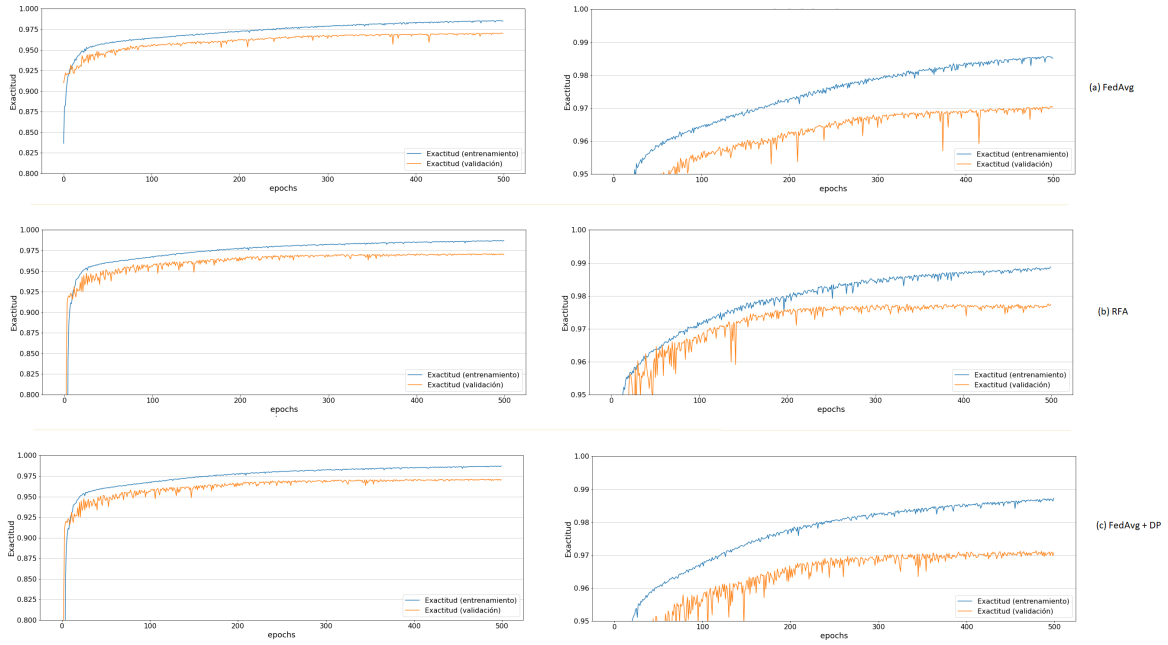

Figura 4.13: Exactitud (*accuracy*) durante el entrenamiento federado (5 clientes), de izquierda a derecha y de arriba abajo: tests #0 (a), #2 (b) y #5 (c), gráficas generales y gráfica aumentada en el rango >95 %.

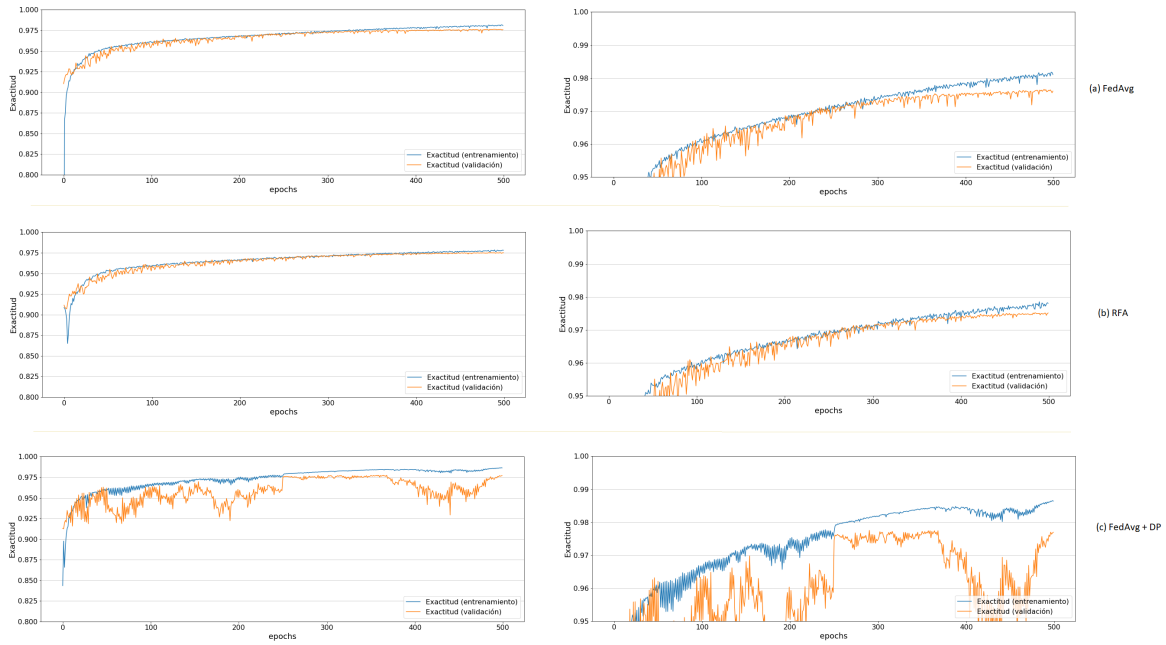

Figura 4.14: Exactitud (*accuracy*) durante el entrenamiento federado (7 clientes), de izquierda a derecha y de arriba abajo: tests #0 (a), #2 (b) y #5 (c), gráficas generales y gráfica aumentada en el rango >95 %.

Tampoco profundizaremos en el análisis de los resultados de los tests redundante (test #1) y los que tratan de controlar valores anómalos, esto es, los test #3 (*zeroing*) y #4 (*clipping*), puesto que son técnicas ideadas para aplicar usualmente en contextos con un mayor número de clientes y una heterogeneidad mayor de los datos entre clientes (*cross-device*, por ejemplo).

No obstante, las gráficas completas de todos los modelos federados pueden consultarse en el anexo A.1.

4.3.1. Análisis de los resultados

4.3.1.1. Métricas de evaluación

Lo primero que se debe destacar, respecto de los modelos federados, es la diferencia de puntuación entre las dos evaluaciones ejecutadas (vía clásica y vía *TFF*). Esto resulta de la forma en que la librería *TFF* calcula sus métricas (en cada cliente) y las agrega a posteriori, como se ha explicado en la subsección 3.7.3. La documentación de la librería *TFF* no explicita las razones de aplicar tal metodología pero puede comprobarse en su código, que es libre.

Por tanto, téngase únicamente en cuenta la puntuación de la librería *TFF* como medida aproximada. En lo sucesivo, nos centraremos en la evaluación sin la librería *TFF*, esto es, a la manera clásica (librerías *TensorFlow* y *Keras*),

El siguiente hecho que resaltamos es que, si bien se aprecia cierta similitud entre todos los modelos entrenados, hay diferencias sensibles en cuanto a las puntuaciones de métricas obtenidas.

En una lectura trasversal de las métricas federadas, observamos que la exactitud presenta una

Figura 4.15: Diagrama de cajas (*boxplot*) de las puntuaciones de la evaluación de la exactitud (*accuracy*), por bloques de entrenamiento. A la izquierda, figura (a), los resultados obtenidos mediante la evaluación clásica (sin la librería *TensorFlow Federated*); a la derecha, figura (b), los resultados obtenidos mediante la evaluación federada (con la librería *TensorFlow Federated*);

diferencia de casi un punto porcentual (0,97 %) entre los modelos con peor y mejor puntuación. Ese contraste es mayor si atendemos al resto de métricas: con la función de pérdida, la diferencia es de 5,81 %; el coeficiente *Jaccard / IoU*, 4,91 %; para la sensibilidad (*recall*), se observa un *gap* de 7,55 %.

Haciendo una lectura por bloques de entrenamiento federado (FED-3, FED-5, FED-7), no hay una tendencia del todo definida. Observando la exactitud, se obtiene mejor puntuación cuantos más clientes participan en la federación, aunque con resultados bastante parejos entre los bloques FED-5 y FED-7. Sin embargo, el análisis del resto de métricas pareciera indicar que el entrenamiento federado más óptimo se encuentra bajo la configuración con cinco clientes (FED-5), seguido de cerca por el bloque FED-7 y, ligeramente más alejado, el bloque con los tres clientes/*datasets* originales (FED-3). Sirvan las figuras 4.15, 4.16, 4.17 y 4.18 como soporte para una mejor comprensión visual. Este comportamiento podría justificarse por el equilibrio más justo entre el número de clientes y el volumen del dataset en cada cliente bajo la configuración FED-5, es decir, que un número concreto de clientes participantes esté favoreciendo un mayor balanceo, de modo que suficientes muestras por cliente aporten generalidad al modelo global. También es muy probable que los resultados se deriven de la aleatoriedad de la selección de los subconjuntos de test en cada configuración federada y en cada cliente.

En términos absolutos, no parece razonable concluir que haya una técnica de agregación federada ganadora, o que ofrezca los mejores resultados en todos los bloques. Aunque sí es cierto que los tests que mejores puntuaciones obtienen en conjunto son el #5 (privacidad diferencial), #2 (*RFA*) y #0 (*FedAvg*), por este orden. Mención especial merecería el test #5, pues todas sus puntuaciones mantienen un alza constante cuanto mayor es el número de clientes federados, hecho único y diferencial con el resto de tests de agregación federada.

