UNIVERSITAT POLITÈCNICA DE VALÈNCIA

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escuela Técnica Superior de Ingeniería Industrial

Estudio y desarrollo de transfer learning aplicado a la segmentación automática de estructuras morfológicas para el apoyo, el análisis y el diagnóstico médico.

Trabajo Fin de Grado

Grado en Ingeniería Biomédica

AUTOR/A: Aviñó Nouselles, José Ramón

Tutor/a: Bosch Roig, Ignacio

Cotutor/a externo: TEN ESTEVE, AMADEO

CURSO ACADÉMICO: 2021/2022

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

ESCUELA TÉCNICA SUPERIOR INGENIERÍA INDUSTRIAL VALENCIA

TRABAJO FIN DE GRADO EN INGENIERÍA BIOMÉDICA

ESTUDIO Y DESARROLLO DE TRANSFER LEARNING APLICADO A LA SEGMENTACIÓN AUTOMÁTICA DE ESTRUCTURAS MORFOLÓGICAS PARA EL APOYO, EL ANÁLISIS Y EL DIAGNÓSTICO MÉDICO

AUTOR: AVIÑÓ NOUSELLES, JOSÉ RAMÓN

TUTOR: BOSCH ROIG, IGNACIO

COTUTOR: TEN ESTEVE, AMADEO

Curso Académico: 2021-22

Agradecimientos

A Amadeo y a Ignacio, por guiarme en este bonito proceso de aprendizaje y tener la paciencia necesaria para solucionar cada una de mis dudas.

A Diana y Leo, por aportar sus conocimientos y su ayuda, así como su actitud positiva y motivadora.

A mi familia, por apoyarme en los momentos más complicados. En momentos de desesperación ahí estaban ellos para hacerme creer en mí, y en el proceso.

A mi hermano, por enseñarme a levantar hierros para desconectar y poder volver con más ganas.

A Marta, por recordarme siempre cuales son mis obligaciones como estudiante y animarme a cumplirlas.

A mis amigos que he conocido en la carrera, y que me han enseñado a darle importancia a lo importante.

Y en especial, a mi madre, por inculcarme valores como el esfuerzo, la disciplina y la perseverancia, siendo ella un claro ejemplo de todos ellos.

Simplemente, gracias a todos los que en algún momento me habéis ayudado, regalándome vuestro tiempo y atención.

Resumen

En el campo de la Imagen Médica, la segmentación morfológica es una fase muy importante dentro del tratamiento, análisis y diagnóstico requerido en diversas aplicaciones clínicas. En la actualidad, es usual encontrar que, en la mayoría de los centros dedicados a este tipo de tarea, la segmentación es llevada a cabo manualmente por un experto, lo cual se traduce en una gran inversión de tiempo que podría ser invertido en otras fases de mayor relevancia en el estudio de la enfermedad, como puede ser la de análisis y diagnóstico de los resultados, además dicha segmentación manual ésta expuesta a factores de error humano.

La creación de modelos de segmentación automática mediante el uso redes neuronales convolucionales (CNN) requiere por lo general gran cantidad de muestras (imágenes ya segmentadas) y recursos computacionales. Actualmente el paradigma de "transfer learning" está siendo aplicado en el campo de la inteligencia artificial con el fin de reducir tanto la cantidad de datos necesaria como los recursos computacionales para la creación de nuevos modelos, a partir de modelos ya preexistentes que fueron creados con finalidades diferentes, aunque con cierto grado de similitud.

Es por ello, que se plantea el desarrollo de nuevas metodologías de segmentación a partir de imágenes de Resonancia Magnética, completamente automatizada, aplicando el paradigma de "transfer learning" sobre métodos de inteligencia artificial basados en técnicas de aprendizaje profundo (Deep learning) que permita generar y almacenar datos de carácter masivo (Big Data) en los pacientes, con el fin de mejorar en el diagnóstico mediante imagen médica.

Palabras clave: Aprendizaje por transferencia; segmentación semántica; red neuronal convolucional; inteligencia artificial; resonancia magnética, aprendizaje profundo, imagen médica, clasificación.

Resum

En el camp de la Imatge Mèdica, la segmentació morfològica és una fase molt important dins del tractament, anàlisi i diagnòstic requerit en diverses aplicacions clíniques. En l'actualitat, és usual trobar que, en la majoria dels centres dedicats a este tipus de tasca, la segmentació és duta a terme manualment per un expert, la qual cosa es traduït en una gran inversió de temps que podria ser invertit en altres fases de major rellevància en l'estudi de la malaltia, com pot ser la d'anàlisi i diagnòstic dels resultats, a més la dita segmentació manual esta exposada a factors d'error humà.

La creació de models de segmentació automàtica per mitjà de l'ús xarxes neuronals convolucionals (CNN) requereix generalment gran quantitat de mostres (imatges ja segmentades) i recursos computacionals. Actualment el paradigma de "transfer learning" està sent aplicat en el camp de la intel·ligència artificial a fi de reduir tant la quantitat de dades necessàries com els recursos computacionals per a la creació de nous models, a partir de models ja preexistents que van ser creats amb finalitats diferents encara que amb un cert grau de similitud.

És per això, que es planteja el desenrotllament de noves metodologies de segmentació a partir d'imatges de Ressonància Magnètica, completament automatitzada, aplicant el paradigma de "transfer learning" sobre mètodes d'intel·ligència artificial basats en tècniques d'aprenentatge profund (Deep learning) que permet generar i emmagatzemar dades de caràcter massiu (Big Data) en els pacients, a fi de millorar en el diagnòstic per mitjà d'imatge mèdica.

Paraules clau: Aprenentatge per transferència; segmentació semàntica; xarxa neuronal convolucional; intel·ligència artificial; ressonància magnètica; aprenentatge profund; imatge mèdica; classificació.

Abstract

In the field of Medical Imaging, morphological segmentation is a very important phase in the treatment, analysis and diagnosis required in various clinical applications. At present, it is usual to find that, in most of the centers dedicated to this type of task, the segmentation is carried out manually by an expert, which translates into a large investment of time that could be invested in other phases of greater relevance in the study of the disease, such as the analysis and diagnosis of the results, in addition to this manual segmentation is exposed to human error factors.

The creation of automatic segmentation models using convolutional neural networks (CNN) generally requires a large number of samples (already segmented images) and computational resources. Currently, the "transfer learning" paradigm is being applied in the field of artificial intelligence in order to reduce both the amount of data required and the computational resources for the creation of new models, based on pre-existing models that were created for different purposes although with a certain degree of similarity.

Therefore, the development of new segmentation methodologies from Magnetic Resonance images is proposed, completely automated, applying the paradigm of "transfer learning" on artificial intelligence methods based on deep learning techniques (Deep learning) to generate and store massive data (Big Data) in patients, in order to improve diagnosis by medical imaging.

Keywords: Transfer learning; semantic segmentation; convolutional neural network, artificial intelligence; magnetic resonance; deep learning; medical imaging; classification.

Índice general

DOCUMENTOS CONTENIDOS EN EL TFG

- Memoria
- Presupuesto

ÍNDICE DE LA MEMORIA

ÍNDICE DEL PRESUPUESTO

Índice de figuras

Índice de tablas

Parte I

Memoria

Capítulo 1
Introducción

1.1. Motivación

En el campo de la medicina, durante los últimos años, la imagen médica ha traído consigo mayores capacidades en el diagnóstico y seguimiento de los pacientes para hacer frente a las diversas patologías existentes. Su utilidad se ha visto potenciada por el uso de nuevos métodos de adquisición que disminuyen la radiación recibida por el paciente y aumentan la resolución y el análisis computacional de las imágenes, el cual permite extraer biomarcadores de imagen (características cuantitativas y objetivas) de las mismas.

Para la extracción de los biomarcadores de imagen, es necesaria la delimitación / segmentación previa de las regiones de interés (ROI). La segmentación de imágenes consiste en delinear estructuras de interés (ROI) y separarlas del resto de la imagen en función del objetivo deseado, ya sea una simple observación de la estructura o la obtención de datos de esta en un proceso de cuantificación. Tiene una gran variedad de usos, incrementando constantemente, desde planificaciones quirúrgicas o cirugía guiada por imagen hasta la detección de anomalías, y todo esto, de forma no invasiva.

La segmentación puede ser realizada de forma manual por un profesional, quien precisa de conocimientos necesarios para realizarla adecuadamente, o se pueden utilizar técnicas automáticas y/o semiautomáticas. En el caso de la segmentación manual, esta depende en gran parte, de la experiencia profesional del especialista, lo cual introduce una variabilidad considerable entre profesionales además de una cantidad de tiempo no despreciable, que podría ser mejor empleado en otras tareas. Estas dependencias además hacen que no sea viable la aplicación para grandes conjuntos de pacientes (Ker et al., 2017; Lecun et al., 2015; *Segmentación de Imágenes Con Redes Convolucionales*, 2022).

Las técnicas automáticas son muy variadas, y pueden depender de muchos factores, como la intensidad de cada píxel, transformaciones locales, o de reconocimiento de diversos patrones, adaptándose a cada situación determinada. La principal ventaja es que no se necesita un profesional para segmentar manualmente la región de interés, una labor tediosa y que supone mucho tiempo, además de estar sujeta a errores humanos. Por tanto, este profesional podría estar invirtiendo dicho tiempo en otras fases de mayor relevancia en el estudio de la enfermedad y, por ende, mejorar la eficiencia en el entorno clínico.

Tradicionalmente se han empleado técnicas computacionales para la segmentación automática basados en la aplicación secuencial de algoritmos o mediante atlas que han obtenido cierto éxito en imágenes de TC y en la segmentación cerebral en RM. Sin embargo, en otras regiones anatómicas y especialmente en RM estas técnicas no han conseguido tener buenos resultados, por lo que en los últimos años se están empleado técnicas de Inteligencia Artificial (IA) principalmente basados en Redes Neuronales Convolucionales (CNN) que está mejorando los resultados obtenidos por las técnicas de computación tradicionales.

La mayor limitación para la generación de estos modelos de segmentación automática mediante CNN desde cero es la gran cantidad de datos etiquetados que se requieren, así como la complejidad que supone establecer en una nueva arquitectura, las conexiones entre capas y la disposición de estas (Cardenas et al., 2019; Ker et al., 2017; Lecun et al., 2015).

Dado que se destina mucho tiempo y carga computacional en el entrenamiento de redes para un determinado objetivo, el paradigma del "transfer learning" propone, entre otras... trabajar con redes pre-entrenadas en objetivos similares, lo cual hace posible beneficiarse de este conocimiento adquirido previamente para aplicarlo al nuevo objetivo, por ejemplo haciendo uso de los filtros de las primeras capas de la red ya pre-entrenada capaces de detectar bordes, texturas, colores, sombras o elementos primarios pueden ser de gran utilidad para el nuevo objetivo (Torrey & Shavlik, 2009).

La idea general del transfer learning es, por tanto, utilizar el conocimiento obtenido en modelos que han sido desarrollos previamente utilizando grandes cantidades de datos etiquetados en entornos en los cuales el obtener grandes cantidades de datos es una gran limitación. La generación de datos etiquetados es costosa, por lo que es importante aprovechar al máximo los conjuntos de datos existentes (*Transfer Learning: A Beginner's Guide | DataCamp*, 2022).

1.2. Objetivos

El objetivo del presente trabajo es evaluar el rendimiento de modelos de segmentación automática con diferentes épocas, batch size y tasas de aprendizaje. Así como, la aplicación de transfer learning (TL) sobre un modelo previo. Para ello se empleará como métrica de validación, el coeficiente DICE y se seleccionará el mejor modelo para predecir segmentaciones de riñones y bazo. Los modelos segmentarán sobre secuencias MECSE (Multi-Echo Chemical Shift–Encoded gradient echo) y THRIVE (T1 Weighted High Resolution Isotropic Volume Examination) que son dos secuencias de adquisición de imágenes que potencian ciertas características, las cuales se explican posteriormente en el punto 4.1.1. Se evaluará también el rendimiento de los modelos entrenados con ambas secuencias, así como el entrenamiento de un modelo específico para cada secuencia. A continuación, se listan los modelos utilizados:

Modelo 1: Segmenta los riñones sobre secuencias MECSE y THRIVE.

Modelo 2: Segmenta los riñones sobre secuencia MECSE.

Modelo 3: Segmenta los riñones sobre secuencia THRIVE.

Modelo 4: Segmenta el bazo sobre secuencia MECSE y THRIVE.

Modelo 5: Segmenta el bazo sobre secuencia MECSE.

Modelo 6: Segmenta el bazo sobre secuencia THRIVE.

1.3. Inteligencia artificial

En 1956, el término Inteligencia Artificial fue acuñado gracias al informático John McCarthy y la ayuda de Marvin Minsky y Claude Shannon. Este concepto se definió en la conferencia de Dartmouth como la ciencia y la ingeniería de la "construcción de máquinas inteligentes, especialmente de programas informáticos inteligentes" (McCarthy, 2007). Sin embargo, esta definición contiene el concepto de inteligencia, el cual posee diferentes significados y con ello surgen diferentes definiciones como “computadora que imita los procesos intelectuales característicos de los humanos, como la capacidad de razonar, descubrir significado, generalizar o aprender de experiencias pasadas para lograr objetivos sin estar programado explícitamente para una acción específica” (Bali et al., 2019) o “el estudio de los agentes inteligentes: cualquier dispositivo que percibe su entorno y realiza acciones que maximizan su probabilidad de alcanzar con éxito sus objetivos” (Zhou et al., 2019). Generalmente, la mayoría de estas definiciones consisten en una convergencia de conceptos como tecnología, computación, operaciones inteligentes, heurística o resolución de problemas. Por lo tanto, una definición apropiada podría ser “la capacidad de un sistema informático para realizar tareas a través de procesos asociados con agentes inteligentes, como razonar, aprender y tomar decisiones”.

1.3.1. Machine learning

El Machine Learning (ML) o aprendizaje automático es un subcampo de la IA que estudia la capacidad de mejorar el rendimiento basándose en la experiencia (*Artificial Intelligence A Modern Approach Fourth Edition*, 2020). En ML se introducen datos y respuestas como entradas y se reciben reglas como salidas. Se entrena aprendiendo de una gran variedad de ejemplos y encontrando patrones en ellos. De forma que, encontrar y aprender estos patrones permite el desarrollo de reglas que posibilitan la automatización de determinadas tareas. Los proyectos de ML suelen ir desde la comprensión del problema hasta la presentación de los resultados. En la figura 1 se muestran unas pautas para completar un proyecto de ML que incluye hacer preguntas, explorar datos, construir modelos y comprender el resultado y, frecuentemente, repetir esto. Si bien los datos son la clave de los algoritmos de ML, antes de analizar los datos es importante asegurarse de que esté claro el planteamiento del problema y formular la pregunta de manera que se trabaje para obtener una respuesta significativa (Ashenden et al., 2021).

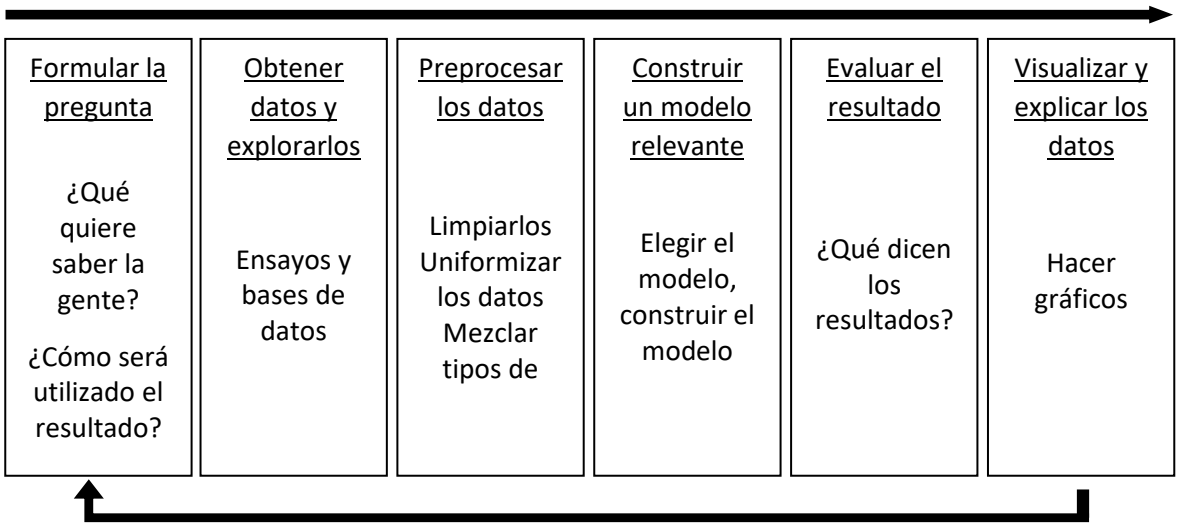

Figura 1. Pautas para completar un proyecto de ML. Obtenido de (Ashenden et al., 2021).