Por último y muy importante, debemos comparar las métricas federadas con las puntuaciones de referencia del modelo CLÁSICO. El modelo clásico arroja una evaluación del coeficiente *Dice* con

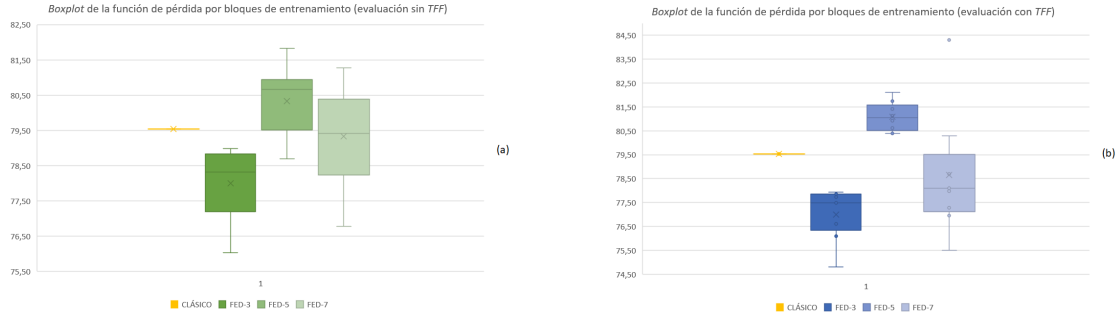

Figura 4.16: Diagrama de cajas (*boxplot*) de las puntuaciones de la evaluación de la función de pérdida (*1 - Dice*), por bloques de entrenamiento. A la izquierda, figura (a), los resultados obtenidos mediante la evaluación clásica (sin la librería *TensorFlow Federated*); a la derecha, figura (b), los resultados obtenidos mediante la evaluación federada (con la librería *TensorFlow Federated*);

Figura 4.17: Diagrama de cajas (*boxplot*) de las puntuaciones de la evaluación del coeficiente *Jaccard / IoU*, por bloques de entrenamiento. A la izquierda, figura (a), los resultados obtenidos mediante la evaluación clásica (sin la librería *TensorFlow Federated*); a la derecha, figura (b), los resultados obtenidos mediante la evaluación federada (con la librería *TensorFlow Federated*);

Figura 4.18: Diagrama de cajas (*boxplot*) de las puntuaciones de la evaluación de la sensibilidad (*recall*), por bloques de entrenamiento. A la izquierda, figura (a), los resultados obtenidos mediante la evaluación clásica (sin la librería *TensorFlow Federated*); a la derecha, figura (b), los resultados obtenidos mediante la evaluación federada (con la librería *TensorFlow Federated*);

valor 79,54 % y una exactitud de 96,54 %. En contraste con los federados, no podemos tampoco concluir hacia una tendencia clara:

- En cuanto a la exactitud, solo cuatro (4) modelos federados puntúan ligeramente mejor que el modelo clásico: la diferencia con el mejor modelo federado (FED-7, #5) es de tres décimas porcentuales (-0,31 %).
- En cuanto a la función de pérdida, hasta doce (12) federados obtienen mejor nota que el clásico: la diferencia aquí es sensiblemente mayor (FED-5, #2), un -2,29 %.
- El coeficiente *Jaccard/IoU*, por su parte, presenta (3) modelos federados con mayor puntuación que el clásico: la diferencia, del -0,70 % (FED-5, #2).
- Por último, para la sensibilidad (*recall*) no hay ningún modelo federado que mejore al clásico. La diferencia respecto del mejor federado es de +2,65 %.

En definitiva, debemos enfatizar de nuevo las limitaciones que han condicionado los experimentos realizados y que tienen su raíz en el volumen de datos de que disponemos. Esto ha impedido testar configuraciones federadas en las que participaran muchos más clientes y que, seguramente, nos hubieran permitido arrojar conclusiones más sólidas. Dicho lo cual, lo que sí podemos concluir es que el modelo clásico ha predominado en términos generales sobre la mayoría de modelos federados, aunque también hemos conseguido crear modelos federados con prestaciones similares, si no superiores, a aquel. Este hecho puede servir de guía para futuras federaciones; no tanto en el criterio de selección del número de clientes, lo cual no es una opción en sí ya que vendrá dado de los contextos reales en que se aplique el aprendizaje federado, sino en la utilización de las técnicas de agregación con mejor desempeño descritas anteriormente que puedan ofrecer resultados en la línea de un teórico aprendizaje clásico de referencia.

4.3.1.2. Predicciones

Todos los modelos, tanto el clásico como los federados seleccionados, presentan capacidades predictivas similares, lo cual está en línea con los resultados de las métricas de evaluación comentados anteriormente.

Desde un punto de vista más subjetivo, los índices de elaboración propia MTF y MTO, subsección 4.2.2, cuadros 4.8, 4.9, 4.10 y 4.11, indican que los modelos tienen mayor tendencia a identificar falsos positivos, esto es, predecir masas tumorales donde no las hay, que a errar en el diagnóstico de masas tumorales reales, o falsos negativos. El índice MTF global a todos los modelos federados es del 12,68 %; el índice MTO puntúa más de cuatro puntos porcentuales menos (8,16 %).

Si comparamos por bloques de entrenamiento, cuadro 4.10, puede apreciarse el intercambio (*trade-off)* entre ambos índices: cuantos más clientes participan de la federación, mayor es el número

de masas tumorales identificadas erróneamente (MTF) y menor es el índice de masas tumorales reales no detectadas (MTO).

En contraste con el aprendizaje clásico, que obtiene un índice MTF superior (15,89 %) y un MTO menor (5,30 %), puede afirmarse que, en general, el modelo clásico tiende a identificar más masas tumorales de las reales aunque, por otro lado, presenta una tasa de falsos negativos menor.

Por otro lado, atendiendo a los resultados agrupados por test o técnica de agregación unitaria, no parece apreciarse una tendencia clara en un sentido u otro.

4.3.1.3. Fase de entrenamiento

Respecto del tiempo empleado para entrenar los modelos, recogido en la tabla 4.12, cabe destacar:

1. El entrenamiento federado es más lento que el entrenamiento clásico: en el mejor de los casos, el entrenamiento federado ha necesitado cuatro veces más de tiempo de ejecución que el entrenamiento clásico,. Dependiendo de la técnica de agregación, puede que el entrenamiento federado resulte seis veces más lento que el clásico.

2. Entrenamientos federados interrumpidos: para los modelos federados de cinco (5) y siete (7) clientes, bloques FED-5 y FED-7, se observó un comportamiento errático en la ejecución del entrenamiento, deteniéndose e interrumpiendo el proceso de manera continua. Ello provocó que el tiempo real de consecución del entrenamiento fuera mayor, al requerir de interveción manual para reiniciarlo. Estos parones se achacan, en principio, a la gestión o la falta de memoria RAM por parte de la librería *TFF* una vez transcurridas decenas de rondas de entrenamiento federado.

Por otro lado, en relación al proceso de entrenamiento en sí, todos los modelos y sus métricas presentan una evolución con un comportamiento normal, sin signos de sobreajuste. Los modelos federados del bloque FED-7 presentan una evolución mucho más pareja de las puntuaciones de *training* y *validation*, salvo el caso #5 (DP) del bloque FED-7 (privacidad diferencial). Puede afirmarse que cuantos más clientes (o *datasets* federados) mayor es el número de rondas de entrenamiento necesarias para alcanzar las cotas más altas de puntuación, antes de entrar en una zona de estabilidad o no mejora de las métricas empleadas. En este sentido, para el bloque FED-7 (7 clientes o *datasets*), se podría explorar un entrenamiento con mayor número de rondas (˜1000), así como afinar el nivel de ruido introducido durante el aprendizaje (ver sección de mejoras futuras, 5.2).

4.3.2. Retos enfrentados

Debemos de destacar una serie de condicionantes que han marcado el desarrollo del proyecto. Respecto de la tarea a resolver, la mayor dificultad estriba en el etiquetado de las imágenes *BUS*: las máscaras *ground-truth*. Este etiquetado (manual) no suele ser extremadamente preciso, lo cual puede suponer un hándicap en el aprendizaje de la red neuronal Yu et al. (2021).

Siguiendo con los datos, disponemos de *datasets* que presentan distintas calidades de representación (distintas resoluciones de imagen *BUS*) y, además, se trata de un conjunto relativamente pequeño para los estándares del Aprendizaje Automático. Si bien es cierto que la red *U-Net* se desenvuelve correctamente con volúmenes de información pequeños, siempre es deseable disponer de cuanta mayor muestra posible, lo cual impacta en la calidad y generalidad del modelo entrenado.

Respecto de la librería TensorFlow Federated (*TFF*), tecnología empleada para el aprendizaje federado, se deben destacar varias dificultades enfrentadas. Primero, se trata de una tecnología de reciente creación que, aunque madura en las capacidades disponibles, presenta carencias propias de un desarrollo joven.

Una limitación es la ausencia de módulo o función que incluya una validación al vuelo, de manera similar al aprendizaje clásico, una suerte de *validation split*, que ayude a detectar un posible sobreajuste durante el entrenamiento. También se experimentaron fallos de ejecución que obligaron a renunciar a ciertas actividades o que requirieron adaptar el código de forma especial. Por ejemplo, en la etapa de preprocesado, fue necesario reordenar las transformaciones por fallos en la ejecución del código (el redimensionado debía ocurrir antes que otras operaciones sobre las imágenes de entrada).

Otro problema ha surgido durante la copia de seguridad y la restauración de modelos (o sus pesos), que no es del todo robusta; varias pruebas realizadas (*adaptive learning rate decay,* Google (2022b)) no pudieron ser sometidas a análisis ni ser incluidas en el presente proyecto, al encontrar errores de ejecución de código, sin solución de soporte disponible.

También surgieron fallos al insertar personalizaciones en ciertos parámetros del aprendizaje federado, como es el cálculo de la función de pérdida mediante una clase personalizada; algo que funcionaba correctamente en el modelo clásico, resultaba erróneo o arrojaba valores anómalos en el proceso federado.

Otro aspecto de la tecnología *TFF* para tener en cuenta es la cantidad de recursos computacionales que necesita para su correcta ejecución, en concreto, la memoria GPU. Observamos un uso constante en cada entrenamiento federado de casi la totalidad de los 32GB de que disponía el procesador GPU utilizado para las pruebas. Ciertos tests no pudieron ser incluidos por fallos de ejecución al exceder dicha capacidad de procesamiento. Otros que sí se incluyeron, como aquellos de los bloques FED-5 y FED-7 sufrieron de continuas interrumpciones por, en principio, falta de recursos GPU, como se indicaba en la subsección 4.3.1.3.

En este sentido, resultó crucial el disponer de la máquina TESLA de la propia Universidad para la consecución de las pruebas. Dado el contexto del proyecto (*datasets* relativamente pequeños y arquitectura y parametrización *U-Net* básicas), debe considerarse un elemento al que prestar mucha atención en futuros usos y pruebas.

Respecto de la personalización de la librería *TFF*, hay que destacar que, si bien la implementación de un aprendizaje federado básico es relativamente sencilla, expandir sus capacidades (por ejemplo, implementar algoritmos como los descritos en la subsección 2.3.5) puede constituir un reto que podría abarcar en sí mismo un proyecto completo.