1.3.2. Deep learning

El Deep Learning (DL) o aprendizaje profundo es un subcampo específico del aprendizaje automático: un nuevo método para aprender representaciones a partir de datos que se focaliza en el aprendizaje de sucesivas capas de representaciones cada vez más significativas. El aprendizaje profundo moderno con frecuencia involucra decenas o incluso cientos de sucesivas capas de representaciones, y todas ellas aprenden automáticamente al exponerse a datos de entrenamiento. Mientras tanto, otros enfoques del aprendizaje automático suelen centrarse en el aprendizaje de sólo una o dos capas de representaciones de los datos; de ahí que a veces se les llame aprendizaje superficial.

En el aprendizaje profundo estas representaciones en capas se aprenden casi siempre mediante modelos llamados redes neuronales, con una estructura de capas apiladas unas sobre otras (*Deep Learning with Python FRANÇOIS CHOLLET M A N N I N G SHELTER ISLAND*, 2018).

El aprendizaje profundo se diferencia del aprendizaje automático tradicional en el tipo de datos con los que opera y en las formas en que aprende.

Los algoritmos de aprendizaje automático utilizan datos estructurados y etiquetados para hacer predicciones, lo que significa identificar ciertas características de los datos de entrada del modelo y organizarlos en tablas. Esto no significa necesariamente que no esté utilizando datos no estructurados; simplemente significa que, si lo hace, generalmente pasa por un preprocesamiento para estructurarlos.

El aprendizaje profundo elimina parte del preprocesamiento de datos normalmente asociado con el aprendizaje automático y permitiendo el uso de conjuntos de datos más grandes. Estos algoritmos pueden ingerir y procesar datos no estructurados, como texto e imágenes, y automatizar la extracción de características, eliminando parte de la dependencia de expertos humanos. Por ejemplo, supongamos que tenemos un conjunto de imágenes de diferentes mascotas y queremos categorizarlas por 'gato', 'perro', 'hámster', etc. Los algoritmos de aprendizaje profundo pueden identificar las características (como las orejas) que son más importantes para distinguir un animal de otro. En el aprendizaje automático, la jerarquía de características la configura manualmente un experto humano. Luego, a través de los procesos de descenso de gradiente y retropropagación, el algoritmo de aprendizaje profundo se autocorrige y se ajusta para una mayor precisión, lo que le permite hacer una predicción más precisa sobre nuevas fotos de animales.

Los modelos de aprendizaje automático y aprendizaje profundo también son capaces de realizar varios tipos de aprendizaje, categorizados ampliamente en aprendizaje supervisado, aprendizaje no supervisado y aprendizaje reforzado. El aprendizaje supervisado utiliza conjuntos de datos etiquetados para categorizar o hacer predicciones; requiere algún tipo de intervención humana para etiquetar correctamente los datos de entrada. Por el contrario, el aprendizaje no supervisado no requiere conjuntos de datos etiquetados y, en cambio, detecta patrones en los datos, agrupándolos por cualquier característica distintiva. El aprendizaje por refuerzo es un proceso en el que un modelo aprende a realizar una acción con mayor precisión en un entorno basado en la retroalimentación para maximizar la recompensa (*What Is Deep Learning? | IBM*, 2020)(*Deep Learning with Python FRANÇOIS CHOLLET M A N N I N G SHELTER ISLAND*, 2018) (*What Is Artificial Intelligence (AI)? | IBM*, 2020).

1.3.3. Transfer learning

El transfer learning o aprendizaje por transferencia es la mejora del aprendizaje en una nueva tarea mediante la transferencia de conocimientos de una tarea relacionada que ya se ha aprendido. Los seres humanos parecen tener formas inherentes de transferir conocimiento entre tareas. Es decir, percibimos y aplicamos conocimientos relevantes de experiencias de aprendizaje anteriores cuando nos enfrentamos a nuevas tareas. Cuanto más relacionada esté una tarea con nuestra experiencia previa, más fácilmente podemos dominarla.

En cambio, los algoritmos tradicionales de aprendizaje automático se ocupan habitualmente de tareas aisladas. El aprendizaje por transferencia intenta cambiar esta situación mediante el desarrollo de métodos para transferir los conocimientos aprendidos en una o varias tareas de origen y utilizarlos para mejorar el aprendizaje en una tarea destino relacionada (Torrey & Shavlik, n.d., 2009).

En otras palabras, el aprendizaje por transferencia, inspirado en la capacidad de los seres humanos para transferir conocimientos entre tareas, tiene como objetivo aprovechar los conocimientos de un dominio de origen para mejorar el rendimiento del aprendizaje o reducir la cantidad de ejemplos categorizados requeridos en un dominio de destino. Su uso es fundamental cuando se trata de tareas con ejemplos etiquetados limitados. El aprendizaje por transferencia ha demostrado su amplia aplicabilidad, por ejemplo, en clasificación de imágenes (Long et al., 2015), clasificación de reseñas (Blitzer et al., 2006), sistemas de diálogo (Mo et al., 2016) y la informática urbana (Wei et al., 2016).

Capítulo 2

Marco teórico

2.1 Introducción a las redes neuronales

Las redes neuronales artificiales son técnicas populares de aprendizaje automático que emulan el mecanismo de aprendizaje de los organismos biológicos. En otras palabras, se trata de modelos matemáticos que intentan simular la estructura y funcionalidades de las redes neuronales biológicas.

Las redes neuronales artificiales son redes de múltiples capas de neuronas formadas por nodos, que se utilizan para clasificación y predicción de datos, siempre que reciban una entrada. Cada entrada a un nodo es ponderada con un peso, es decir, multiplicada por un peso individual. En el nodo se suman las entradas ponderadas y el sesgo, en caso de haberlo, para después transmitirse hacia la salida. Basándose en el símil de la red neuronal biológica: las entradas serían las dendritas, los pesos serían las sinapsis, el cuerpo de la neurona sería el nodo y la salida sería el axón. La fuerza de las conexiones sinápticas cambia normalmente en respuesta a estímulos externos, y este cambio es el que produce el aprendizaje en los organismos vivos, de forma similar, las redes neuronales artificiales aprenden a ajustar adecuadamente los pesos para aprender.

Las redes neuronales artificiales pueden considerarse como un algoritmo que relaciona las entradas de la red con las salidas al mismo tiempo que se van optimizando sus parámetros para conseguir que el modelo generalice con la mayor precisión posible el aprendizaje adquirido (Aggarwal, 2018; Kriegeskorte & Golan, 2019).

2.1.1. Perceptrón

Rosenblatt propuso el modelo más simple y antiguo de red neuronal artificial, llamado Perceptrón (Rosenblatt, 1958). Consiste en una sola neurona, que a partir de unas entradas realiza una discriminación lineal para generar una salida (figura 2). Por tanto, se tiene una capa de entrada y otra de salida; aunque en realidad sólo se cuentan las capas en las que se realizan cálculos, es por esto por lo que se dice que el Perceptrón es una arquitectura de una sola capa.

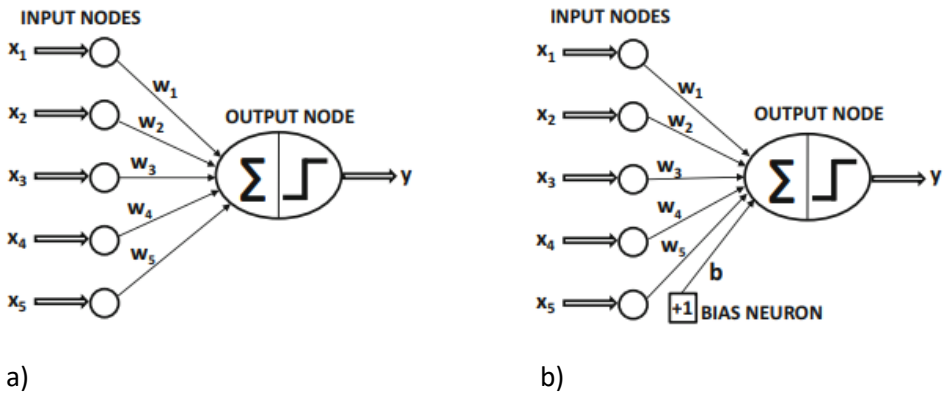

Figura 2. Arquitectura básica de un Perceptrón. a) Perceptrón sin bias; b) Perceptrón con bias. Obtenido de (Aggarwal, 2018).

En cada nodo o neurona se realizan dos operaciones. La primera operación se trata del sumatorio de las entradas X = [x_1,..., x_d] ponderadas por el peso W = [w_1,..., w_d] de cada una de ellas, donde d es el número de entradas, e in sería el valor total después de ponderar los pesos con sus entradas y sumarle el término bias (si lo hay).

$$in = b + \sum_d x_d \cdot w_d \tag{2.1}$$

El término b simboliza el sesgo (bias en inglés), y puede incluirse en la arquitectura del Perceptrón o no. Este término representa la parte invariante de los datos de entrada, es decir, si X es la entrada y queremos predecir la salida que es la Y; se crea por ejemplo un modelo:

$$Y = m \cdot X \tag{2.2}$$

El cual al entrenarse encuentra por sí mismo el valor idóneo de la constante m. En este caso, el modelo es una línea que tiene la restricción de pasar por el origen de coordenadas y muchas veces para los datos introducidos es imposible que el algoritmo ajuste el modelo para que pase por el origen. En cambio, si le añadimos el sesgo al modelo, se queda de la forma:

$$Y = m \cdot X + b \tag{2.3}$$

De esta forma, tiene plena libertad para entrenarse y encontrar un modelo que se ajuste lo mejor posible a los datos dados, en este caso sin pasar por el origen de coordenadas si así se requiere.

La segunda operación se trata de aplicar una función de activación a las entradas ponderadas para obtener una salida. Esta función de activación transforma los datos, y puede ser lineal como es el caso del Perceptrón o no lineal, como es en la mayoría de los casos ya que se obtienen mejores resultados, especialmente en redes profundas.

2.1.2. Perceptrón Multicapa

El Perceptrón Multicapa (figura 3) es una red neuronal artificial que contiene más de una capa de cálculo, entre la capa de entrada y salida contiene capas intermedias adicionales llamadas capas ocultas; se llaman ocultas porque los cálculos realizados no son visibles para el usuario. La arquitectura específica de las redes neuronales multicapa se denomina *feed-forward* porque las capas sucesivas se alimentan unas a otras en la dirección de avance desde la entrada hasta la salida, lo que permite extraer las características relevantes para llevar a cabo la clasificación

correctamente. Esta arquitectura supone que todos los nodos de una capa están conectados a los de la capa siguiente, por tanto, se puede hablar de capas *fully connected*.

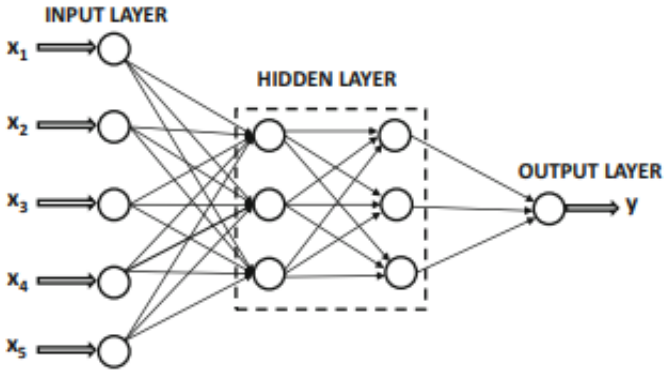

Figura 3. Arquitectura básica de un Perceptrón Multicapa. Obtenido de (Aggarwal, 2018).

2.1.3. Función de entrada

La neurona tiene una entrada global que abarca todas las entradas simples ponderadas que la constituyen. Se representa de la siguiente forma:

$$in = (x_1 \cdot w_1) * (x_2 \cdot w_2) * \dots (x_n \cdot w_n) \quad (2.4)$$

donde x_i son las entradas simples, w_i los pesos por los que se ponderan, el subíndice n el número de entradas a la neurona, siendo "*" el operador elegido (sumatorio, producto, máximo…). La función de entrada viene determinada por el operador, es decir, una vez multiplicada cada entrada por su respectivo peso, el operador utilizado da nombre a la función de entrada. Las más comúnmente utilizadas y conocidas son (Jorge Matich, 2001):

1. Sumatorio. Se suman todas las entradas ya ponderadas.
2. Producto. Se multiplican todas las entradas ya ponderadas.
3. Máximo. Una vez ponderadas todas las entradas, se elige el valor más alto.

Por defecto se utiliza casi siempre la función de entrada sumatorio.

2.1.4. Función de activación

Si no se utiliza ninguna función de activación en la red neuronal, la función de entrada pasa a ser la salida de la red neuronal (in=y). En este caso, la señal de salida será una función lineal simple, es decir, un polinomio de primer orden, y a pesar de ser una ecuación simple y fácil de resolver, su complejidad es limitada y carece de la capacidad de aprender y reconocer patrones complejos a partir de los datos. Por tanto, actúa como un modelo de regresión lineal con un rendimiento y potencia limitados en la mayoría de los casos. Lo que se espera de una red neuronal es que realice tareas más complicadas como el procesado de datos complejos como es el caso de imágenes, audios, vídeos, texto, etc.

Para tratar este tipo de datos complejos se necesitan funciones no lineales, que tienen grado mayor que uno y una curvatura cuando se trazan. La red neuronal debe ser capaz de procesar, aprender y representar cualquier dato o función que se le proporcione, de hecho, cualquier proceso imaginable puede representarse como un cálculo funcional en redes neuronales. Por lo tanto, la función de activación se necesita principalmente para dotar a la red neuronal de esa capacidad de extraer información compleja y permitir la representación de patrones funcionales aleatorios no lineales entre la entrada y la salida. De esta forma, con la función de activación se transforman los datos de entrada introduciendo la no linealidad en ellos (Sharma et al., 2020).

$$y = f(in) \tag{2.5}$$

Donde y es la salida de la red neurona, y la función f es la función de activación que se aplica sobre la entrada global (in) ya ponderada con sus respectivos pesos.

Algunas de las más habituales son:

Función escalón

Es la función de activación más simple que existe y solía utilizarse antiguamente en el Perceptrón para generar salidas binarias como se observa en la figura 4. Es lineal y su principal limitación es que no puede utilizarse para la clasificación multiclase, en la cual existen más de dos clases en la variable objetivo (y). Además, el gradiente de esta función es cero, es decir, su derivada respecto de x es cero y, por tanto, esto supone un problema para el algoritmo de retropropagación durante el entrenamiento. Matemáticamente puede definirse como:

$$\begin{aligned} f(x) &= 1, x \geq 0 \\ f(x) &= 0, x < 0 \end{aligned} \tag{2.6}$$

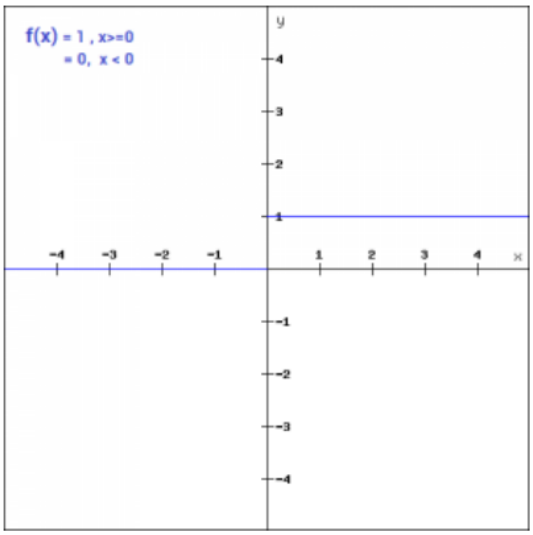

Figura 4. Representación de la función escalón. Obtenido de (Sharma et al., 2020).