Capítulo 5

Conclusiones

Cerraremos la exposición con las conclusiones derivadas del ejercicio de este proyecto TFM que, en lo referente al aprendizaje federado, deberían ser entendidas en el marco y las limitaciones del trabajo realizado. Para finalizar, expondremos todas las ideas no ejecutadas que podrían abordarse a fin de mejorar el presente esfuerzo y profundizar en el conocimiento del aprendizaje federado.

5.1. Eficacia del aprendizaje federado

Podemos afirmar que, bajo algunas configuraciones, el aprendizaje federado ha conseguido alcanzar las mismas cotas de eficacia, y en algunos casos superiores, marcadas por un modelo entrenado de forma clásica. No obstante, la diferencia de resultados entre los modelos federados invita a pensar que debe procederse con cautela.

Por otro lado, la disparidad de resultados respecto de la técnica de agregación aplicada advierte que la elección de ésta será un factor decisivo en los resultados finales. ¿Significa esto que debemos renunciar a aplicar técnicas de compresión o encriptación en pos de una mayor eficacia? Volveremos a ello en la sección 5.2. Desde luego, la experiencia adquirida en el presente proyecto aconseja, primero, empezar con el algoritmo *FedAvg* como punto de referencia y, después, implementar alguna técnica o *tuning*, si fuera posible.

Respecto del proceso de entrenamiento en sí, debe destacarse que el aprendizaje federado ha necesitado de entre cuatro y seis veces más tiempo que el entrenamiento clásico. Recordemos que estamos en un contexto de simulación, con clientes siempre disponibles, sin limitaciones de red y un entorno computacional idéntico para todos los modelos de aprendizaje. Este es un factor que considerar a la hora de preparar entornos de simulación más realistas o desplegar el aprendizaje federado a producción.

5.2. Trabajos futuros

En esta sección se presentan posibles acciones que podrían mejorar los resultados obtenidos o bien aportarían un conocimiento mayor del aprendizaje federado y de la tecnología empleada.

5.2.1. Tarea a resolver

Primeramente, habría que intentar extender la capacidad predictiva a la tarea de clasificación de la masa tumoral, es decir, no solo localizarla sino también determinar si es benigna o maligna. Para ello, habría que sustituir o complementar el modelo de red neuronal *U-Net*.

Por ejemplo, en Eroğlu et al. (2021), se entrenan tres redes neuronales distintas (*Alexnet, MobilenetV2 y Resnet50*) para la extracción de características que luego pasan a un clasificador (*SVM, KNN*) para la toma de decisión.

Una arquitectura de más reciente creación, *Mask R-CNN* He et al. (2017), es capaz de resolver ambas tareas (clasificación y segmentación) al mismo tiempo. En su caso, normalmente hay que aportar como máscara *ground truth* un etiquetado en forma de caja (*bounding box*) que habría que crear previamente, pues no es parte de las máscaras de los *datasets* tratados en el proyecto. No obstante, este esfuerzo sería relativamente bajo y podría ser una vía de mejora muy interesante.

Por otro lado, con el único fin de mejorar el desempeño en la segmentación, sería recomendable explorar variantes de *U-Net*, como en Byra et al. (2020), que utiliza una llamada *SK-UNet*, citada en la sección 2.2.3. Aplicar este ejemplo sería bastante directo pues el código para probarlo es público y disponible en Internet. Otro ejemplo sería Iglovikov and Shvets (2018), que aplica *U-Net* con una extracción de características de una red preentrenada (arquitectura *VGG16*).

5.2.2. *Datasets*

En el presente proyecto se han utilizado imágenes de ultrasonido para tratar de detectar tumores mamarios. Podríamos tratar de conseguir otros datos, públicos a ser posible. Como ejemplo, puede revisarse Wen (2021).

Si quisiéramos mantener la fuente técnica de los datos (ultrasonidos) pero no disponemos de más *datasets*, podría explorarse el uso de redes *GAN* para generar imágenes sintéticas, como en Chang et al. (2020) o en Chen et al. (2021).

Si pudiésemos ampliar el espectro de datos, sería interesante experimentar con mamografías de distinta fuente técnica Berg et al. (2004); Kelly et al. (2009).

5.2.3. Preprocesado

Podría profundizarse aún más en las pruebas que no resultaron satisfactorias, mencionadas en la subsección 3.5.3: filtros gaussiano, de mediana, detección de bordes y *thresholding*.

Sería muy recomendable tratar de reducir el tamaño de las imágenes sin perjuicio de eficacia: de 128x128 a 64x64 píxeles, por ejemplo. Esto redundaría positivametne en la eficiencia computacional y podría solventar los fallos de ejecución enfrentados por falta de memoria RAM.

5.2.4. Entrenamiento y evaluación

Una de las primeras tareas a acometer sería prolongar los entrenamientos, tanto clásico como federados, esto es, aumentar el número de rondas de federación. En el caso del entrenamiento clásico, tenemos la certeza de que habría mejora (*1000 epochs*, ver anexo B.1); en el caso federado, los recursos disponibles no han hecho posible este ejercicio.

También podrían explorarse más funciones de pérdida de las mencionadas en la subsección 3.7.1, como la *Tversky Loss*. En Abraham and Khan (2019) afirman obtener mejores resultados que con el *coeficiente Dice*.

Igualmente, se pueden explorar otras métricas para guiar la función de pérdida, ya comentadas en Jadon (2020), Ma (2021) o Punn and Agarwal (2021). De hecho, la segunda referenciada ofrece código implementado para Python y *TensorFlow*.

Otra alternativa similar, si se quiere usar *PyTorch*, se encuentra disponible en Jadon (2021).

Enlazando con el aprendizaje federado, se podría profundizar en las implicaciones de usar la evaluación con la librería *TFF*, como se explicaba en la sección 4.3.

5.2.5. Aprendizaje federado

En este área visualizamos múltiples opciones de investigación y mejora.

En primer lugar, sobre el estudio realizado en el presente proyecto, sería conveniente investigar por qué ha habido peor desempeño general en los modelos de compresión (test #6) y de encriptación (tests #7 y #8) pues se supone que todos implementan el mismo algoritmo de agregación con mejor desempeño (*FedAvg*, tests #0 y #1, por ejemplo). Pudiera tratarse de alguna depuración de código de la librería *TFF* pendiente de solventar.

Más allá de las tareas realizadas, pero preservando el mismo entorno de simulación ocupado para el proyecto, se deberían probar más algoritmos de aprendizaje federado distintos a *FedAvg*. La librería *TFF*, en su actual desarrollo (esto es, la versión *nightly*) permitiría aplicar *FedProx*. El inconveniente de la versión *nightly* es que no dispone de soporte técnico, por lo que sería conveniente esperar al siguiente *release* estable de *TFF* (0.20).

Existe la opción de implementar de manera propia algoritmos Google (2022h,d,e,f), como los mencionados en la sección 2.3.5, o también la de aplicar más técnicas federadas, recogidas bajo el paraguas investigador de Google Google (2022b).

Una de las características evaluadas por este proyecto ha sido la privacidad diferencial. Según las pruebas, los modelos finales han resultado más o menos acertados. Esto se debe, en parte, a la

cantidad de ruido que es posible introducir en el proceso de agregación de cada cliente federado. Se podrían realizar pruebas en este sentido, como se indica en Google (2022g).

Profundizando mucho más en cuanto a la privacidad de los datos y los posibles ataques que puede sufrir un entorno de aprendizaje federado, destacan Bagdasaryan et al. (2020) y Melis et al. (2019), que proponen sendos algoritmos capaces de detectar un intento de suplantación del modelo agregado desde un modelo adversario cliente (por ejemplo, un dispositivo infectado con *malware*) con capacidad de inferir características del aprendizaje general. En Truex et al. (2019) se realiza un estudio completo, aplicando varios algoritmos federados para asegurar la privacidad diferencial en modelos de red neuronal y máquinas de vector soporte (*SVM*).

Por otro lado, estaría la opción de testar el desempeño del aprendizaje federado en contextos más realistas. A continuación, se comentan algunos casos de éxito que podrían servir de ayuda.

En Cheng et al. (2021) se explica una configuración, *Secureboost*, que preserva la privacidad de los datos en un entorno federado, guiándose por las directivas de protección de datos a nivel europeo (*GDPR*). En Sattler et al. (2020), por ejemplo, se atajan los incrementos de intercambio de información (tráfico de red) proponiendo un sistema de compresión propio (*STC, sparse ternary compression*). También en Marfoq et al. (2020) y Konečný et al. (2016) proponen algoritmos específicos para mejorar la velocidad del entrenamiento federado y evalúan su desempeño en entornos de red comunes según los estándares actuales.

Hay que tener en cuenta que cualquier implementación más realista con la librería *TFF* puede requerir cambios en la programación, según aconseja o apunta la propia Google Google (2021).

Por último, quedaría hablar de la generalidad de los modelos resultantes en este proyecto. Hemos hecho notar que el conjunto de datasets de que se ha dispuesto para el entrenamiento es relativamente escaso. ¿Cuál es, pues, la capacidad de generalización de los modelos finales? ¿Cómo se desempeñarán con muestras no vistas antes? Liu et al. (2021) pone el foco en ese riesgo de aplicar un modelo entrenado mediante aprendizaje federado en un dominio nuevo, y propone un modelo más complejo que permite a cada cliente federado acceder a distribuciones multi-fuente durante el entrenamiento.

Bibliografía

Abadi, M., Chu, A., Goodfellow, I., McMahan, H. B., Mironov, I., Talwar, K., and Zhang, L. (2016). Deep learning with differential privacy. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (ACM CCS), pp. 308-318, 2016*.

Abraham, N. and Khan, N. M. (2019). A novel focal tversky loss function with improved attention u-net for lesion segmentation. In *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*. IEEE.

Acuratio (2022). Acuratio. `https://acuratio.com/` Fecha de visualización: 2022-01-26.

Al-Dhabyani, W., Gomaa, M., Khaled, H., and Fahmy, A. (2020). Dataset of breast ultrasound images. *Data in Brief*, 28:104863.

Arivazhagan, M. G., Aggarwal, V., Singh, A. K., and Choudhary, S. (2019). Federated learning with personalization layers. *arXiv:1912.00818*.

Bagdasaryan, E., Veit, A., Hua, Y., Estrin, D., and Shmatikov, V. (2020). How to backdoor federated learning. In Chiappa, S. and Calandra, R., editors, *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2938–2948. PMLR.

Bannier, E., Barker, G., Borghesani, V., Broeckx, N., Clement, P., Emblem, K. E., Ghosh, S., Glerean, E., Gorgolewski, K. J., Havu, M., Halchenko, Y. O., Herholz, P., Hespel, A., Heunis, S., Hu, Y., Hu, C.-P., Huijser, D., Vayá, M. I., Jancalek, R., Katsaros, V. K., Kieseler, M.-L., Maumet, C., Moreau, C. A., Mutsaerts, H.-J., Oostenveld, R., Ozturk-Isik, E., Espinosa, N. P. L., Pellman, J., Pernet, C. R., Pizzini, F. B., Trbalić, A. Š., Toussaint, P.-J., di Oleggio Castello, M. V., Wang, F., Wang, C., and Zhu, H. (2021). The open brain consent: Informing research participants and obtaining consent to share brain imaging data. *Human Brain Mapping*, 42(7):1945–1951.

Benjamin Planche, E. A. (2019). *Hands-On Computer Vision with TensorFlow 2*. Packt Publishing.

Berg, W. A., Gutierrez, L., NessAiver, M. S., Carter, W. B., Bhargavan, M., Lewis, R. S., and Ioffe, O. B. (2004). Diagnostic accuracy of mammography, clinical examination, US, and MR imaging in preoperative assessment of breast cancer. *Radiology*, 233(3):830–849.

Bnsreenu (2021). U-net: implementación de arquitectura (i). https://github.com/bnsreenu/python_for_microscopists/blob/master/208_multiclass_Unet_sandstone.py Fecha de publicación: 2021-03-11.

Byra, M., Galperin, M., Ojeda-Fournier, H., Olson, L., O'Boyle, M., Comstock, C., and Andre, M. (2019). Breast mass classification in sonography with transfer learning using a deep convolutional neural network and color conversion. *Medical Physics*, 46(2):746–755.

Byra, M., Jarosik, P., Szubert, A., Galperin, M., Ojeda-Fournier, H., Olson, L., O'Boyle, M., Comstock, C., and Andre, M. (2020). Breast mass segmentation in ultrasound with selective kernel u-net convolutional neural network. *Biomedical Signal Processing and Control*, 61:102027.

Chang, Q., Qu, H., Zhang, Y., Sabuncu, M., Chen, C., Zhang, T., and Metaxas, D. (2020). Synthetic learning: Learn from distributed asynchronized discriminator gan without sharing medical image data. *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 13856-13866*.

Chen, K., Guo, Y., Yang, C., Xu, Y., Zhang, R., Li, C., and Wu, R. (2021). Enhanced breast lesion classification via knowledge guided cross-modal and semantic data augmentation. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2021*, pages 53–63. Springer International Publishing.

Cheng, H., Shan, J., Ju, W., Guo, Y., and Zhang, L. (2010). Automated breast cancer detection and classification using ultrasound images: A survey. *Pattern Recognition*, 43(1):299–317.

Cheng, K., Fan, T., Jin, Y., Liu, Y., Chen, T., Papadopoulos, D., and Yang, Q. (2021). SecureBoost: A lossless federated learning framework. *IEEE Intelligent Systems*, 36(6):87–98.

Chollet, F. (2016). Xception: Deep learning with depthwise separable convolutions. *arXiv:1610.02357v3*.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302.

Dong, M., Lu, X., Ma, Y., Guo, Y., Ma, Y., and Wang, K. (2015). An efficient approach for automated mass segmentation and classification in mammograms. *Journal of Digital Imaging*, 28(5):613–625.

Dwork, C. (2011). A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95.

Dwork, C. and Roth, A. (2013). The algorithmic foundations of differential privacy. *Foundations and Trends® in Theoretical Computer Science*, 9(3-4):211–407.

Ek, S., Portet, F., Lalanda, P., and Vega, G. (2021). A Federated Learning Aggregation Algorithm for Pervasive Computing: Evaluation and Comparison. In *19th IEEE International Conference on Pervasive Computing and Communications PerCom 2021*, Kassel (virtual), Germany.

Eroğlu, Y., Yildirim, M., and Çinar, A. (2021). Convolutional neural networks based classification of breast ultrasonography images by hybrid method with respect to benign, malignant, and normal using mRMR. *Computers in Biology and Medicine*, 133:104407.

Fallah, A., Mokhtari, A., and Ozdaglar, A. (2020). Personalized federated learning: A meta-learning approach. *arXiv:2002.07948v4*.

FedJAX (2022). Fedjax. `https://github.com/google/fedjax?ref=pythonrepo.com` Fecha de visualización: 2022-01-26.

FedML (2022). Fedml. `https://fedml.ai/` Fecha de visualización: 2022-01-26.

FeTS (2022). The federated tumor segmentation (fets) initiative. `https://www.med.upenn.edu/cbica/fets/` Fecha de visualización: 2022-01-26.

FeTSChallenge (2022). Federated tumor segmentation challenge 2021. `https://fets-ai.github.io/Challenge/` Fecha de visualización: 2022-01-26.

Flower (2022). Flower. `https://flower.dev/` Fecha de visualización: 2022-01-26.

Fujioka, T., Mori, M., Kubota, K., Oyama, J., Yamaga, E., Yashima, Y., Katsuta, L., Nomura, K., Nara, M., Oda, G., Nakagawa, T., Kitazume, Y., and Tateishi, U. (2020). The utility of deep learning in breast ultrasonic imaging: A review. *Diagnostics*, 10(12):1055.

Girshick, R. (2015). Fast R-CNN. *arXiv:1504.08083v2*.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv:1311.2524v5*.

Google (2017). Federated learning: Collaborative machine learning without centralized training data. `https://ai.googleblog.com/2017/04/federated-learning-collaborative.html` Fecha de publicación: 2017-04-06.

Google (2021). tff.simulation.datasets.clientdata. `https://www.tensorflow.org/federated/api_docs/python/tff/simulation/datasets/ClientData` Fecha de publicación: 2021-09-02.

Google (2022a). Tensorflow federated. `https://www.tensorflow.org/federated` Fecha de visualización: 2022-01-26.

Google (2022b). Tensorflow federated research. https://github.com/google-research/federated Fecha de visualización: 2022-01-26.

Google (2022c). Tuning recommended aggregations for learning. https://www.tensorflow.org/federated/tutorials/tuning_recommended_aggregators Fecha de publicación: 2022-01-06.

Google (2022d). Tutorial google "building your own federated learning algorithm". https://www.tensorflow.org/federated/tutorials/building_your_own_federated_learning_algorithm Fecha de publicación: 2022-01-06.

Google (2022e). Tutorial google çustom federated algorithms, part 1: Introduction to the federated core". https://www.tensorflow.org/federated/tutorials/custom_federated_algorithms_1 Fecha de publicación: 2022-01-06.

Google (2022f). Tutorial google çustom federated algorithms, part 2: Implementing federated averaging". https://www.tensorflow.org/federated/tutorials/custom_federated_algorithms_2 Fecha de publicación: 2022-01-06.

Google (2022g). Tutorial google "differential privacy in tff". https://www.tensorflow.org/federated/tutorials/federated_learning_with_differential_privacy Fecha de publicación: 2022-01-06.

Google (2022h). Tutorial google ïmplementing custom aggregations". https://www.tensorflow.org/federated/tutorials/custom_aggregators Fecha de publicación: 2022-01-06.

Gupta, A., Harrison, P. J., Wieslander, H., Pielawski, N., Kartasalo, K., Partel, G., Solorzano, L., Suveer, A., Klemm, A. H., Spjuth, O., Sintorn, I.-M., and Wählby, C. (2018). Deep learning in image cytometry: A review. *Cytometry Part A*, 95(4):366–380.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask R-CNN. *arXiv:1703.06870v3*.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv:1512.03385v1*.

He, Y. (2021). Awesome federated learning. https://github.com/chaoyanghe/Awesome-Federated-Learning Fecha de publicación: 2021-10-21.

Hsu, T.-M. H., Qi, H., and Brown, M. (2020). Federated visual classification with real-world data distribution. In *Computer Vision – ECCV 2020*, pages 76–92. Springer International Publishing.

Hu, J., Shen, L., Albanie, S., Sun, G., and Wu, E. (2017). Squeeze-and-excitation networks. *arXiv:1709.01507v4*.

Iglovikov, V. and Shvets, A. (2018). TernausNet: U-Net with VGG11 Encoder Pre-Trained on ImageNet for Image Segmentation. *arXiv:1801.05746v1*.

ILSVRC (2022). Ilsvrc del imagenet challenge. `https://image-net.org/challenges/LSVRC/index.php` Fecha de visualización: 2022-01-25.

Ivan Vasilev, Daniel Slater, G. S. (2019). *Python Deep Learning - Second Edition*. Packt Publishing.

Jaccard, P. (1912). THE DISTRIBUTION OF THE FLORA IN THE ALPINE ZONE.1. *New Phytologist*, 11(2):37–50.

Jadon, S. (2020). A survey of loss functions for semantic segmentation. *2020 IEEE International Conference on Computational Intelligence in Bioinformatics and Computational Biology*.

Jadon, S. (2021). SemSegLoss: A python package of loss functions for semantic segmentation. *Software Impacts, Volume 9, 2021,100078, ISSN 2665-9638*.

Kaggle (2021). U-net: implementación de arquitectura (ii). `https://www.kaggle.com/shiratorizawa/u-net-mask-from-actual-images` Versión 12. Fecha de publicación: 2021-06-23.

Kairouz, P., McMahan, H. B., Avent, B., Bellet, A., Bennis, M., Bhagoji, A. N., Bonawitz, K., Charles, Z., Cormode, G., Cummings, R., D'Oliveira, R. G. L., Eichner, H., Rouayheb, S. E., Evans, D., Gardner, J., Garrett, Z., Gascón, A., Ghazi, B., Gibbons, P. B., Gruteser, M., Harchaoui, Z., He, C., He, L., Huo, Z., Hutchinson, B., Hsu, J., Jaggi, M., Javidi, T., Joshi, G., Khodak, M., Konečný, J., Korolova, A., Koushanfar, F., Koyejo, S., Lepoint, T., Liu, Y., Mittal, P., Mohri, M., Nock, R., Özgür, A., Pagh, R., Raykova, M., Qi, H., Ramage, D., Raskar, R., Song, D., Song, W., Stich, S. U., Sun, Z., Suresh, A. T., Tramèr, F., Vepakomma, P., Wang, J., Xiong, L., Xu, Z., Yang, Q., Yu, F. X., Yu, H., and Zhao, S. (2019). Advances and open problems in federated learning. *Foundations and Trends in Machine Learning Vol 14 Issue 1-2*.