Función sigmoide

Es la función de activación más utilizada ya que es una función no lineal (figura 5). Esta función transforma los valores de entrada en salidas en el rango entre 0 y 1. Asimismo, es continuamente

diferenciable y tiene forma de S suavizada, siendo derivable en su totalidad. Además, no es una función simétrica respecto de cero, lo que significa que todas las salidas de la neurona tendrán los mismos signos (positivo). Este problema se puede solucionar escalando la propia función. Su ecuación es la siguiente:

$$f(x) = \frac{1}{1+e^{-x}} \tag{2.7}$$

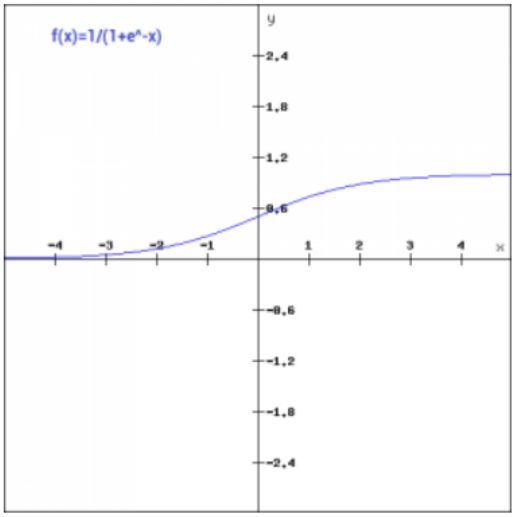

Figura 5. Representación de la función sigmoide. Obtenido de (Sharma et al., 2020).

Función tangente hiperbólica

Es una función no lineal, similar a la función sigmoide, pero es simétrica alrededor del origen, lo que resulta en diferentes signos para las salidas de la neurona (positivos y negativos). Concretamente, genera salidas en el rango entre -1 y 1 como se puede observar en la figura 6. Se trata de una función continua, diferenciable y centrada en el cero como la sigmoide, pero es preferible a esta, ya que tiene un gradiente más pronunciado que facilita el entrenamiento y no está restringido a variar en una determinada dirección. A continuación, se muestra tanto su ecuación como su gráfica:

$$tangh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} = 2 \cdot sigmoide\ (2x) - 1 = \frac{2}{1+e^{-2x}} - 1 \tag{2.8}$$

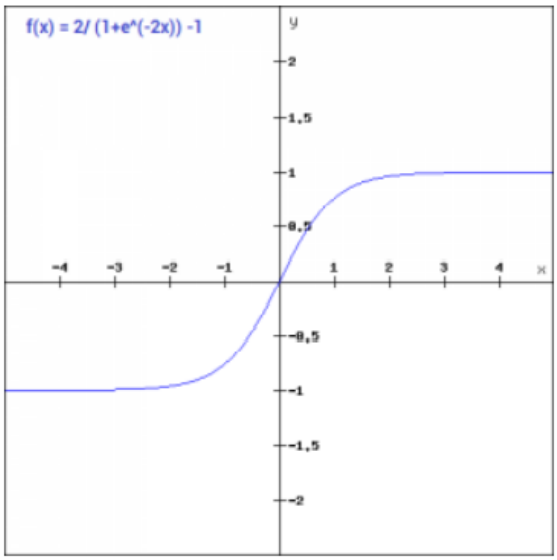

Figura 6. Representación de la función tangente hiperbólica. Obtenido de (Sharma et al., 2020).

Función ReLU

La Rectified Linear Unit (ReLU) significa unidad de línea rectificada y es una función no lineal muy utilizada en redes neuronales convolucionales (figura 7). Su principal ventaja frente al resto es su mayor eficiencia debido a la activación selectiva de sus neuronas, es decir, no se activan todas a la vez, sino que se activan únicamente las que tengan una salida mayor que cero. Concretamente, convierte los valores negativos a 0 y mantiene igual los positivos. En algunos casos, el gradiente vale cero, por lo que los pesos y sesgos no se actualizarán en el algoritmo de retropropagación durante el entrenamiento. Se define matemáticamente como:

$$f(x) = x, x \geq 0$$
$$f(x) = 0, x < 0 \qquad (2.9)$$

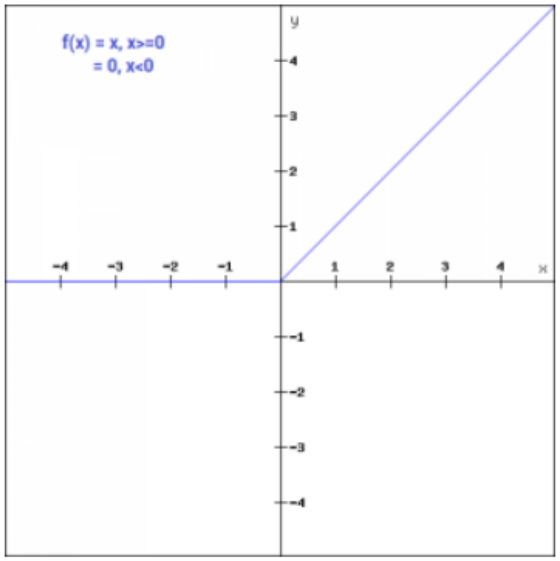

Figura 7. Representación de la función ReLU. Obtenido de (Sharma et al., 2020).

Función softmax

La función softmax combina múltiples funciones sigmoides. A diferencia de la función sigmoide que devuelve salidas en el rango entre 0 y 1 que pueden ser tratadas como probabilidades de datos particulares de una clase para clasificación binaria; la función Softmax permite la clasificación multiclase, obteniendo cómo de probable es que una entrada pertenezca a cada una de las clases. En definitiva, se trata de una función exponencial normalizada que hace posible pasar de un vector de valores reales arbitrarios a otro de valores reales entre 0 y 1. Normalmente se utiliza como capa final de los clasificadores y esta capa de salida de la red tendrá el mismo número de neuronas que número de clases del modelo. Su ecuación es la siguiente:

$$\text{softmax(x)} = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{-x_k}} \text{para j} = 1, \dots, \text{K} \tag{2.10}$$

2.1.5. Función de pérdidas

También llamada función de coste, se utiliza para mejorar la predictibilidad de la red. Esta muestra qué tan cerca está la salida del valor objetivo deseado. Además, debe ser diferenciable para que mediante el gradiente de la función calcule cómo deben actualizarse los pesos de la red. En aprendizaje profundo, esta actualización la realiza el algoritmo de retro propagación automáticamente a través de la programación dinámica para conocer cómo tienen que cambiar los pesos con el fin de minimizar la función de pérdidas. Al elegir la función de pérdidas es esencial considerar que debe ser sensible a la aplicación de la red.

Si se trata un problema de regresión donde la salida es numérica lo ideal es utilizar como función de pérdidas el *error cuadrático medio*.

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.11}$$

Donde las y_i son las observaciones reales, las $\hat{y}_i$ las predicciones y la n el número de observaciones.

En cambio, si se trata de predicciones correspondientes a probabilidades de pertenencia a ciertas clases, principalmente se utilizan estas dos funciones:

- *Regresión logística*. Se hace uso de esta cuando se dispone sólo de dos clases, es decir, para clasificación binaria. Donde el valor real está en el rango entre -1 y 1 representado por la y, y la predicción es la $\hat{y}$, así, a mayor correlación, menor será el valor de la exponencial y, por tanto, menor será el valor de la función de pérdidas (Aggarwal, 2018).

$$\text{L} = \log(1 + \exp(-y \cdot \hat{y})) \tag{2.12}$$

- *Entropía cruzada.* Se utiliza para clasificación multiclases. Se trabaja con regresión logística multinomial, de forma que a medida que aumenta la probabilidad de pertenecer a una determinada clase *n*, menores son las pérdidas y mejor es el modelo. Siendo $\hat{y}_1, \dots, \hat{y}_n$ las probabilidades de pertenecer a una determinada clase, se obtienen las pérdidas asociadas a $\hat{y}_t$, que es la probabilidad asociada a la clase *t* que sirve como *groundtruth* de la muestra (Aggarwal, 2018).

$$L = -\sum_{i=1}^{n} y_t \log(\hat{y}_i) \quad (2.13)$$

Para el presente trabajo se utilizará una función de pérdidas diferente debido a que el problema a resolver es la segmentación semántica. Aquí la red generalmente se centra en las zonas más representativas como es la región de fondo que posee mayor cantidad de píxeles que la región de interés, lo que favorece el desbalanceo de clases. Esto supone problemas para comparar el error frente al total de los datos. Por ello se ha utilizado el coeficiente Dice (Dice, 1945; Skrifter et al., 1948). El coeficiente Dice permite comparar la similitud entre dos imágenes, a partir de la similitud entre sus regiones por separado.

Se define el Dice como dos veces la intersección entre el conjunto de datos del *groundtruth* y el de la predicción, dividido por la suma del *groundtruth* y de la predicción. Se añade el término smooth de valor unitario sumado tanto al numerador como al denominador para evitar la división entre cero cuando ninguno de los dos píxeles contiene píxeles de la región de fondo.

$$DICE = \frac{2 \cdot |A \cap B| + sm}{|A| + |B| + sm} \quad (2.14)$$

'A' es el conjunto de datos del *groundtruth*, es decir, las imágenes segmentadas por el investigador y revisadas por el especialista. 'B' es el conjunto de datos de la predicción, es decir, los valores de salida calculados por la red neuronal. 'Sm' es el factor suavizador de valor unitario.

Por defecto Dice se calcula con salidas binarias, pero en nuestro caso, la red contiene salidas que son probabilidades (valores en el rango entre 0 y 1) a raíz de utilizar la función sigmoide como capa de salida.

$$DICE = \frac{2 \cdot \sum_{i}^{N} p_i g_i + sm}{\sum_{i}^{N} p_i + \sum_{i}^{N} g_i + sm} \quad (2.15)$$

Donde p_i es la probabilidad para cada píxel del *groundtruth* y g_i para cada píxel de la predicción, siendo N el número de píxeles de la imagen.

Por tanto, si el coeficiente Dice tiene un valor de 1 es porque la predicción y el *groundtruth* son iguales, mientras que si tiene un valor de 0 es porque son completamente diferentes. En el caso de estar entre ese rango, indica el grado de similitud que poseen.

2.1.6. Optimizador

La función del optimizador es la de, a partir del valor de pérdidas generado por la función de pérdidas, indicar como deben modificarse los pesos de la red para converger hacia el mínimo global y así minimizar dicha función. Se plantea pues la actualización de los pesos como un problema de optimización.

Se decide utilizar el optimizador Adam (*Complete Guide to Adam Optimization | by Layan Alabdullatef | Towards Data Science*, 2020), que es un algoritmo para la optimización del gradiente de primer orden de funciones estocásticas objetivo, basado en estimaciones adaptativas de momentos de orden inferior. Es sencillo de implementar, computacionalmente eficiente, con pocos

requisitos de memoria, invariable al reescalado diagonal de los gradientes e idóneo para problemas con gran cantidad de datos y/o parámetros.

Adam hace uso de dos *momentum* para la actualización de los pesos. El primer *momentum*, F, se genera a raíz del suavizado exponencial del gradiente de primer orden.

$$F_i = \beta_1 F_i + (1 - \beta_1)\left(\frac{\delta L}{\delta w_i}\right) \quad \forall i \tag{2.16}$$

El segundo, A, se calcula a partir del promedio exponencial de los gradientes al cuadrado. Está relacionado con los valores de los gradientes anteriores y proporciona más peso a los calculados en iteraciones más próximas a la actual.

$$A_i = \beta_2 A_i + (1 - \beta_2)\left(\frac{\delta L}{\delta w_i}\right)^2 \quad \forall i \tag{2.17}$$

A partir de ambos *momentum* se actualizan los pesos de forma que:

$$w_i = w_i - \frac{\alpha_t}{\sqrt{A_i + \epsilon}} F_i \quad \forall i \tag{2.18}$$

$$\alpha_t = \alpha \left(\frac{\sqrt{1-\beta_2^t}}{1-\beta_1^t}\right) \tag{2.19}$$

Donde α_t es la tasa de aprendizaje que depende de la iteración t. Esta ecuación (2.19) representa el ajuste sobre la tasa de aprendizaje para corregir el valor inicial de los dos *momentum*, importante en las primeras iteraciones.

De esta forma, se tiene F_i y A_i como los *momentum*, β_1, β_2 y ϵ como hiperparámetros y w_i como los pesos a actualizar.

2.1.7. Algoritmo de propagación directa y retro propagación

El algoritmo de propagación directa y retro propagación, o en inglés forward-backward propagation, fue propuesto por Rumelhart, Hinton y Williams en 1986 (Rumelhart et al., 1986) y es utilizado en las redes neuronales multicapa para llevar a cabo el proceso de optimización de los pesos, ya que se tienen en cuenta los pesos de todas las capas anteriores y esto deriva en una complicada función. En cambio, en una red neuronal monocapa no tendría sentido su uso, puesto que no puede propagarse a través de las capas ya que solo hace uso de una única capa.

Este algoritmo calcula el gradiente de la función de pérdidas respecto a cada uno de los pesos w_i de la red. A pesar del gran número de parámetros, el uso de programación dinámica para el cálculo es clave. El algoritmo se divide en dos fases:

1. *Forward propagation* o propagación hacia delante. Es la fase durante la cual se hacen las predicciones. Se introduce un conjunto de datos de entrada a la red, sobre los cuales se

realizan una serie de cálculos a medida que van hacia delante a través de la red. En cada nodo de la red neuronal se lleva a cabo la preactivación y la activación. La preactivación consiste en la suma ponderada de las entradas disponibles con sus respectivos pesos (ver ecuación 2.1). La activación determina si se transmite o no la información a la siguiente capa de neuronas en base a la función de activación utilizada. Posteriormente, la red genera una salida y se calcula la función de pérdidas que muestra el error de la salida respecto de la referencia.

2. *Backward propagation* o propagación hacia detrás. En esta fase se trata de adaptar los pesos de la red basándose en el error que se obtiene previamente durante la fase forward para reducir al mínimo la función de coste y así conseguir minimizar el error de la predicción. Esta tarea la realiza un optimizador, en este caso se utiliza el *Stochastic Gradient Descend (SGD).* Su función es actualizar los pesos para cada iteración mediante la siguiente fórmula:

$$\theta^{t+1} = \theta^{t} - \alpha \frac{\delta L(X,\theta^{t})}{\delta \theta} \tag{2.20}$$

Siendo t la iteración, θ los diferentes parámetros de la red que incluye α, W y b, que respectivamente son la tasa de aprendizaje, los pesos y los sesgos. Por tanto, se realiza la derivada parcial de la función de pérdidas respecto de los parámetros, esto es, calcular el gradiente de la función deseada respecto a los pesos de la red mediante la regla de la cadena en dirección hacia detrás. De esta forma, se consigue saber cuánto debe cambiar el peso en cada neurona para conseguir la reducción del error en la última capa a partir de los gradientes locales. En definitiva, la idea es ir en dirección contraria a la dirección del gradiente, pues este va dirigido hacia donde el crecimiento es mayor, para así encontrar un mínimo global que permita obtener las predicciones adecuadas a la salida.

Capítulo 3

Estado del arte

3.1 Redes neuronales convolucionales

Las Convolutional Neural Network (CNN) en inglés o redes neuronales convolucionales propuestas por Lecun (Lecun et al., 1998) se han convertido en uno de los métodos más exitosos en el campo del reconocimiento de patrones, utilizadas para tratar estructuras con una gran dependencia espacial como son las imágenes. Estas redes surgen a raíz del estudio de Hubel y Wiesel en el córtex visual de los gatos (Hubel & Wiesel, 1959), en el cual se descubre que ciertas áreas del cerebro se activan ante determinados estímulos visuales. Asimismo, se descubre que las neuronas son sensibles a la forma y orientación de los elementos del campo visual. Con esto, se propone la extracción de características jerárquica, es decir, en las primeras capas extraer las características más generales como bordes o formas simples y en las capas más profundas extraer estructuras más complejas y con más detalle.

Se diferencian de otras formas de redes neuronales artificiales en que, en lugar de centrarse en la totalidad del dominio del problema, se aprovechan los conocimientos sobre el tipo específico de entrada. Esto, a su vez, permite establecer una arquitectura de red mucho más sencilla.

En estas estructuras, los filtros entrenados localmente son utilizados para extraer las características visuales de la imagen de entrada. Se utiliza la convolución, una operación matemática que multiplica punto a punto dos matrices, la de entrada con el kernel (matriz de pesos) (Aggarwal, 2018) . Esta operación proporciona un valor proporcional a la información compartida por las dos imágenes a la vez que reduce el tamaño de la imagen original.