Kaissis, G. A., Makowski, M. R., Rückert, D., and Braren, R. F. (2020). Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311.

Kelly, K. M., Dean, J., Comulada, W. S., and Lee, S.-J. (2009). Breast cancer detection using automated whole breast ultrasound and mammography in radiographically dense breasts. *European Radiology*, 20(3):734–742.

Konečný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., and Bacon, D. (2016). Federated learning: Strategies for improving communication efficiency. *arXiv:1610.05492v2*.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Leaf (2022). Leaf. `https://leaf.cmu.edu/` Fecha de visualización: 2022-01-26.

Lecun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, D., Kar, A., Ravikumar, N., Frangi, A. F., and Fidler, S. (2020). Federated simulation for medical imaging. In *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, pages 159–168. Springer International Publishing.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2018). Federated optimization in heterogeneous networks. *arXiv:1812.06127v5*.

Li, T., Sanjabi, M., Beirami, A., and Smith, V. (2019a). Fair resource allocation in federated learning. *arXiv:1905.10497v2*.

Li, W., Milletarì, F., Xu, D., Rieke, N., Hancox, J., Zhu, W., Baust, M., Cheng, Y., Ourselin, S., Cardoso, M. J., and Feng, A. (2019b). Privacy-preserving federated brain tumour segmentation. In *Machine Learning in Medical Imaging*, pages 133–141. Springer International Publishing.

Liu, B., Cheng, H., Huang, J., Tian, J., Tang, X., and Liu, J. (2010). Fully automatic and segmentation-robust classification of breast tumors based on local texture analysis of ultrasound images. *Pattern Recognition*, 43(1):280–298.

Liu, Q., Chen, C., Qin, J., Dou, Q., and Heng, P.-A. (2021). FedDG: Federated Domain Generalization on Medical Image Segmentation via Episodic Learning in Continuous Frequency Space. *arXiv:2103.06030v1*.

Liu, T. (2021). Recurso gráfico: aprendizaje federado. `https://ml.berkeley.edu/blog/posts/federated/` Fecha de publicación: 2021-03-16.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2015). SSD: Single Shot MultiBox Detector. *Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9905. Springer, Cham*.

Long, J., Shelhamer, E., and Darrell, T. (2014). Fully convolutional networks for semantic segmentation. *arXiv:1411.4038v2*.

Ma, J. (2021). Loss functions for image segmentation. `https://medium.com/@junma11/loss-functions-for-medical-image-segmentation-a-taxonomy-cefa5292eec0` `https://github.com/JunMa11/SegLoss` Fecha de publicación: 2021-11-10.

Marfoq, O., Xu, C., Neglia, G., and Vidal, R. (2020). Throughput-optimal topology design for cross-silo federated learning. *arXiv:2010.12229v2*.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., and y Arcas, B. A. (2016). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017. JMLR: W&CP volume 54*.

Mejri, M., Mejri, A., Mejri, O., and Fekih, C. (2021). Semantic segmentation and object detection towards instance segmentation: Breast tumor identification. *arXiv:2108.03287v1*.

Melis, L., Song, C., Cristofaro, E. D., and Shmatikov, V. (2019). Exploiting unintended feature leakage in collaborative learning. In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE.

Michelucci, U. (2019). *Advanced Applied Deep Learning*. Springer-Verlag GmbH.

Mohri, M., Sivek, G., and Suresh, A. T. (2019). Agnostic federated learning. In Chaudhuri, K. and Salakhutdinov, R., editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4615–4625. PMLR.

Ning, Z., Zhong, S., Feng, Q., Chen, W., and Zhang, Y. (2021). SMU-net: Saliency-guided morphology-aware u-net for breast lesion segmentation in ultrasound image. *IEEE Transactions on Medical Imaging*, pages 1–1.

NVidia (2019). Recurso gráfico: aprendizaje federado. `https://blogs.nvidia.com/blog/2019/10/13/what-is-federated-learning` Fecha de publicación: 2019-10-13.

OpenMined (2020). Recurso gráfico: aprendizaje federado. `https://www.youtube.com/watch?v=YnzoxibzkdI` Fecha de publicación: 2020-11-12.

OpenMined (2022). Pysyft. `https://github.com/OpenMined/PySyft` Fecha de visualización: 2022-01-26.

Pillutla, K., Kakade, S. M., and Harchaoui, Z. (2019). Robust aggregation for federated learning. *arXiv:1912.13445v2*.

Ponraj, D., Jenifer, M., Poongodi, P., and Manoharan, S. (2011). A survey on the preprocessing techniques of mammogram for the detection of breast cancer. *Journal of Emerging Trends in Computing and Information Sciences*, 2.

Punn, N. S. and Agarwal, S. (2021). RCA-IUnet: A residual cross-spatial attention guided inception U-Net model for tumor segmentation in breast ultrasound imaging. *Machine Vision and Applications, Springer, 2022*.

Ragab, D. A., Sharkas, M., Marshall, S., and Ren, J. (2019). Breast cancer detection using deep convolutional neural networks and support vector machines. *PeerJ*, 7:e6201.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *arXiv:1506.02640v5*.

Redmon, J. and Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. *arXiv:1612.08242v1*.

Redmon, J. and Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *arXiv:1804.02767v1*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. *arXiv:1506.01497v3*.

Rodrigues, P. S. (2017). Breast ultrasound image. Mendeley.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *Lecture Notes in Computer Science*, pages 234–241. Springer International Publishing.

Sattler, F., Wiedemann, S., Muller, K.-R., and Samek, W. (2020). Robust and communication-efficient federated learning from non-i.i.d. data. *IEEE Transactions on Neural Networks and Learning Systems*, 31(9):3400–3413.

Sheller, M. J., Edwards, B., Reina, G. A., Martin, J., Pati, S., Kotrotsou, A., Milchenko, M., Xu, W., Marcus, D., Colen, R. R., and Bakas, S. (2020). Federated learning in medicine: facilitating multi-institutional collaborations without sharing patient data. *Scientific Reports*, 10(1).

Sheller, M. J., Reina, G. A., Edwards, B., Martin, J., and Bakas, S. (2019). Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, pages 92–104. Springer International Publishing.

Shi, J., Zhou, S., Liu, X., Zhang, Q., Lu, M., and Wang, T. (2016). Stacked deep polynomial network based representation learning for tumor classification with small ultrasound image dataset. *Neurocomputing*, 194:87–94.

Siddhesh (2021). Federatedlearningvesselsegmentation. `https://github.com/siddhesh1598/FederatedLearningVesselSegmentation` Fecha de publicación: 2021-05-15.

Siegel, R. L., Miller, K. D., and Jemal, A. (2016). Cancer statistics, 2016. *CA: A Cancer Journal for Clinicians*, 66(1):7–30.

Siegel, R. L., Miller, K. D., and Jemal, A. (2019). Cancer statistics, 2019. *CA: A Cancer Journal for Clinicians*, 69(1):7–34.

Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv:1409.1556v6*.

SORENSEN, T. A. (1948). A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biol. Skar.*, 5:1–34.

StackOverflow (2017). U-net: implementación de arquitectura (iii). `https://stackoverflow.com/questions/44014534/why-could-u-net-mask-image-with-smaller-mask` Fecha de publicación: 2017-05-17.

Stavros, A. T. (2004). *Breast ultrasound*. Lippincott Williams & Wilkins.

Sung, H., Ferlay, J., Siegel, R. L., Laversanne, M., Soerjomataram, I., Jemal, A., and Bray, F. (2021). Global cancer statistics 2020: GLOBOCAN estimates of incidence and mortality worldwide for 36 cancers in 185 countries. *CA: A Cancer Journal for Clinicians*, 71(3):209–249.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Truex, S., Baracaldo, N., Anwar, A., Steinke, T., Ludwig, H., Zhang, R., and Zhou, Y. (2019). A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security - AISec'19*. ACM Press.

Vepakomma, P., Gupta, O., Swedish, T., and Raskar, R. (2018). Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv:1812.00564v1*.

Wang, H., Sreenivasan, K., Rajput, S., Vishwakarma, H., Agarwal, S., yong Sohn, J., Lee, K., and Papailiopoulos, D. (2020a). Attack of the tails: Yes, you really can backdoor federated learning. *arXiv:2007.05084v1*.

Wang, H., Yurochkin, M., Sun, Y., Papailiopoulos, D., and Khazaeni, Y. (2020b). Federated learning with matched averaging. *arXiv:2002.06440v1*.

Wei, K., Li, J., Ding, M., Ma, C., Yang, H. H., Farokhi, F., Jin, S., Quek, T. Q. S., and Poor, H. V. (2020). Federated learning with differential privacy: Algorithms and performance analysis. *IEEE Transactions on Information Forensics and Security*, 15:3454–3469.

Wen, Y. (2021). Medical datasets collections for artificial intelligence-based medical image analysis. *arXiv:2102.01549v3*.

Wu, X., Liang, Z., and Wang, J. (2020). FedMed: A federated learning framework for language modeling. *Sensors*, 20(14):4048.

Xian, M., Zhang, Y., Cheng, H. D., Xu, F., Huang, K., Zhang, B., Ding, J., Ning, C., and Wang, Y. (2018). BUSIS: A Benchmark for Breast Ultrasound Image Segmentation. *arXiv:1801.03182v2*.

Yap, M. H., Edirisinghe, E., and Bez, H. (2010). Processed images in human perception: A case study in ultrasound breast imaging. *European Journal of Radiology*, 73(3):682–687.

Yap, M. H., Goyal, M., Osman, F., Martí, R., Denton, E., Juette, A., and Zwiggelaar, R. (2020). Breast ultrasound region of interest detection and lesion localisation. *Artificial Intelligence in Medicine*, 107:101880.