Para cada capa habrá una estructura 3D que incluye alto, ancho y profundidad. Para la imagen de entrada, un píxel es cada punto espacial determinado por la altura y anchura, y su color se obtiene

de la profundidad. En cuanto a la profundidad de las capas ocultas, se refiere al número de canales en cada capa, o lo que es lo mismo, el número de mapas de activación de la capa el cual posee información de varios tipos de formas extraídas de regiones locales de la imagen de entrada (Aggarwal, 2018).

Los mapas de características (*feature map* en inglés) cuyo tamaño se ha reducido mediante la operación de agrupación (*pooling*, en inglés) son las imágenes de entrada de la siguiente convolución. El valor de una capa viene dado por la relación espacial que se hereda entre capas, para así saber que parte de la capa anterior corresponde con la actual. Este proceso continúa hasta que se extraen las características profundas. El número de capas ocultas vendrá dado por la complejidad y el tamaño del set de datos de entrada. Después de estos pasos, un clasificador suele tomar una decisión sobre estas características (Ravi et al., 2017).

3.1.1. Arquitectura

Una CNN consiste en una capa de entrada, una capa de salida y múltiples capas ocultas. Las capas ocultas suelen ser de los tipos: convolucional, de submuestreo *(pooling*), de activación (función no lineal) y de normalización (*batch normalization* en inglés). A veces, también se incluyen la capa densa (*fully connected* en inglés) y la capa *dropout*, pero para construir nuestra red convolucional no haremos uso de ellas.

Generalmente las capas ocultas se organizan de una forma similar, aunque puede variar el orden de las capas en función de los datos de entrada y del objetivo, es por eso por lo que existen multitud de CNN tipo, entre las cuales uno escoge una y la adapta a sus necesidades. Más adelante, en el punto 4.2.1.3 se detallará la arquitectura utilizada.

3.1.1.1. Encoder-decoder

Es una arquitectura utilizada para obtener las mismas dimensiones en la salida que en la entrada. Se introduce un vector de datos de entrada, el cual se va comprimiendo a la vez que se concentran las características principales. Posteriormente, se va reconstruyendo un nuevo vector que a la salida será de nuevo del mismo tamaño que el que se tenía a la entrada.

Se compone de dos fases: *encoder*, durante la cual se comprimen los datos mediante el uso de capas convolucionales y capas *pooling*; *decoder*, durante la cual se reconstruye la dimensión original mediante la inversa de las operaciones del *encoder* con las capas de *UpSampling* y de convolución traspuesta. Para la compresión de los datos se reduce el número de neuronas en las capas intermedias.

Su uso es clave en redes profundas, ya que el aumento de la profundidad de capas ocultas permite reducir la dimensionalidad y extraer características importantes para la red. Esto, junto al uso de funciones de activación no lineales potencian la representación de los datos.

Para las CNN, el uso del *encoder-decoder* se centra en el uso de relaciones espaciales entre los puntos para posibilitar la extracción de características que puedan visualizarse en la imagen de salida. Como en las CNN, estas relaciones espaciales se generan gracias a las convoluciones y se consigue reducir la jerarquía de los datos.

De forma general, las capas se organizan simétricamente, de modo que una capa en el *decoder* deshace el efecto correspondiente a la capa en el *encoder*.

3.1.2. Tipos de capas

Existen multitud de tipos de capas, pero en el presente trabajo se explican las que resultan interesantes para la segmentación semántica, y en concreto, para el tipo de arquitectura que más se adapta a las necesidades del trabajo: U-Net.

3.1.2.1. Capa convolucional

La capa de convolución es una estructura con un número de filtros de tamaño fijo que permite aplicar funciones complejas a la imagen de entrada (Ravi et al., 2017). Este proceso se lleva a cabo deslizando los filtros entrenados localmente sobre la imagen. Cada filtro tiene los mismos valores de peso y sesgo durante el proceso en toda la imagen. Esto se denomina mecanismo de reparto de pesos y permite representar la misma característica en toda la imagen (Chandrakumar & Kathirvel, 2016; Ravi et al., 2017; Sankar et al., 2016). El campo receptivo local de una neurona representa el área a la que la neurona está conectada en la capa anterior. El tamaño del campo receptivo está determinado por el tamaño de los filtros. Siendo *m x n* el tamaño de la imagen de entrada, *c x c* el tamaño de kernel, *i* la imagen, *w* el peso y *b* el sesgo. La salida $o_{o,o}$ se puede calcular como en la ecuación (2.11), donde *f* es la función de activación que puede ser tanto una función ReLU como una sigmoide para este caso (Nielsen, 2015; Sankar et al., 2016).

$$o_{o,o} = f\ (b + \sum_{t=0}^{c} \sum_{r=0}^{c} w_{t,r} i_{0+t,0+r}) \tag{2.21}$$

Por tanto, la operación de convolución consiste en un producto escalar que multiplica punto a punto cada posición de la imagen por cada posición del kernel. Por posición se hace referencia a la altura, anchura y profundidad del filtro y la imagen. El kernel es una matriz de tres dimensiones que contiene la información de los pesos que serán optimizados por el algoritmo de retropropagación para extraer las características importantes, las cuales suelen ser bordes, enfoque, desenfoque, iluminación, etc.

Se definen los siguientes hiperparámetros en la convolución:

- El número de filtros o kernels, se especifica antes de entrenar la red y dependerá de los datos de entrada y su complejidad. Se suelen utilizar múltiplos de 2 pero se determina empíricamente.
- El tipo de filtro, aunque este hiperparámetro no se especifica antes del entrenamiento ya que la red lo genera automáticamente mediante el aprendizaje. Es un modelo de caja negra, pues no se sabe exactamente las operaciones que realiza para llegar al resultado final, pero se ajustan los pesos durante el entrenamiento mediante el algoritmo de retropropagación y al final se obtienen determinados filtros. Cada uno de estos filtros obtiene ciertas características de la imagen, algunos de ellos son el filtro derivador para detectar bordes o formas geométricas, el filtro promediador, el filtro gaussiano que reduce el ruido de la imagen, etc.
- El tamaño de kernel, que suele ser un filtro de tamaño 3x3, 5x5 o 7x7. Se debe tener en cuenta que su tamaño debe ser menor al de la imagen y debe tener la misma profundidad que la capa a la que se aplica. Si la imagen es en color, un kernel de por ejemplo 3x3, sería de 3x3x3, es decir, un filtro con 3 kernels de 3x3 que luego se suman y se le añade el sesgo para obtener una única salida.

- El *stride* o paso que es la cantidad de desplazamiento del filtro después de cada uso, es decir, el número de casillas que se desplaza de izquierda a derecha y de arriba a abajo. El valor del *stride* suele ser 1 por defecto, pero se puede aumentar para obtener imágenes de menor tamaño, lo cual resulta útil en redes convolucionales para reducir la cantidad de datos a procesar entre una capa y otra de la red.
- A veces, también se aplica el *padding*, que consiste en agregar píxeles nuevos de valor cero alrededor de los bordes de la imagen original para que la imagen resultante de la convolución sea del mismo tamaño que la original, de esta forma, es posible crear redes más profundas para extraer características más específicas de las imágenes. También se utiliza el *padding* cuando hay información importante en las esquinas de la imagen, así, la información estará más centrada y el filtro la detectará mejor.

Una vez definidos estos parámetros ya se efectúa la convolución en cada posición de la imagen para obtener en la salida los mapas de características o de activación resultantes de cada capa. Finalmente, los mapas de características se concatenan formando un vector de características que recibe el nombre de código CNN.

Posteriormente a la capa convolucional se suele aplicar la función de activación ReLU o sigmoide que se aplica a los mapas de características resultantes de la convolución. Su función es introducir la no linealidad como se ha explicado anteriormente en el apartado 2.1.4.

3.1.2.2. Capa pooling

Después de la función de activación se aplica un método de submuestreo mediante la capa *pooling* para conservar la información importante y descartar los detalles irrelevantes. Esta capa se utiliza por dos razones. Primero, se utiliza para reducir el tamaño de la matriz del mapa de características y compactar la representación, con lo que se reduce el coste computacional ya que se reduce el número de parámetros que tiene que aprender la red, por lo que, consecuentemente, se reduce el sobreajuste del modelo. Al reducirse el coste computacional, se puede invertir ese coste ahorrado en hacer la red más profunda aplicando un mayor número de filtros, lo que se traduce en capturar un mayor número de características de la imagen y esto permite una mayor especialización de los filtros. Segundo, se utiliza porque proporciona una invarianza a pequeñas translaciones (invarianza local) o una invariabilidad a cambios en las condiciones de iluminación, así como robustez ante el desorden (*Redes Neuronales Convoluciones — Introducción al Aprendizaje Automático*, 2021; Wang et al., 2016).

Existen tres tipos de operaciones *pooling*, y en función de los datos de entrada se utiliza una u otra:

1. *Max pooling*: Devuelve el valor de píxel máximo de la región seleccionada. Selecciona los píxeles más brillantes de la imagen y es útil cuando el fondo de la imagen es oscuro y nos interesan los píxeles más claros. Es la operación más utilizada de las tres.
2. *Min pooling*: Devuelve el valor de píxel mínimo de la región seleccionada. Contrariamente al *max pooling*, selecciona los píxeles menos brillantes de la imagen y es útil cuando el fondo de la imagen es claro y nos interesan los píxeles más oscuros.
3. *Average pooling*: Devuelve el valor de píxel promedio de todos los píxeles de la región seleccionada. Suaviza la imagen y, por tanto, es posible que no se identifiquen correctamente las características de nitidez.

La región seleccionada para aplicar la operación *pooling* es un grupo de píxeles de tamaño equivalente al tamaño del filtro (*Maxpooling vs Minpooling vs Average Pooling | by Madhushree Basavarajaiah | Medium*, 2019).

3.1.2.3. Capa batch normalization

Para entrenar una red neuronal se realiza un preprocesamiento a los datos de entrada. Por ejemplo, se suelen normalizar todos los datos de entrada para que se asemejen a una distribución normal con media cero y varianza uno. Esto se hace para que no difieran mucho los datos entre ellos y evitar la saturación temprana de las funciones de activación no lineales entre otras razones. Esto ayuda a la red a trabajar mejor, pero solo beneficia a la capa de entrada ya que en las capas intermedias la distribución está en constante cambio durante el entrenamiento. Esto ralentiza el proceso de entrenamiento porque cada capa debe aprender a adaptarse a una nueva distribución en cada paso de entrenamiento, es lo que se llama 'internal covariate shift'.

Para solucionar este problema surge el método *batch normalization* que consiste en normalizar los pesos de cada capa con una media aproximada de cero y una varianza cercana a uno. De esta forma se estabiliza el proceso de aprendizaje y se reduce considerablemente el número de épocas (Batch Normalization: Theory and How to Use It with Tensorflow | by Federico Peccia | Towards Data Science, n.d.; Dropout y Batch Normalization, n.d.; Ioffe & Szegedy, 2015).

3.1.2.4. Capa de activación

También llamada capa de no linealidad, técnicamente no es una capa como tal (debido a que no se aprenden parámetros ni pesos dentro de esta capa) sino que activa una función de activación no lineal para dotar al modelo de la no linealidad necesaria. A veces se omite en los diagramas de la arquitectura de la red, ya que se asume que después de una convolución se aplica una función de activación. Suele utilizarse la función ReLU porque permiten un entrenamiento de la red más rápido.

3.1.2.5. Capa de UpSampling

Esta capa se utiliza para conseguir mayor representación espacial de la información y así poder recuperar el tamaño original de la imagen de entrada. Consiste en interpolar nuevos píxeles a partir del tipo de interpolación seleccionado y el tamaño que se desea obtener (ambos parámetros deben especificarse).

3.1.2.6. Capa de convolución traspuesta

También denominada deconvolución, tiene como objetivo recuperar la resolución de la imagen a partir de un filtro traspuesto. Esto se consigue con la aplicación de un kernel sobre cada una de las posiciones del mapa de entrada. Hay que tener en cuenta que el kernel en ciertas posiciones no solapa totalmente la imagen y hay mayor contribución en las zonas centrales del mapa de salida. A pesar de esto, la red tiene la capacidad de aprender desajustes como estos y los compensa para que no haya errores; o también se puede evitar aplicando la técnica de *padding*.

Capítulo 4

Materiales y métodos

4.1. Materiales

4.1.1. Set de datos

El set de datos utilizado para el presente trabajo proviene del proyecto previo 2021-595-1 TIAbdoSeg "DESARROLLO DE TÉCNICAS BASADAS EN TRANSFER LEARNING PARA LA SEGMENTACIÓN MÁSIVA MULTIÓRGANO", los cuales se encontraban anonimizados en formato DICOM y se han transformado al formato NIFTI con el que hemos trabajado.

El modelo original del que partimos para la aplicación del transfer learning fue desarrollado con 150 casos, con el objetivo de segmentar el hígado sobre secuencias MECSE (Jimenez-Pastor et al., 2021). En nuestro trabajo contamos una N aproximada del 10%, concretamente, 16 casos.

Para cada caso, RM abdominal, contamos con las dos potenciaciones de interés, THRIVE y MECSE. THRIVE (T1 Weighted High Resolution Isotropic Volume Examination) es una técnica optimizada de obtención de imágenes tridimensionales ponderadas en T1 que combina la codificación de la sensibilidad, la cobertura de grandes volúmenes y la supresión uniforme de la grasa. Además, proporciona imágenes isotrópicas de alta resolución en tiempo de retención de la respiración cortos (*MRI - T1W High Resolution Isotropic Volume Examination - MR-TIP: Database*, 2018).

MECSE (Multi-Echo Chemical Shift–Encoded gradient echo) es una técnica con eco de gradiente codificada por desplazamiento químico múltiple para la cuantificación no invasiva de la grasa que permite la estimación simultánea de la fracción de protones móviles relacionados con la grasa

minimizando la influencia de T1, relacionada principalmente con los diferentes tiempos de relajación de la grasa y el agua (Jimenez-Pastor et al., 2021; Martí-Aguado et al., 2020).

4.1.2. Software y hardware

Para desarrollar la parte computacional de las redes neuronales del trabajo se ha hecho uso de Python 3.8 como lenguaje de programación y se ha utilizado el *framework* Keras ya que es una librería de alto nivel sustentada por TensorFlow como *backend* y ofrece un código optimizado y ampliamente validado que permite simplificar la programación de algoritmos basados en aprendizaje profundo. Como entorno de programación se ha utilizado PyCharm 2021.3.3.

En relación con el hardware, se ha utilizado un ordenador Victus by HP Laptop con las siguientes especificaciones:

Tabla 1. Especificaciones del equipo utilizado.

Nombre del procesador	11th Gen Intel® Core™ i7 11800H 8 núcleos
Velocidad del procesador	2.3 GHz
Memoria RAM	16 GB
Tipo de sistema	Sistema operativo de 64 bits
Tarjeta gráfica	NVIDIA GeForce RTX 3050 Laptop GPU

A pesar de estas potentes especificaciones que proporcionan una alta capacidad de computación de 8.6 como se puede observar en la siguiente figura:

```
2022-08-14 19:20:39.655074: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1720] Found device 0 with properties:
pciBusID: 0000:01:00.0 name: NVIDIA GeForce RTX 3050 Laptop GPU computeCapability: 8.6
coreClock: 1.5GHz coreCount: 16 deviceMemorySize: 4.00GiB deviceMemoryBandwidth: 178.84GiB/s
```

Figura 8. Mensaje de la consola de Python.

Han aparecido algunas limitaciones durante los entrenamientos, que han limitado el batch size a un máximo de 20 (se obtiene empíricamente). Si se supera, aparece este mensaje en la consola y se detiene el entrenamiento:

2022-08-15 15:32:51.767760: W tensorflow/core/framework/op_kernel.cc:1763] OP_REQUIRES failed at fused_batch_norm_op.cc:1302 : Resource exhausted: OOM when allocating tensor with shape[30,32,192,192] and type float on /job:localhost/replica:0/task:0/device:GPU:0 by allocator GPU_0_bfc

4.2. Metodología

4.2.1. Segmentación

4.2.1.1. Preparación de los datos

Los datos se han transformado de formato DICOM a NIFTI. Se ha realizado una segmentación manual de los órganos de interés: riñones (excluyendo el seno renal) y bazo. Esto se ha realizado en 16 casos, tanto sobre la secuencia THRIVE como sobre la MECSE, obteniendo unas máscaras con dicha segmentación en formato NIFTI. Las segmentaciones se realizaron con los programas ITK-

SNAP (*ITK-SNAP Home*, 2018) y MITK (*Medical Imaging Interaction Toolkit: Main Page*, 2022), los cuales tienen una interfaz sencilla y fácil de utilizar. Estas segmentaciones fueron posteriormente revisadas por una radióloga, Diana.