Yap, M. H., Pons, G., Marti, J., Ganau, S., Sentis, M., Zwiggelaar, R., Davison, A. K., and Marti, R. (2018). Automated breast ultrasound lesions detection using convolutional neural networks. *IEEE Journal of Biomedical and Health Informatics*, 22(4):1218–1226.

Yu, K., Chen, S., and Chen, Y. (2021). Tumor segmentation in breast ultrasound image by means of res path combined with dense connection neural network. *Diagnostics*, 11(9):1565.

Yurochkin, M., Agarwal, M., Ghosh, S., Greenewald, K., Hoang, N., and Khazaeni, Y. (2019). Probabilistic federated neural matching. ICLR 2019 Conference Blind Submission.

Zeiler, M. D. and Fergus, R. (2014). Visualizing and understanding convolutional networks. In *Computer Vision – ECCV 2014*, pages 818–833. Springer International Publishing.

Zeng, D., Liang, S., Hu, X., and Xu, Z. (2021). FedLab: A Flexible Federated Learning Framework. *arXiv:2107.11621v3*.

Zhuang, W., Gan, X., Wen, Y., and Zhang, S. (2021). Easyfl: A low-code federated learning platform for dummies. *IEEE Internet of Things Journal (Early Access) 2022*.

Anexo A

Apéndice A: gráficas de entrenamiento

A.1. Entrenamiento de modelos *DL*: resultados completos

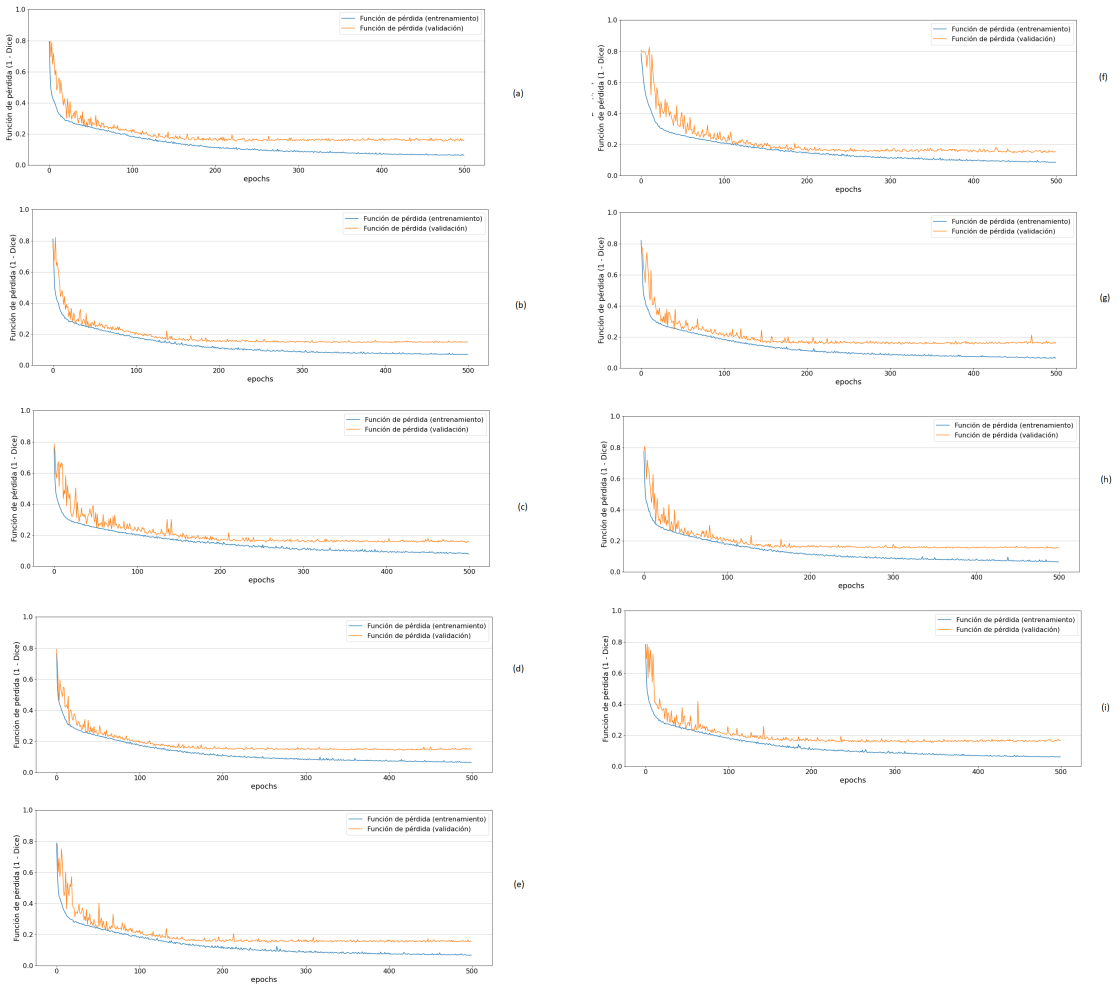

Figura A.1: Función de pérdida (*1 – Dice*) durante el entrenamiento federado (3 clientes): tests #0 (a), #1 (b), #2 (c), #3 (d), #4 (e), #5 (f), #6 (g), #7 (h), #8 (i).

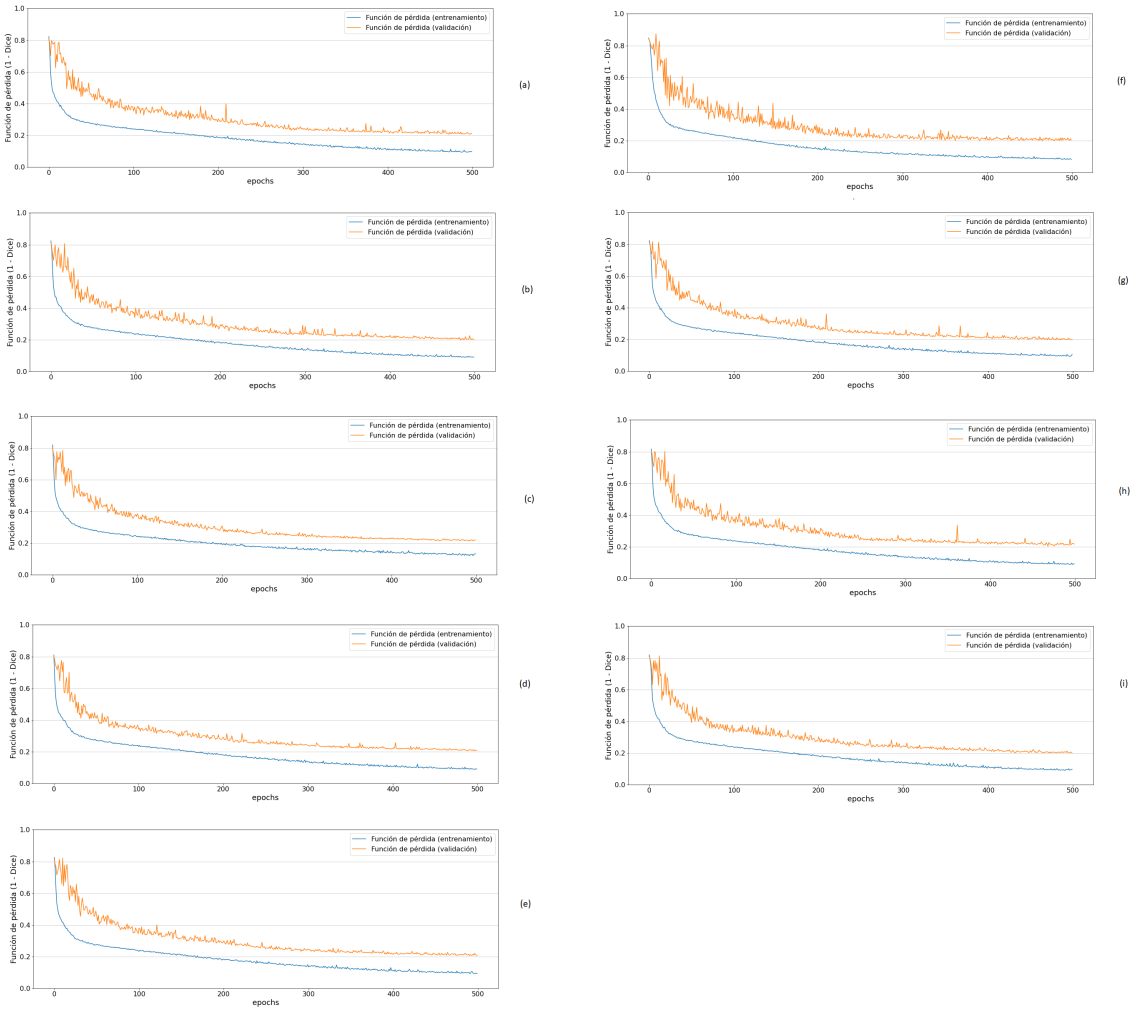

Figura A.2: Función de pérdida (*1 – Dice*) durante el entrenamiento federado (5 clientes): tests #0 (a), #1 (b), #2 (c), #3 (d), #4 (e), #5 (f), #6 (g), #7 (h), #8 (i).