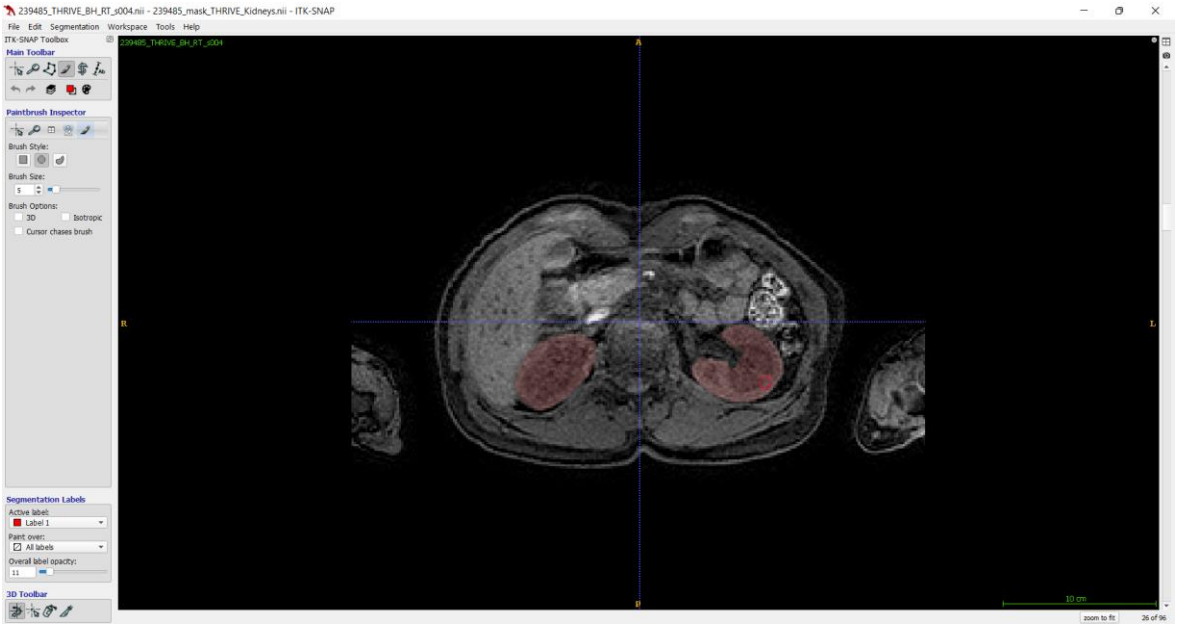

Figura 9. Entorno de segmentación de ITK-SNAP. Corte axial de segmentación de riñones.

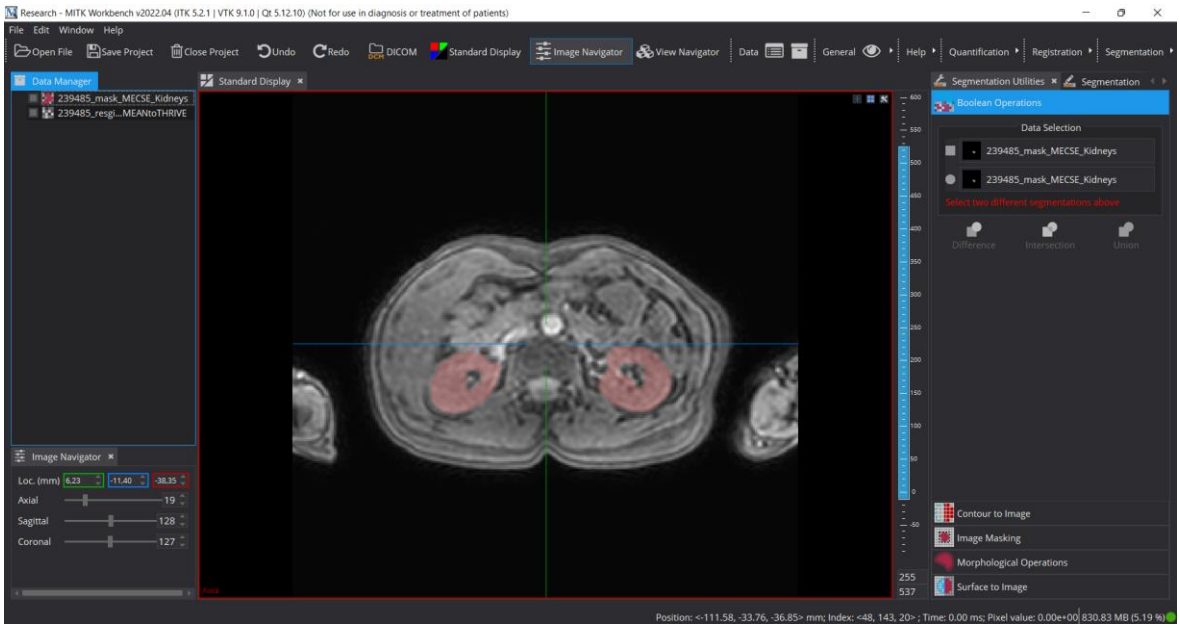

Figura 10. Entorno de segmentación de MITK. Corte axial de segmentación de riñones.

4.2.1.2. Entrenamiento

El entrenamiento es lo que permite a la red neuronal aprender las características de las imágenes. Se le pasa un set de datos a la entrada pasando desde la capa de entrada hasta la capa de salida en la etapa forward. Posteriormente, se compara la salida producida por la red con las imágenes del

groundtruth y se calcula el error de la predicción. Seguidamente, se realiza la etapa de *backpropagation* para actualizar los pesos. Este proceso se repite iterativamente hasta conseguir un resultado óptimo.

Para entrenar nuestro modelo adecuadamente y saber si está funcionado bien, el set de datos se divide en dos conjuntos: entrenamiento (*train*) y pruebas (*test*). Generalmente se toma 75-25% o 80-20% para entrenamiento y test respectivamente. En este trabajo, de un total de 16 casos, se han tomado 75% para entrenamiento, es decir, 12 casos; y 25% para test, es decir, 4 casos.

El conjunto de entrenamiento se utiliza para alimentar la red con imágenes y mediante el aprendizaje actualizar los pesos. El conjunto de test, que nunca pasa por el modelo, se utiliza cuando la red ya ha sido entrenada para evaluar su funcionamiento frente a un set de datos externo, el cual no ha sido visto previamente. En resumen, observa una entrada y genera una predicción, la cual se contrasta con el valor real.

Este proceso está regido por la arquitectura de la red y el conjunto de hiperparámetros elegido, los cuales se desarrollan posteriormente en los puntos 4.2.1.3 y 4.2.1.4

4.2.1.3. Arquitectura de la red

Se ha hecho uso de la arquitectura U-Net, la cual es ampliamente conocida y utilizada para segmentación de imágenes biomédicas (Ronneberger et al., 2015) y presenta una arquitectura de *encoder-decoder* convolucional. La parte contráctil (*encoder*) y la parte expansiva (*decoder*) son prácticamente simétricas y por eso se obtiene la arquitectura en forma de 'U', la cual le otorga dicho nombre. La parte contráctil extrae características mediante las cuales se consigue una representación abstracta de las imágenes de entrada para después, realizar una reconstrucción en la parte expansiva que consigue la clasificación individual de los píxeles de la imagen en la que se encuentran. Además, esta arquitectura cuenta con canales de concatenación (conexión de salto) entre los bloques convolucionales del *encoder* y del *decoder* que hacen posible reutilizar características concatenándolas a nuevas capas, lo que permite retener y recuperar información espacial de capas anteriores que se pierde durante la etapa en la que se reduce la dimensionalidad. También hace uso de conexiones residuales donde utiliza la suma de elementos para incorporar información de capas anteriores.

La etapa contráctil sigue una estructura parecida a una red neuronal convolucional estándar. Está formada por cuatro bloques. Cada uno de estos bloques está formado por: capa convolucional 3x3, capa *batch normalization*, capa de activación ReLU, se repiten estas 3 capas y finaliza con una capa *max pooling*.

En cada capa de convolución se utiliza un kernel de tamaño 3x3 y un *padding same* y un stride de 1. La capa de *max pooling* se aplica sobre regiones 2x2 para reducir el tamaño del mapa de entrada a la mitad. Asimismo, después de cada bloque se dobla el número de filtros usados: 32, 64, 128, 256.

En la etapa expansiva se tiene cinco bloques. El primer bloque es igual que los de la etapa contráctil y dobla los filtros de 256 a 512 pero ya no contiene la capa *max pooling* para reducir la dimensionalidad. En los cuatro bloques restantes, cada uno empieza con una capa de convolución traspuesta que reduce a la mitad el número de filtros y mediante un kernel 2x2 dobla el tamaño de la imagen deshaciendo el efecto de la capa *max pooling* de la etapa contráctil. Estos bloques tienen la misma estructura que antes, pero con la capa de convolución traspuesta añadida al principio, sin

la capa de *max pooling* al final y con una capa convolucional 1x1 al final del bloque. Además, se concatena la capa de la convolución traspuesta con el bloque de la etapa contráctil correspondiente (para conectar *encoder* con *decoder*) y con el siguiente bloque de la etapa expansiva. También se dispone de capas *Up Sampling* en la etapa expansiva que se suman con los bloques anterior y posterior. Estas conexiones entre *encoder-decoder* permiten la supervisión profunda, la cual aprovecha las salidas complementarias de las capas ocultas para calcular la pérdida final como la pérdida de salida más la suma de las pérdidas complementarias. De esta forma, actúa como una especie de regularización de características, lo que conduce a una reducción significativa del error de prueba, pero no necesariamente del error de entrenamiento; y resulta en una convergencia más rápida, especialmente en presencia de pequeños datos de entrenamiento (Lee et al., 2014). Finamente, la etapa expansiva, y la arquitectura, termina con una capa de activación sigmoide para generar valores entre 0 y 1.

La arquitectura descrita se presenta a continuación de forma esquemática:

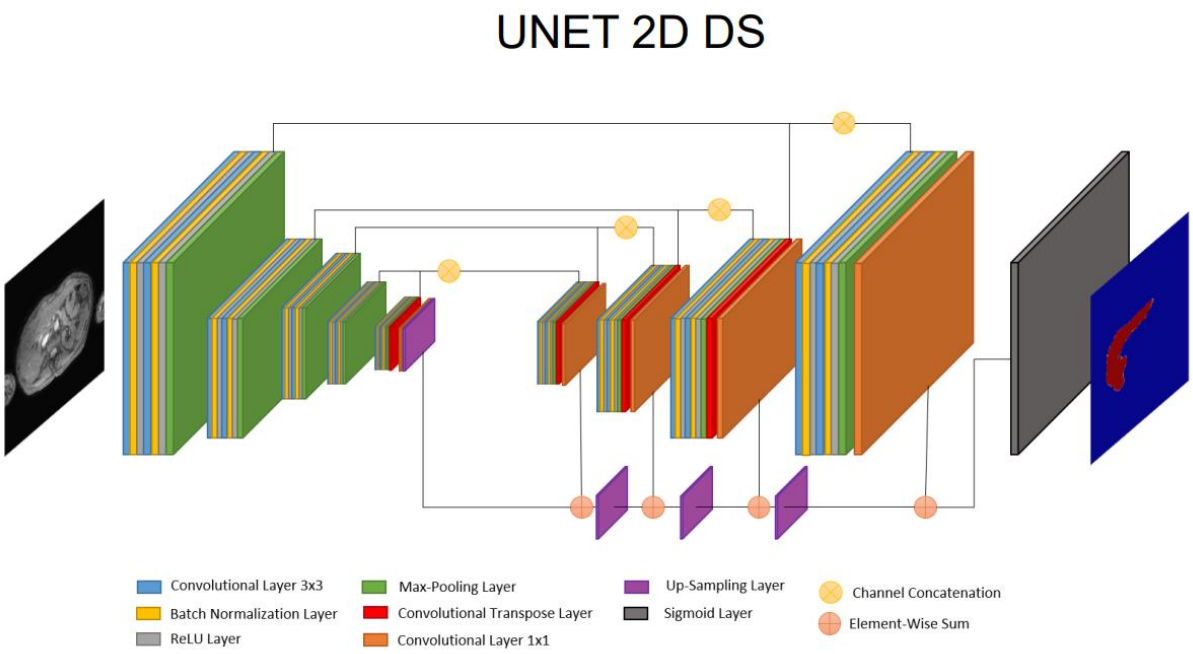

Figura 11. Representación de la arquitectura U-Net utilizada. Obtenido de (Jimenez-Pastor et al., 2021).

4.2.1.4. Hiperparámetros

Estos hiperparámetros son los valores elegidos por el programador durante la fase de diseño de la red neuronal y que determinaran como será el entrenamiento de la red. Son parámetros que se deben especificar, a diferencia de los pesos de la red que se obtienen de los datos. Los valores adecuados para estos hiperparámetros son desconocidos y deben obtenerse empíricamente, aunque se suelen usar valores que han funcionado previamente en otros estudios para guiarse y se van modificando.

- Número de épocas. Es el número de veces que el set de datos de entrenamiento al completo pasa por la red neuronal completando un ciclo *forward-backward*. Por tanto, cada una de las imágenes del set de entrenamiento realiza tantas etapas *forward-backward* como número de épocas se especifique.

Tamaño de *batch*. El *batch* es un subconjunto de imágenes dentro del set de datos. Cuando el set de datos es muy grande se divide en *batches*, y se completará una época cuando el número de iteraciones multiplicado por el tamaño de *batch* sea igual al set de datos. Por ejemplo, en un set de datos de 2000 imágenes, si se define un tamaño de *batch* de 500 imágenes, se completará una época después de 4 iteraciones (4 iteraciones x 500 imágenes/*batch* =2000 imágenes). Por tanto, los pesos se actualizarán en función del número de *bacthes* de forma directamente proporcional. Existen tres posibilidades:

- Tamaño de *batch* = Tamaños set de datos → Se recalculan pesos una vez por época
- Tamaño de *batch* = 1 imagen → Tendrá tantos *batches*/iteraciones como imágenes tiene el set de datos y, por tanto, los pesos cambiarán con respectos a las características de la imagen.
- Tamaño de *batch* = 32-256 imágenes → Se actualizan los pesos varias veces en una misma época, es lo ideal y lo que se suele utilizar.

Tasa de aprendizaje. Este hiperparámetro determina la velocidad de aprendizaje del modelo, es decir, la cantidad de tiempo empleada para que el modelo se acerque al mínimo global del plano definido por el gradiente. A mayor tasa de aprendizaje, más rápida es la convergencia hacia el mínimo global, pero podría no encontrarlo nunca debido al uso de saltos más grandes. En contraposición, a menor tasa de aprendizaje, la convergencia es más lenta y podría atascarse en un mínimo local del que no podría salir.

Ratio asociado al *momentum*. El *momentum* es un parámetro de valor comprendido entre 0 y 1 que debe determinarse por ensayo y error. Simplemente se introduce como una fracción m que pondera la actualización de los pesos y almacena información de los gradientes anteriores. Sirve de guía para mejorar la convergencia hacia el mínimo ayudando al optimizador en zonas propensas a errores donde el gradiente es muy pronunciado.

Función de pérdidas. Compara la salida respecto al *groundtruth* para saber cómo de buena es la salida de la red. Se desea minimizar esta función, que se consigue optimizando los parámetros de la red para obtener mejores predicciones. Se ha elegido el coeficiente Dice.

Optimizador. El optimizador indica cómo han de modificarse cada uno de los pesos de la red para converger hacia el mínimo global a partir del valor de pérdidas. Su función es optimizar los parámetros de la red para minimizar la función de pérdidas. Se ha utilizado el optimizador Adam explicado previamente.

4.2.2. Entrenamientos realizados

Una vez diseñada la arquitectura, se ha optado por entrenar los diferentes modelos con diferentes parámetros para posteriormente comprobar la métrica de validación del coeficiente Dice, observando su evolución a través de gráficas. De cada entrenamiento, se han guardado los pesos de las épocas que han mejorado el Dice anterior a lo largo de todo el entrenamiento, así como el modelo y los pesos de la última época. A continuación, se enumeran los modelos y los entrenamientos realizados, para posteriormente en el capítulo de resultados, extraer conclusiones a través de analizarlos y compararlos.

Modelo 1: Segmentar riñones sobre secuencias MECSE y THRIVE.