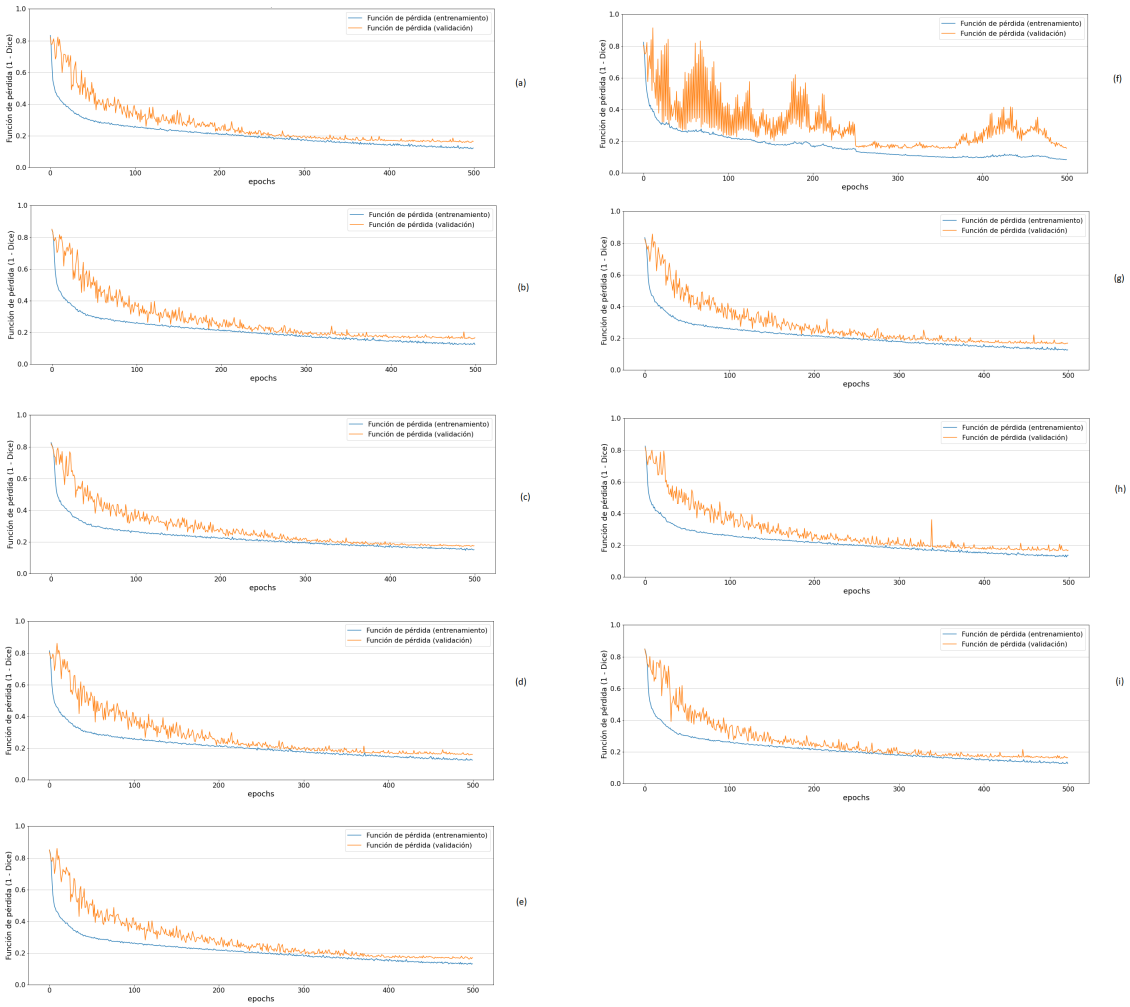

Figura A.3: Función de pérdida (*1 – Dice*) durante el entrenamiento federado (7 clientes): tests #0 (a), #1 (b), #2 (c), #3 (d), #4 (e), #5 (f), #6 (g), #7 (h), #8 (i).

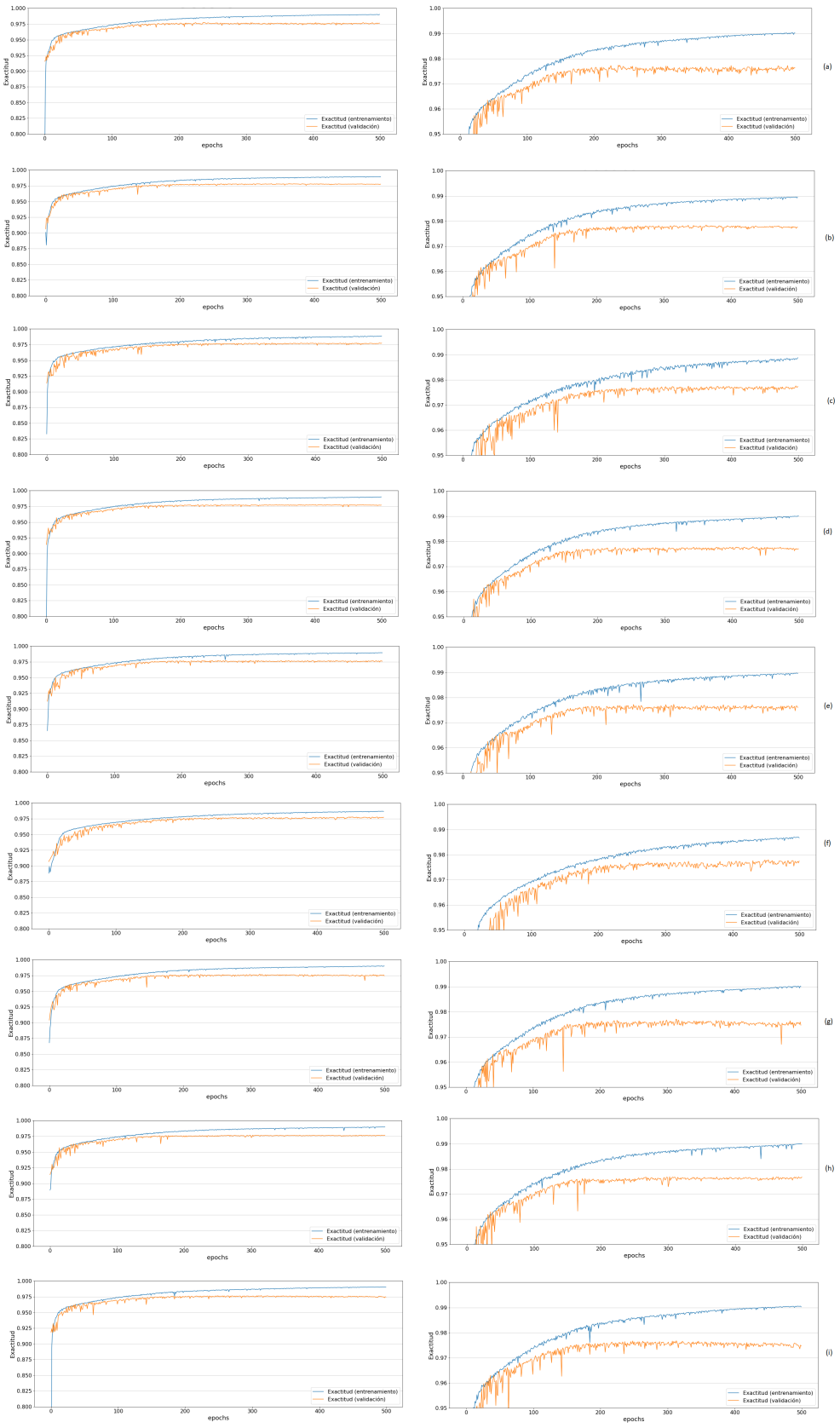

Figura A.4: Exactitud (*accuracy*) durante el entrenamiento federado (3 clientes): tests #0 (a), #1 (b), #2 (c), #3 (d), #4 (e), #5 (f), #6 (g), #7 (h), #8 (i). La columna izquierda muestra la gráfica completa; la columna de la derecha, la gráfica aumentada en el rango $>95\,\%$.

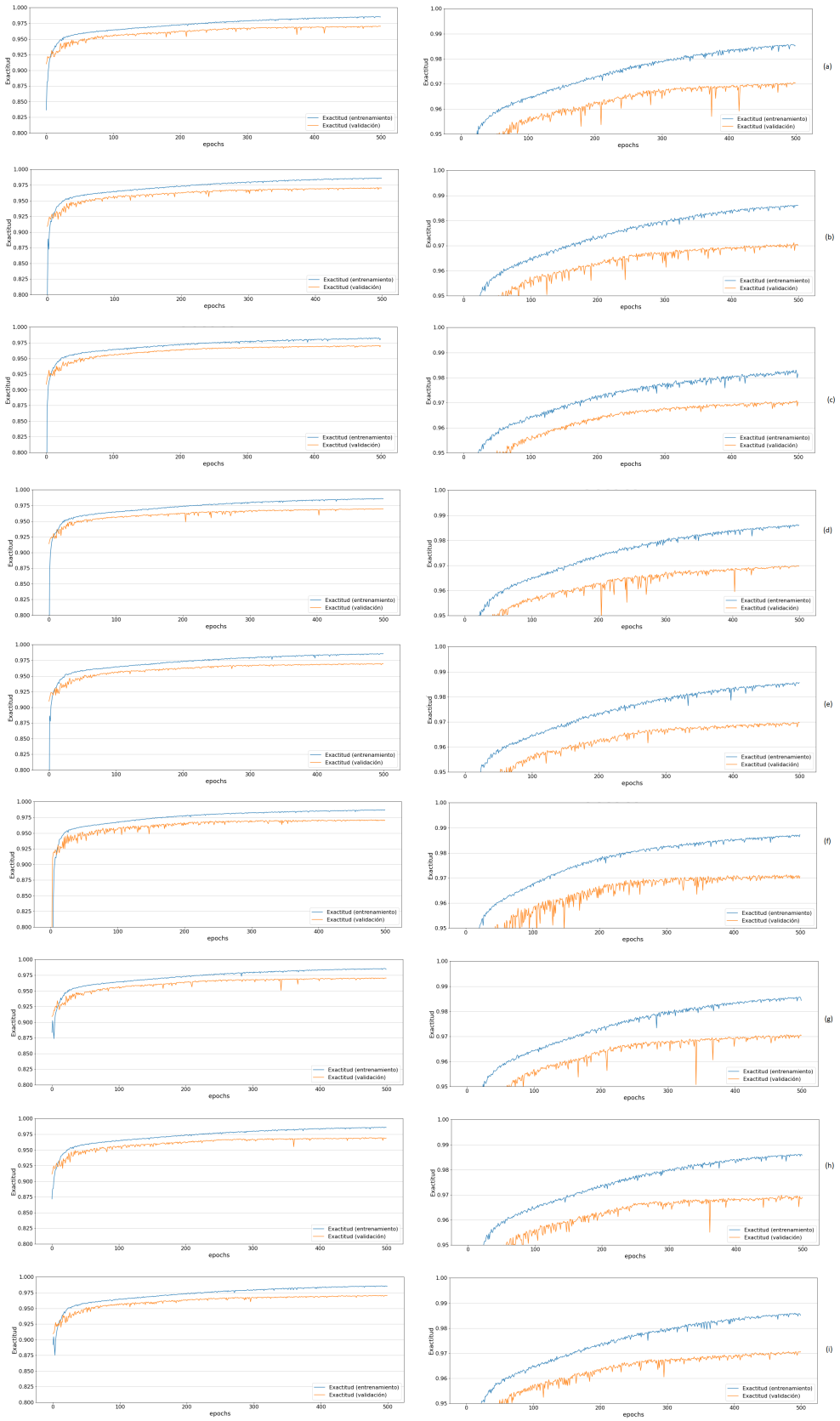

Figura A.5: Exactitud (*accuracy*) durante el entrenamiento federado (5 clientes): tests #0 (a), #1 (b), #2 (c), #3 (d), #4 (e), #5 (f), #6 (g), #7 (h), #8 (i). La columna izquierda muestra la gráfica completa; la columna de la derecha, la gráfica aumentada en el rango $>95\,\%$.