- 1.a Entrenamiento con 30 épocas

- *Batch size* de 10 y tasa de aprendizaje de 1,00E-3
    - 1.a.1 Pesos iniciales aleatorios
    - 1.a.2 Pesos iniciales de hígado (TL)

- 1.b Entrenamiento con 100 épocas
    - *Batch size* de 10 y tasa de aprendizaje de 1,00E-3
        - 1.b.1 Pesos iniciales aleatorios
        - 1.b.2 Pesos iniciales de hígado (TL)

- 1.c Entrenamiento con 300 épocas
    - *Batch size* de 10
        - 1.c.1 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3
        - 1.c.2 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4
        - 1.c.3 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-3
        - 1.c.4 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-4
    - *Batch size* de 20
        - 1.c.5 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3
        - 1.c.6 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4
        - 1.c.7 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-3
        - 1.c.8 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-4

Modelo 2: Segmentar riñones sobre secuencia MECSE.

- 2.a Entrenamiento con 100 épocas
    - *Batch size* de 10
        - 2.a.1 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3
        - 2.a.2 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4
        - 2.a.3 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-3
        - 2.a.4 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-4
- 2.b Entrenamiento con 300 épocas
    - *Batch size* de 10
        - 2.b.1 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4
        - 2.b.2 Pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3
        - 2.b.3 Pesos iniciales de hígado (TL), tasa de aprendizaje de 1,00E-3

Modelo 3: Segmentar riñones sobre secuencia THRIVE.

- 3.a Entrenamiento con 100 épocas
    - *Batch size* de 10 y tasa de aprendizaje de 1,00E-3
        - 3.a.1 Pesos iniciales aleatorios
        - 3.a.2 Pesos iniciales de hígado (TL)

Modelo 4: Segmentar bazo sobre secuencia MECSE y THRIVE.

- 4.a Entrenamiento con 100 épocas
    - *Batch size* de 10 y tasa de aprendizaje de 1,00E-3
        - 4.a.1 Pesos iniciales aleatorios
        - 4.a.2 Pesos iniciales de hígado (TL)
        - 4.a.3 Pesos iniciales de riñones (TL)

Modelo 5: Segmentar bazo sobre secuencia MECSE.

- 5.a Entrenamiento con 100 épocas
  - *Batch size* de 10 y tasa de aprendizaje de 1,00E-3
    - 5.a.1 Pesos iniciales aleatorios
    - 5.a.2 Pesos iniciales de hígado (TL)
    - 5.a.3 Pesos iniciales de riñones (TL)

Modelo 6: Segmentar bazo sobre secuencia THRIVE.

- 6.a Entrenamiento con 100 épocas
  - *Batch size* de 10 y tasa de aprendizaje de 1,00E-3
    - 6.a.1 Pesos iniciales aleatorios
    - 6.a.2 Pesos iniciales de hígado (TL)
    - 6.a.3 Pesos iniciales de riñones (TL)

Tabla 2. Tabla de referencias.

ENTRENAMIENTO							**RESULTADOS (DICE)**		
Id Modelo	Dominio	Objetivo	*Batch size*	Épocas	Tasa de aprendizaje	Pesos iniciales	Máximo	Media	Varianza
1.a.1	MECSE y THRIVE	Riñones	10	30	1,00E-3	Aleatorios	0,856	0,667	0,0506
1.a.2	MECSE y THRIVE	Riñones	10	30	1,00E-3	Hígado	0,858	0,762	0,0103
1.b.1	MECSE y THRIVE	Riñones	10	100	1,00E-3	Aleatorios	0,879	0,761	0,0176
1.b.2	MECSE y THRIVE	Riñones	10	100	1,00E-3	Hígado	0,862	0,776	0,005
1.c.1	MECSE y THRIVE	Riñones	10	300	1,00E-3	Aleatorios	0,871	0,704	0,0071
1.c.2	MECSE y THRIVE	Riñones	10	300	1,00E-4	Aleatorios	0,766	0,644	0,0091
1.c.3	MECSE y THRIVE	Riñones	10	300	1,00E-3	Hígado	0,852	0,772	0,0043
1.c.4	MECSE y THRIVE	Riñones	10	300	1,00E-4	Hígado	0,692	0,588	0,002
1.c.5	MECSE y THRIVE	Riñones	20	300	1,00E-3	Aleatorios	0,807	0,677	0,0091
1.c.6	MECSE y THRIVE	Riñones	20	300	1,00E-4	Aleatorios	0,728	0,587	0,0098
1.c.7	MECSE y THRIVE	Riñones	20	300	1,00E-3	Hígado	0,816	0,719	0,0028
1.c.8	MECSE y THRIVE	Riñones	20	300	1,00E-4	Hígado	0,552	0,519	0,00004
2.a.1	MECSE	Riñones	10	100	1,00E-3	Aleatorios	0,867	0,72	0,0307
2.a.2	MECSE	Riñones	10	100	1,00E-4	Aleatorios	0,678	0,529	0,0215
2.a.3	MECSE	Riñones	10	100	1,00E-3	Hígado	0,859	0,784	0,0048
2.a.4	MECSE	Riñones	10	100	1,00E-4	Hígado	0,593	0,511	0,0016
2.b.1	MECSE	Riñones	10	300	1,00E-4	Aleatorios	0,779	0,658	0,0199
2.b.2	MECSE	Riñones	10	300	1,00E-3	Aleatorios	0,851	0,761	0,0154

2.b.3	MECSE	Riñones	10	300	1,00E-3	Hígado	0,894	0,791	0,0045
3.a.1	THRIVE	Riñones	10	100	1,00E-3	Aleatorios	0,738	0,604	0,0262
3.a.2	THRIVE	Riñones	10	100	1,00E-3	Hígado	0,829	0,722	0,0073
4.a.1	MECSE y THRIVE	Bazo	10	100	1,00E-3	Aleatorios	0,656	0,5342	0,0073
4.a.2	MECSE y THRIVE	Bazo	10	100	1,00E-3	Hígado	0,659	0,54	0,0042
4.a.3	MECSE y THRIVE	Bazo	10	100	1,00E-3	Riñones	0,712	0,646	0,0019
5.a.1	MECSE	Bazo	10	100	1,00E-3	Aleatorios	0,687	0,566	0,0152
5.a.2	MECSE	Bazo	10	100	1,00E-3	Hígado	0,735	0,597	0,0056
5.a.3	MECSE	Bazo	10	100	1,00E-3	Riñones	0,738	0,64	0,0032
6.a.1	THRIVE	Bazo	10	100	1,00E-3	Aleatorios	0,678	0,534	0,0151
6.a.2	THRIVE	Bazo	10	100	1,00E-3	Hígado	0,67	0,504	0,01
6.a.3	THRIVE	Bazo	10	100	1,00E-3	Riñones	0,685	0,527	0,0065

Capítulo 5
Resultados

En este capítulo se va a analizar la influencia de los hiperparámetros y el transfer learning en los entrenamientos de los diferentes modelos. Se distinguen diferentes modelos en función del órgano segmentado: riñón (modelos 1.x.x, 2.x.x y 3.x.x) o bazo (4.x.x, 5.x.x y 6.x.x). Estos órganos son segmentados con diferentes secuencias: MECSE y THRIVE (1.x.x y 4.x.x), sólo MECSE (2.x.x y 5.x.x) o sólo THRIVE (3.x.x y 6.x.x).

En cada uno de los entrenamientos de estos modelos se seleccionaron unos hiperparámetros: número de épocas, batch size y tasa de aprendizaje; así como la inicialización de los pesos del modelo (aleatoria, hígado o riñón mediante la aplicación de transfer learning). El rango de valores elegido para los diferentes hiperparámetros se aproxima analizando lo que ha funcionado en otros proyectos similares en la literatura científica, en este caso, el proyecto que ha servido de referencia ha sido 'Precise whole liver automatic segmentation and quantification of PDFF and R2* on MR images' (Jimenez-Pastor et al., 2021).

Para poder analizar adecuadamente el efecto de los hiperparámetros a lo largo de todas las comparaciones realizadas, se mantendrán fijos todos los hiperparámetros a excepción del que se desee evaluar. Conforme se vayan extrayendo conclusiones de los entrenamientos que se van realizando, se irán limitando los hiperparámetros para los valores que proporcionen mejores resultados. Los resultados se valoran mediante la métrica de validación mencionada anteriormente: el coeficiente Dice, la cual indica que a mayor Dice en validación, mejor es el modelo. Se representa gráficamente el coeficiente Dice frente al número de épocas de entrenamiento para poder analizar y seleccionar los mejores hiperparámetros que generarán los mejores modelos. Además, también se obtendrá numéricamente el promedio, la varianza y el valor máximo de los coeficientes DICE, para analizar con mayor precisión los resultados.

5.1. Modelo de riñón entrenado con secuencias MECSE y THRIVE

En este primer modelo se analiza la influencia del batch size, tasa de aprendizaje, pesos de inicialización y número de épocas. Al tratarse del primer modelo analizado, se consideran los diferentes rangos de valores para todos los hiperparámetros, para poder fijar los que mejor funcionan e ir reduciendo la variabilidad de los entrenamientos posteriores.

En las próximas comparativas, para seleccionar el hiperparámetro batch size, la tasa de aprendizaje y los pesos de inicialización, se seleccionarán 300 épocas para tener suficientes datos y poder ver adecuadamente la evolución de los pesos.

5.1.1. Comparativas para seleccionar el hiperparámetro batch size.

En estas primeras comparativas se ha evaluado la influencia del batch size, ya que es el hiperparámetro limitado por la capacidad computacional del ordenador. El máximo batch size que hemos podido utilizar como se ha comentado en el punto 4.1.2 es de 20. Así pues, se han realizado entrenamientos con batch size tanto de 10 como de 20 que se muestran a continuación:

Se realizan las comparaciones por pares, donde solo cambia el hiperparámetro del batch size, es decir, se comparan dos gráficas que tienen los mismos hiperparámetros a excepción del batch size.
1ª comparativa (1.c.1 con 1.c.5)

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios y tasa de aprendizaje 1,00E-3.

Se diferencian en que el modelo 1.c.1 tiene batch size de 10 y el 1.c.5 batch size de 20.

Id (1.c.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,704; Varianza DICE = 0,0071

Id (1.c.5). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 20.*

Media DICE = 0,677; Varianza DICE = 0,0091

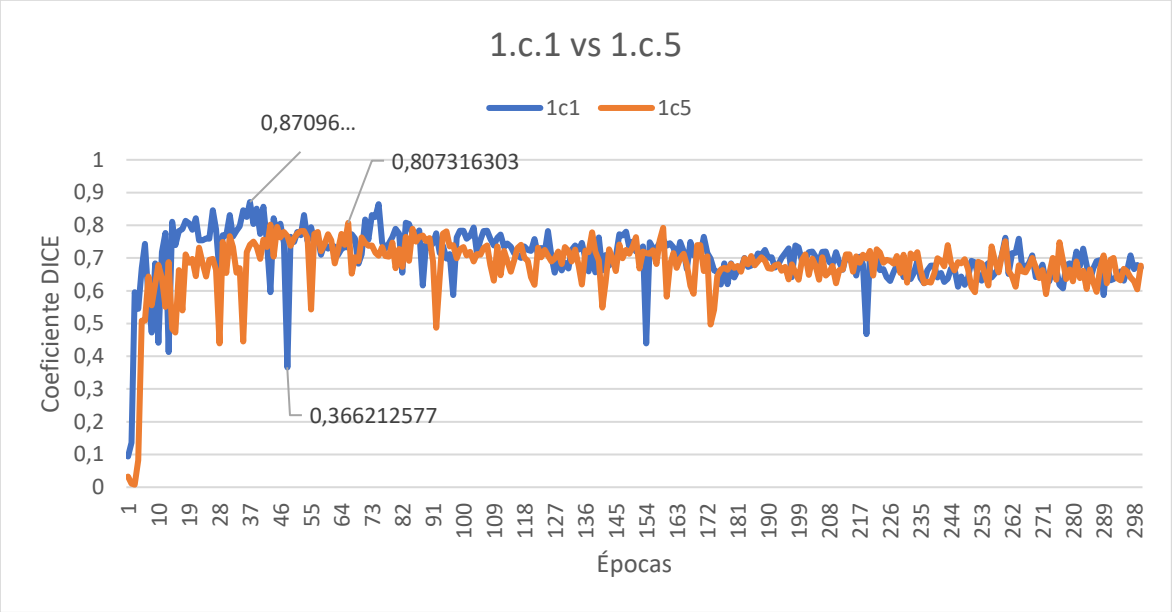

Figura 12. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.1 y 1.c.5

Se observan mejores Dice's máximos (0,87 frente a 0,8) y en promedio (0,7 frente a 0,67) cuando se utiliza un batch size de 10, aunque hay algunos mínimos como sobre la época 50 y la 155 aproximadamente, que son muy acentuados. Además, el modelo 1.c.1 tiene menor varianza (0,007 frente a 0,009).

2ª comparativa (1.c.2 con 1.c.6)

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios y tasa de aprendizaje 1,00E-4.

Se diferencian en que el modelo 1.c.2 tiene batch size de 10 y el 1.c.6 batch size de 20.

Id (1.c.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,644; Varianza DICE = 0,0091

Id (1.c.6). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4 y batch size de 20.*

Media DICE = 0,587; Varianza DICE = 0,0098

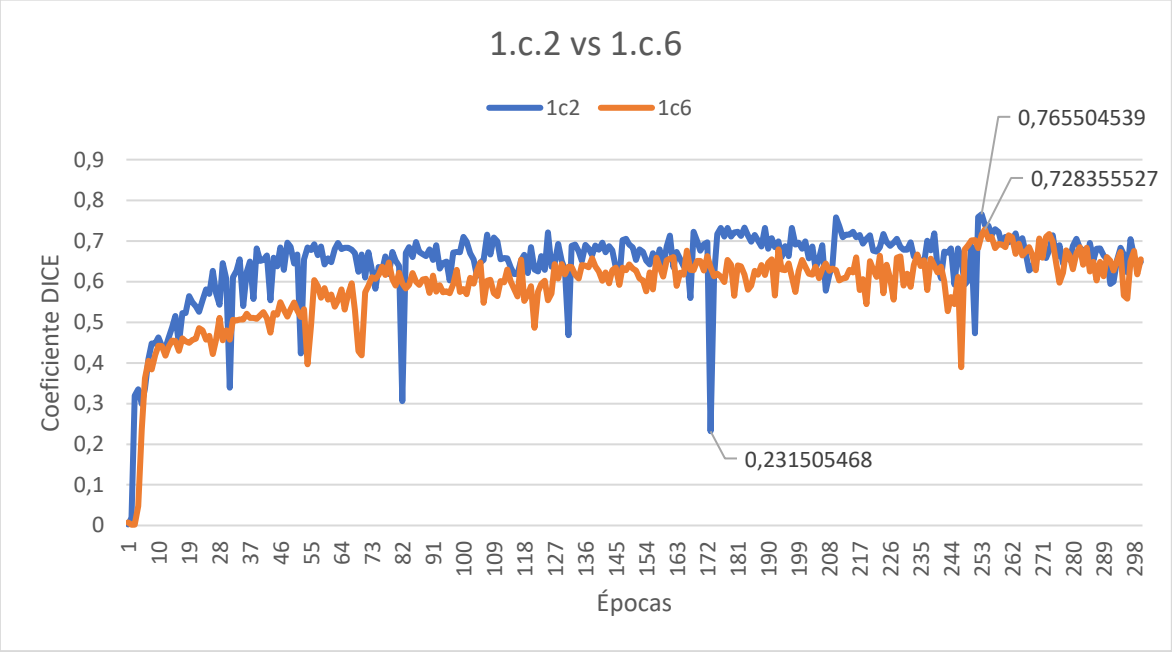

Figura 13. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.2 y 1.c.6.

Se trata de la misma comparación que antes, pero cambiando la tasa de aprendizaje de 1,00E-3 a 1,00E-4. Se observan, de nuevo, mejores Dice's con batch size de 10, ya que en promedio tienen un valor de 0,64 frente a un promedio de 0.59 cuando el batch size es de 20. Aunque se aprecia un mínimo de 0,23 sobre la época 170 en el modelo 1.c.2, sigue siendo mejor opción que el modelo 1.c.6. Asimismo, el modelo 1.c.2 tiene un máximo mayor (0,76 frente a 0,72).

3ª comparativa (1.c.3 con 1.c.7)

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado y tasa de aprendizaje 1,00E-3.

Se diferencian en que el modelo 1.c.3 tiene batch size de 10 y el 1.c.7 batch size de 20.

Id (1.c.3). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,772; Varianza DICE = 0,0043

Id (1.c.7). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 20.*

Media DICE = 0,712; Varianza DICE = 0,0028

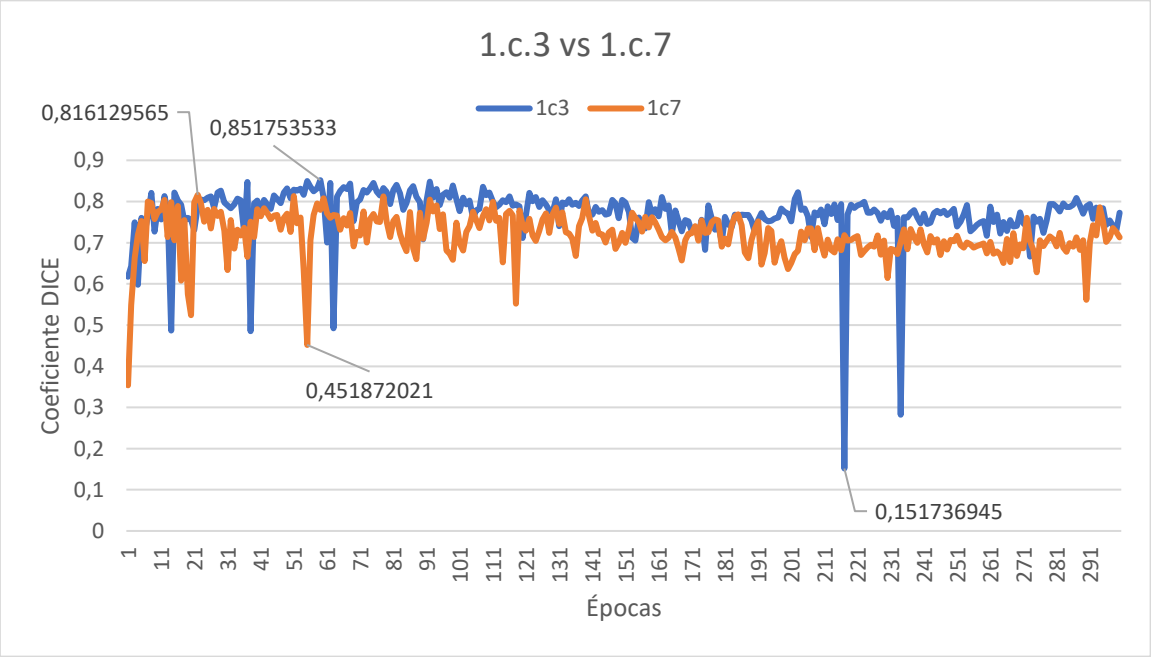

Figura 14. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.3 y 1.c.7.

En este caso, se mantiene la misma tasa de aprendizaje que en la primera comparativa, pero cambian los pesos iniciales, que se inicializan con unos de modelo de hígado. Se observa, otra vez, mejores Dice's para un batch size de 10 que de 20. En este caso, hay dos épocas donde los valores del Dice son muy bajos, reduciéndose prácticamente de 0,8 a 0,15. En el caso del modelo 1.c.7 con batch size de 20, el coeficiente DICE como mucho se reduce de 0,8 a 0,45 y posee menor varianza (0,0028 frente a 0,0043) aunque los valores en promedio siguen siendo más bajos (0,71 frente a 0,77). Es por esto, que el modelo 1.c.3 con batch size de 10 ofrece mejores resultados.

4ª comparativa (1.c.4 con 1.c.8)

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado y tasa de aprendizaje 1,00E-4.

Se diferencian en que el modelo 1.c.4 tiene batch size de 10 y el 1.c.8 batch size de 20.

Id (1.c.4). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,588; Varianza DICE = 0,002

Id (1.c.8). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-4 y batch size de 20.*

Media DICE = 0,519; Varianza DICE = 0,00004

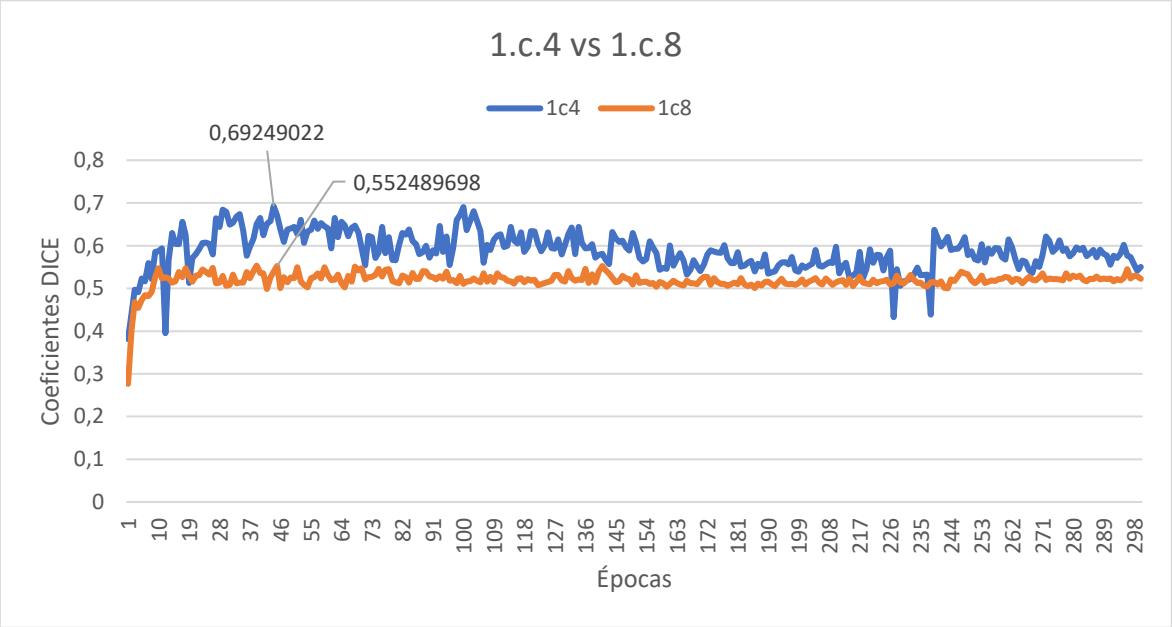

Figura 15. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.4 y 1.c.8.

Se tiene mayor promedio para el modelo 1.c.4 con batch size de 10 (0,59 frente a 0,52) aunque tiene mayor varianza (0,002 frente a 0,00004). El modelo 1.c.4 tiene mayor máximo que el modelo 1.c.8 (0,69 frente a 0,55) aunque también tiene mayores bajadas. Con esto se concluye que el modelo con batch size de 10 es mejor opción, ya que los coeficientes del modelo con batch size de 20 son demasiado bajos.

De estas comparativas podemos concluir que se prefiere un batch size de 10 que, de 20, a pesar de tener algunas caídas más pronunciadas. Se escogerán los pesos de épocas donde haya cierta estabilidad y no haya cerca ninguna subida o bajada pronunciada.

Con esto, limitamos la realización de próximos entrenamientos a un batch size de 10.

5.1.2. Comparativas para seleccionar la tasa de aprendizaje.

Se muestran a continuación las gráficas de resultados teniendo ahora solo en cuenta las de batch size de 10 y comparando los diferentes estudios con distintas tasas de aprendizaje.

5ª comparativa (1.c.1 con 1.c.2)

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios y batch size de 10.

Se diferencian en que el modelo 1.c.1 tiene una tasa de aprendizaje de 1,00E-3 y el 1.c.2 de 1,00E-4.

Id (1.c.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE,300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,704; Varianza DICE = 0,0071

Id (1.c.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,644; Varianza DICE = 0,0091

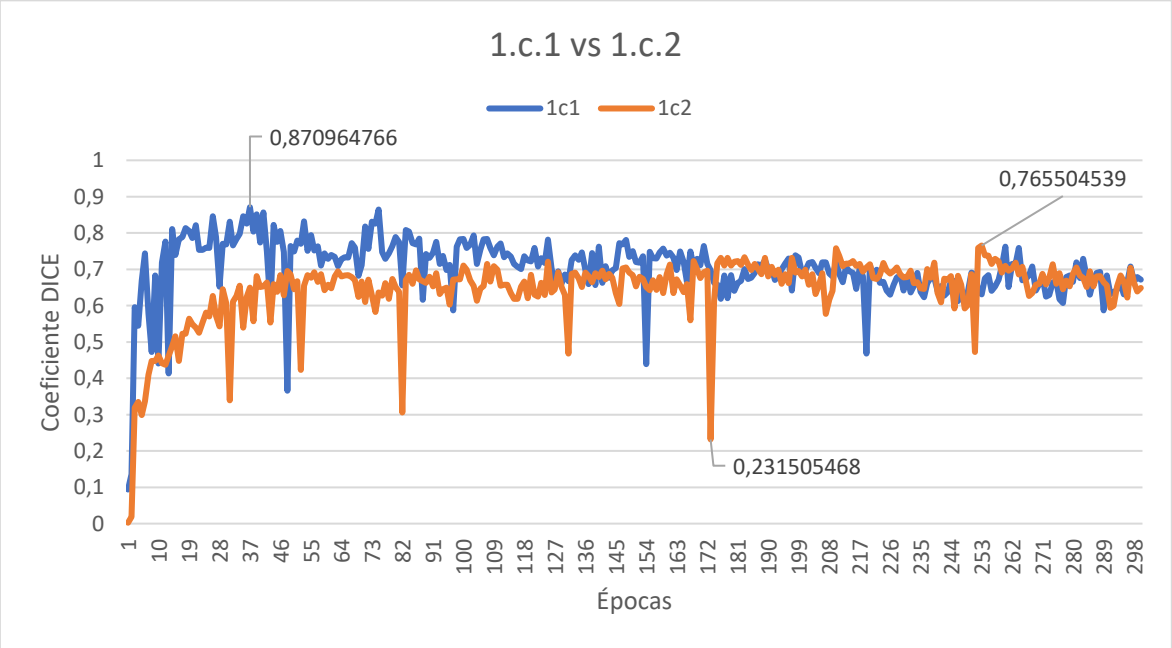

Figura 16. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.1 y 1.c.2.

Se observan Dice's más altos con la tasa de aprendizaje de 1,00E-3 aunque disminuyen un poco con el paso de las épocas. De todos modos, el promedio de los coeficientes Dice (0,7 frente 0,64) y el valor máximo (0,87 frente a 0,76) obtenido con la tasa de aprendizaje de 1,00E-3 son mejores que con la tasa de 1,00E-4.

6ª comparativa (1.c.3 con 1.c.4)

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado y batch size de 10.

Se diferencian en que el modelo 1.c.3 tiene una tasa de aprendizaje de 1,00E-3 y el 1.c.4 de 1,00E-4.

Id (1.c.3). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,772; Varianza DICE = 0,0043

Id (1.c.4). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,588; Varianza DICE = 0,002

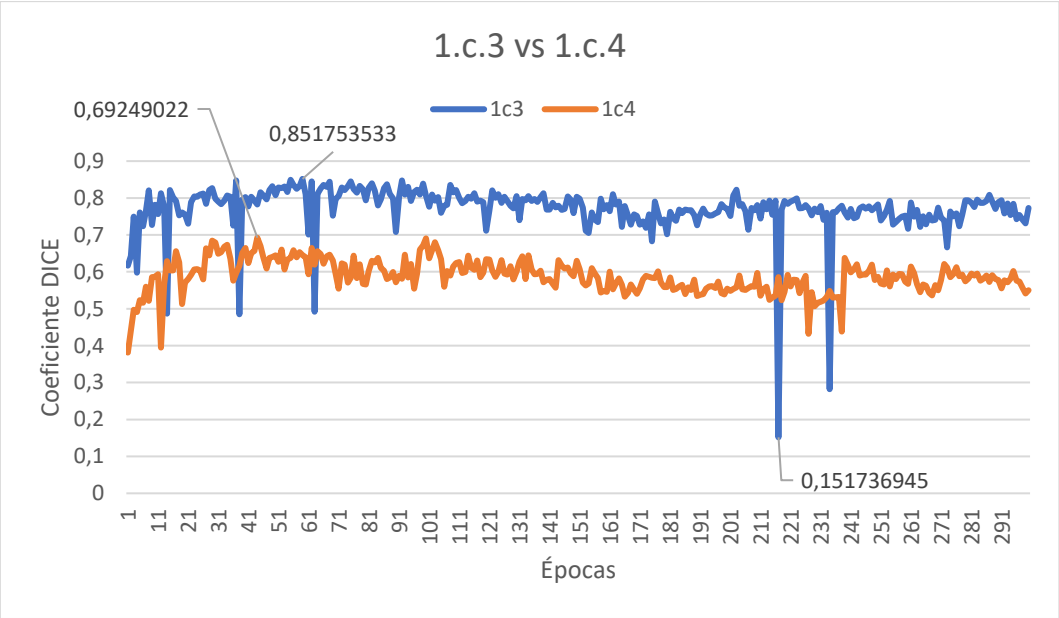

Figura 17. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.3 y 1.c.4.

Se observa, con diferencias significativas, un mejor promedio de Dice's (0,77 frente a 0,59) para la tasa de aprendizaje de 1,00E-3 a pesar de dos fuertes caídas.

En este nuevo análisis centrado en las tasas de aprendizaje, se observa que hay mejores Dice's con la tasa de aprendizaje de 1,00E-3 que con la de 1,00E-4, aunque produce algunas bajadas pronunciadas puntualmente.

5.1.3. Comparativas para seleccionar los pesos de inicialización.

Para comparar las diferentes inicializaciones, se vuelven a mostrar las gráficas de resultados, pero ahora agrupadas solo cuando cambia la inicialización, y manteniendo los hiperparámetros que mejor han funcionado hasta el momento: batch size de 10 y tasa de aprendizaje de 1,00E-3.

Aquí se hace uso del transfer learning al utilizar los pesos de un modelo pre-entrenado, concretamente de hígado. Este modelo fue entrenado para solucionar un problema similar al que se quiere resolver, es decir, fue entrenado para segmentar el hígado y aquí se quiere segmentar los riñones. Esto nos sirve para cuando existe escasez de muestras, como es el caso.

<u>7ª comparativa (1.c.1 con 1.c.3)</u>

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.c.1 se inicializa con pesos iniciales aleatorios y el 1.c.3 con pesos iniciales de hígado.

Id (1.c.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,704; Varianza DICE = 0,0071

Id (1.c.3). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,772; Varianza DICE = 0,0043

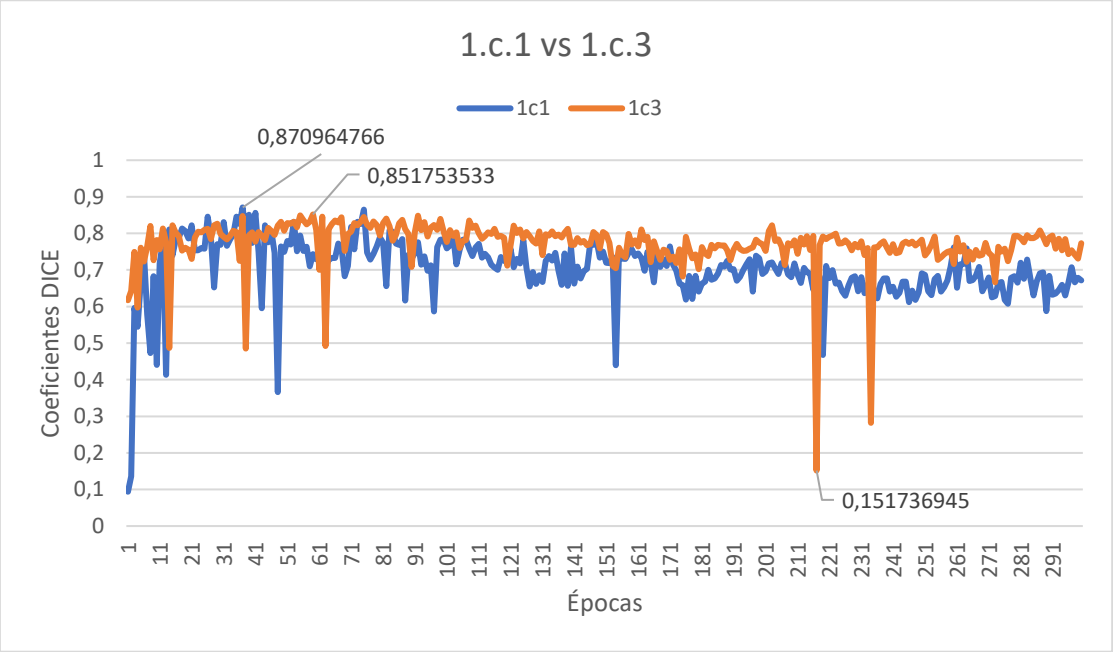

Figura 18. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.1 y 1.c.3.