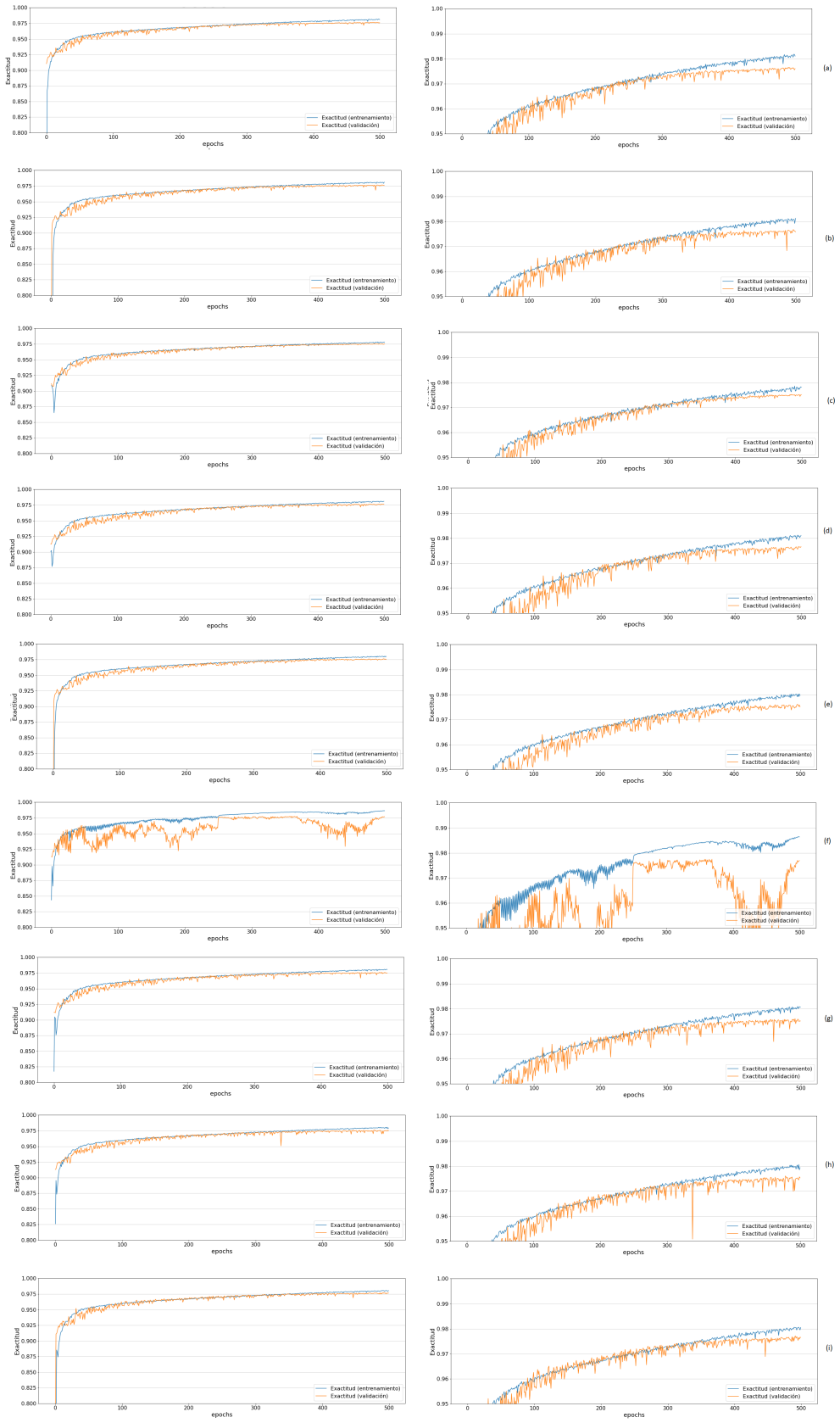

Figura A.6: Exactitud (*accuracy*) durante el entrenamiento federado (7 clientes): tests #0 (a), #1 (b), #2 (c), #3 (d), #4 (e), #5 (f), #6 (g), #7 (h), #8 (i). La columna izquierda muestra la gráfica completa; la columna de la derecha, la gráfica aumentada en el rango $>95\,\%$.

Anexo B

Apéndice B: gráficas de entrenamiento (II)

B.1. Entrenamiento de modelo clásico prolongado

Un entrenamiento clásico de 1000 *epochs* consigue mejores resultados en todas métricas: coeficiente *Dice* (80,63 %, +1,09 puntos respecto del modelo clásico); Exactitud (96,78 %, +1,24); coeficiente *IoU* (80,85 %, +0,58) y sensibilidad (78,85 %, +0,59). Las figuras B.1 y B.2 muestran la evolución del entrenamiento.

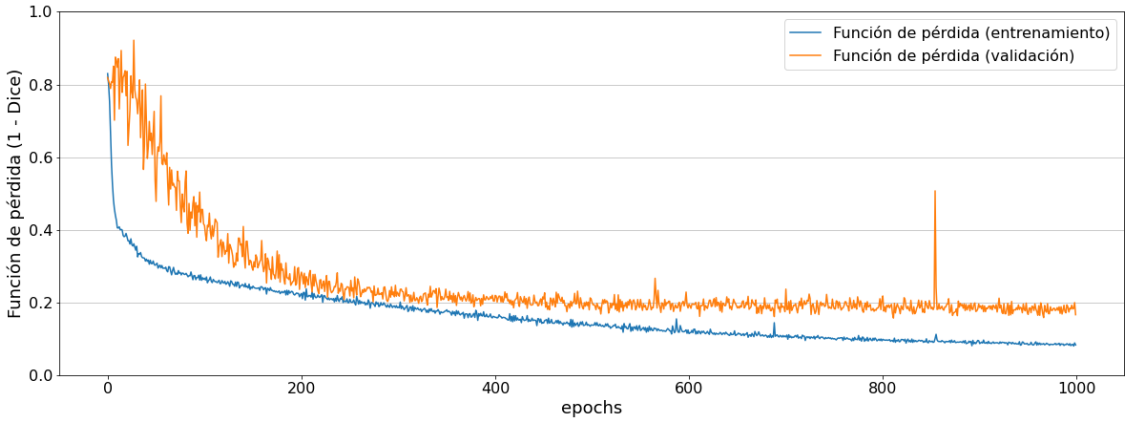

Figura B.1: Función de pérdida durante el entrenamiento clásico prolongado (1000 *epochs*).

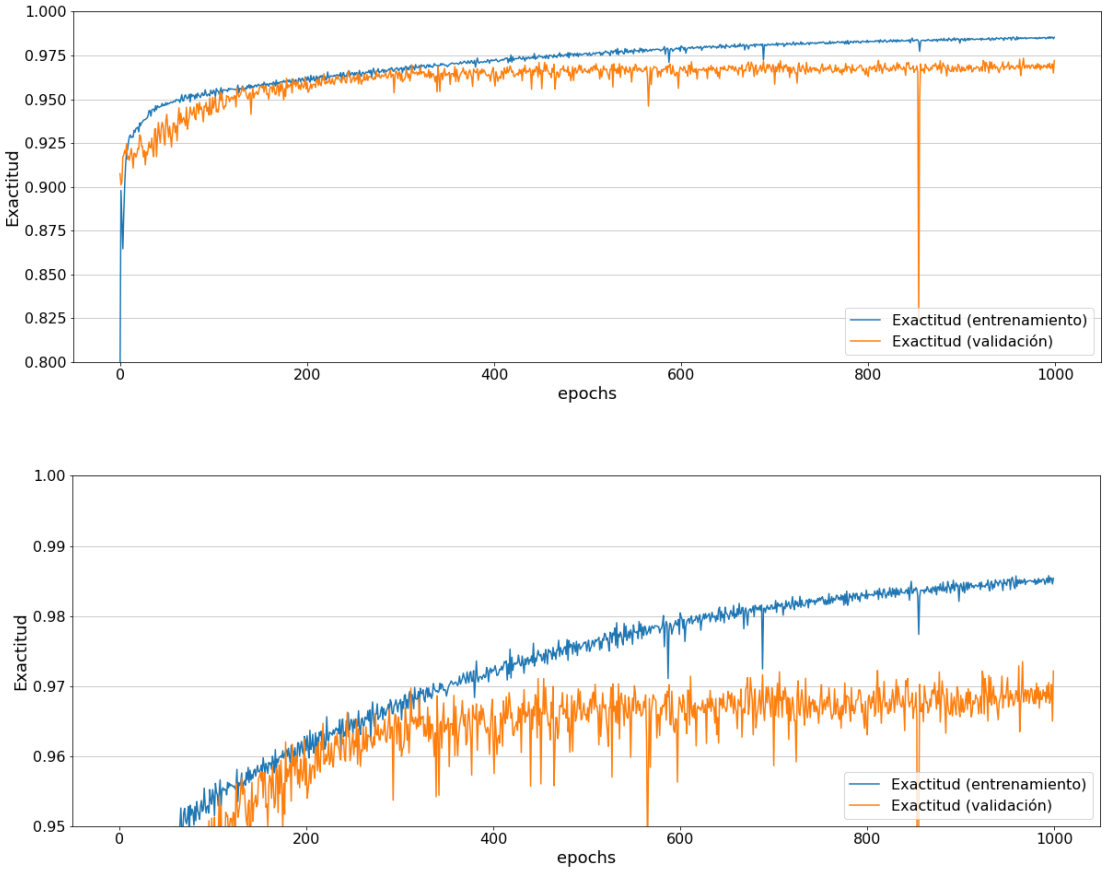

Figura B.2: Exactitud (*accuracy*) durante el entrenamiento clásico prolongado (1000 *epochs*). De arriba abajo: gráfica general y gráfica aumentada en el rango $>95\,\%$.