Se observa cierta similitud en los coeficientes, aunque el promedio de los coeficientes del modelo 1.c.3 que utiliza transfer learning es mayor (0,77 frente a 0,7) y también se aprecia como este modelo se mantiene por encima en la mayor parte de las épocas. A pesar de esto, el mayor DICE lo obtenemos en el modelo 1.c.1 con 0,87 frente al DICE máximo del modelo 1.c.3 que es 0,85. También cabe mencionar la caída del coeficiente en el modelo 1.c.3 pasando de 0,8 a 0,15, aunque este modelo tiene menor varianza (0,0043 frente a 0,007). En general, estos datos muestran que el modelo inicializado con pesos de hígado por transfer learning es mejor que el inicializado con pesos aleatorios. De todos modos, se seguirá analizando la inicialización de los modelos en próximas comparativas.

5.1.4. Comparativas para seleccionar el número de épocas

Ahora, se van a comparar modelos entrenados con 30, 100 y 300 épocas. Todos ellos con un batch size de 10 y una tasa de aprendizaje de 1,00E-3. Primero para pesos iniciales aleatorios y luego para pesos iniciales de hígado.

8ª comparativa (1.a.1, 1.b.1 y 1.c.1)

Estos tres modelos tienen en común: entrenados con secuencias MECSE y THRIVE, pesos iniciales aleatorios, tasa de aprendizaje 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.a.1 es entrenado con 30 épocas, el modelo 1.b.1 con 100 y el modelo 1.c.1 con 300.

Id (1.a.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE, 30 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,667; Varianza DICE = 0,0506

Id (1.b.1). *Gráfica obtenida del modelo de riñón entrenado sobre secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,761; Varianza DICE = 0,0176

Id (1.c.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,704; Varianza DICE = 0,0071

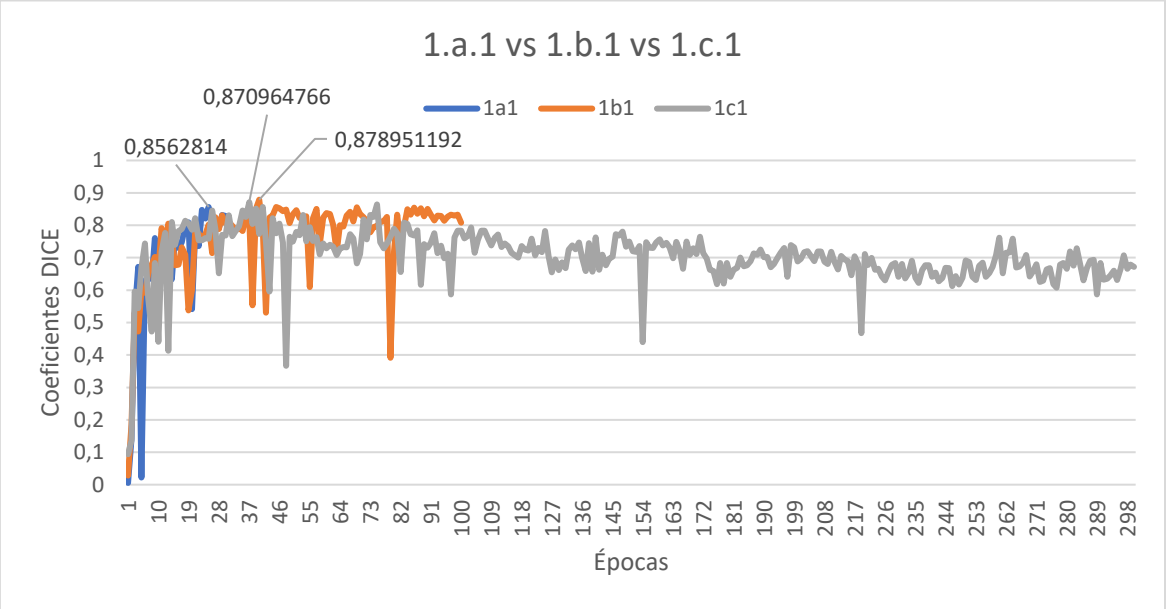

Figura 19. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.a.1, 1.b.1 y 1.c.1.

Se observa que 30 épocas son demasiado pocas para observar una tendencia y compararla con otros modelos, pues cuantas menos épocas se tienen mayor es la varianza del DICE (0,0071 para 300 épocas, 0,0176 para 100 épocas y 0,0506 para 30 épocas). En cuanto a los modelos de 100 y 300 épocas, los valores de los coeficientes son parecidos (con un promedio de 0,76 y 0,7 respectivamente), así como también son parecidos los valores máximos de DICE (0,871 y 0,879 respectivamente). Por lo tanto, 100 épocas son suficientes para predecir el comportamiento y sacar conclusiones, pudiendo prescindir de realizar todos los entrenamientos con 300 épocas siempre.

<u>9ª comparativa (1.a.2, 1.b.2 y 1.c.3)</u>

Estos tres modelos tienen en común: entrenados con secuencias MECSE y THRIVE, pesos iniciales de hígado, tasa de aprendizaje 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.a.2 es entrenado con 30 épocas, el modelo 1.b.2 con 100 y el modelo 1.c.3 con 300.

Id (1.a.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 30 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,762; Varianza DICE = 0,0103

Id (1.b.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,776; Varianza DICE = 0,005

Id (1.c.3). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,772; Varianza DICE = 0,0043

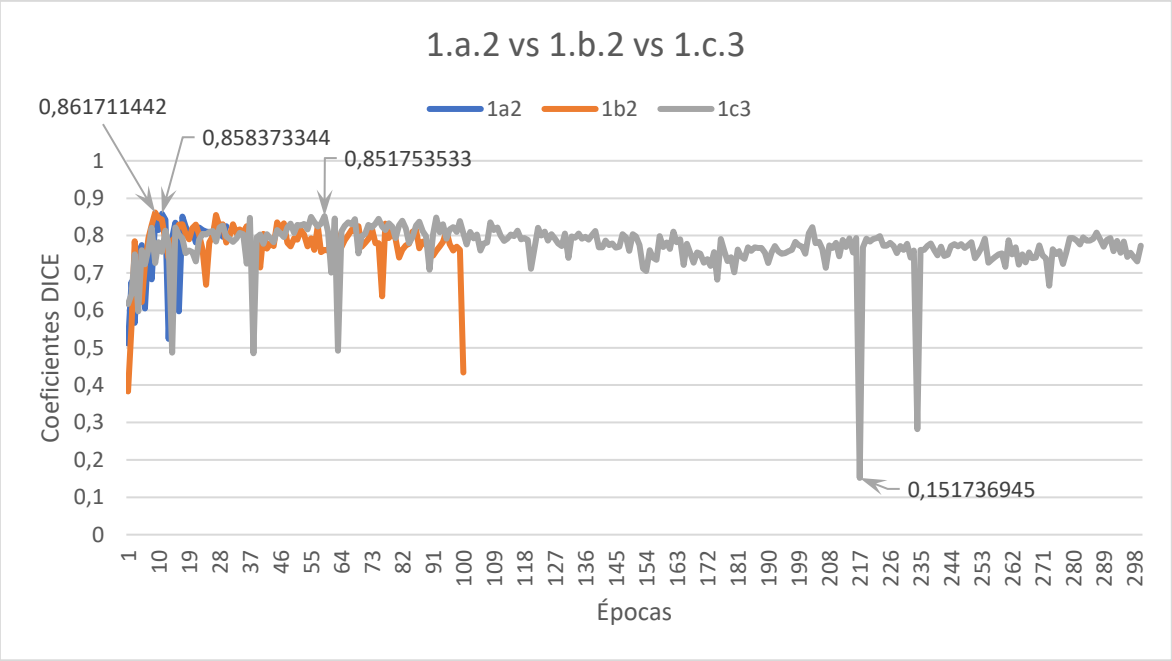

Figura 20. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.a.2, 1.b.2 y 1.c.3.

Nos encontramos ante la misma problemática que en la comparativa anterior, con 30 épocas hay demasiada varianza en el DICE y no se intuye bien la tendencia de los coeficientes por falta de datos. En cambio, si se comparan los modelos de 100 y 300 épocas, tienen una varianza similar (0,005 y 0,0043 respectivamente) y prácticamente el mismo promedio (0,776 y 0,772 respectivamente). En cuanto a los máximos son también casi iguales (0,858 para 100 épocas y 0,852 para 300 épocas).

Por tanto, se puede afirmar que los entrenamientos con 100 épocas generalizan bien el comportamiento del modelo. Es por esto, que los siguientes entrenamientos serán restringidos a 100 épocas.

5.1.5. Comparativa con diferentes inicializaciones.

Ahora, se vuelve a analizar el efecto del transfer learning con la inicialización de los modelos, pero entrenados con menor cantidad de épocas.

<u>10ª comparativa (1.b.1 con 1.b.2)</u>

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 100 épocas, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.b.1 es entrenado con pesos iniciales aleatorios y el modelo 1.b.2 con pesos iniciales de hígado.

Id (1.b.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,761; Varianza DICE = 0,0176

Id (1.b.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,776; Varianza DICE = 0,005

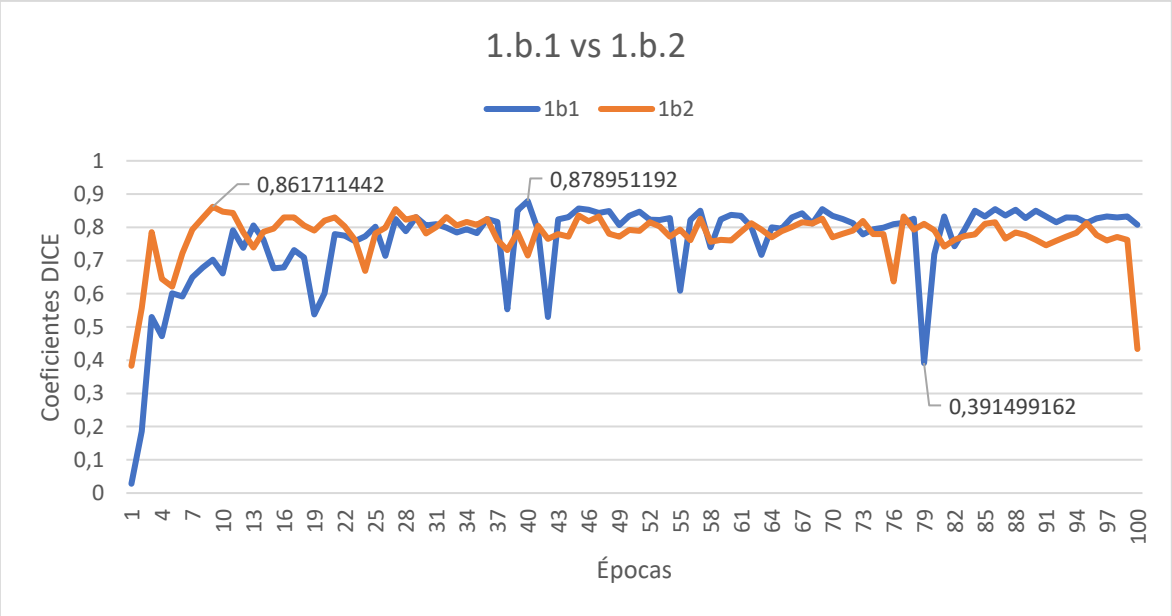

Figura 21. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.b.1 y 1.b.2.

Si se observa el promedio de los coeficientes, el modelo 1.b.2 tiene mayor promedio por poco (0,776 frente a 0,761) y menor varianza (0,005 frente a 0,0176). A pesar de esto, el valor máximo del modelo 1.b.2 es menor que el valor máximo del modelo 1.b.1 (0,862 frente a 0,879). Además, observando la figura 21 parece que en las últimas épocas el modelo 1.b.2 sea superior a pesar de tener menor promedio. Debido a esta similitud entre modelos se decide recurrir a la misma comparativa, pero con 300 épocas (ya realizada en el apartado 5.1.3 anteriormente) y así disponer de más datos para poder analizar con mayor precisión y salir de dudas.

7ª comparativa (1.c.1 con 1.c.3) → TRAÍDA DEL APARTADO 5.1.3

Estos dos modelos tienen en común: entrenados con secuencias MECSE y THRIVE, 300 épocas, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.c.1 se inicializa con pesos iniciales aleatorios y el 1.c.3 con pesos iniciales de hígado.

Id (1.c.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,704; Varianza DICE = 0,0071

Id (1.c.3). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,772; Varianza DICE = 0,0043

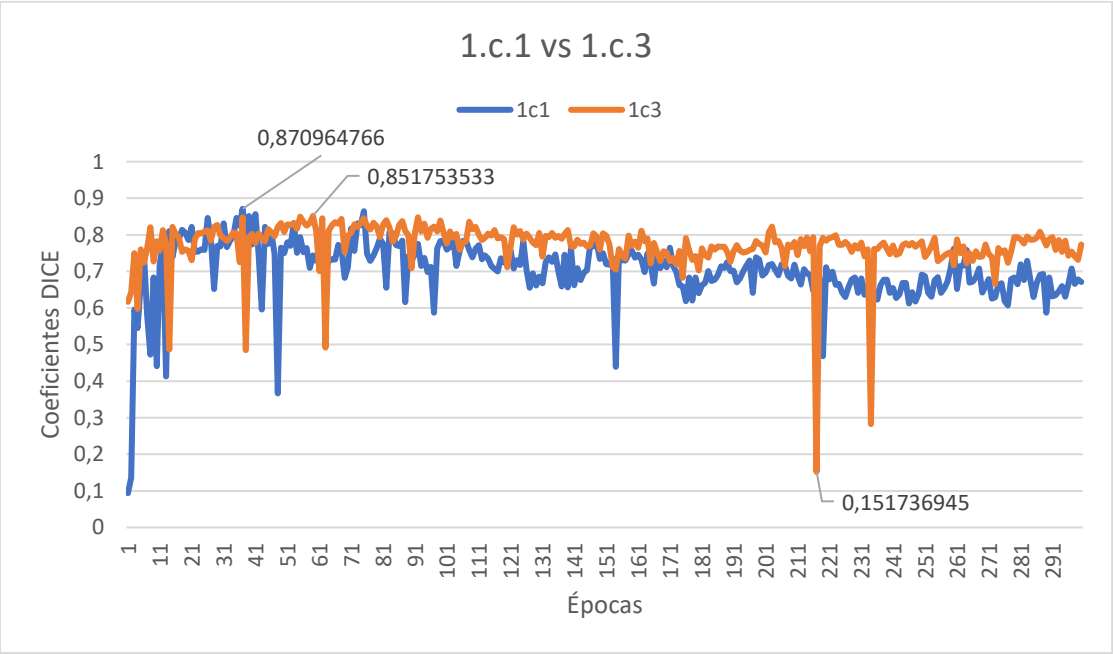

Figura 18. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.1 y 1.c.3.

5.2. Modelo de riñón entrenado con secuencia MECSE

Tras el estudio anterior, se mantienen los hiperparámetros elegidos de 100 épocas y batch size de 10, variando primero la tasa de aprendizaje y después la inicialización de los pesos en el modelo de riñón entrenado con secuencia MECSE. Posteriormente se comparará este modelo entrenado con secuencia MECSE con el modelo entrenado con secuencias MECSE y THRIVE.

5.2.1. Comparativa de la tasa de aprendizaje con pesos iniciales de hígado

Aquí se compara la tasa de aprendizaje 1,00E-3 frente a 1,00E-4 para los modelos de riñón entrenados con secuencia MECSE entrenados con 100 épocas, batch size de 10 e inicializados con los pesos de hígado.

1ª comparativa (2.a.3 con 2.a.4)

Estos dos modelos tienen en común: entrenados con secuencia MECSE, 100 épocas, pesos iniciales de hígado y batch size de 10.

Se diferencian en que el modelo 2.a.3 tiene una tasa de aprendizaje de 1,00E-3 y el 2.a.4 de 1,00E-4.

Id (2.a.3). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,784; Varianza DICE = 0,0048

Id (2.a.4). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,511; Varianza DICE = 0,0016

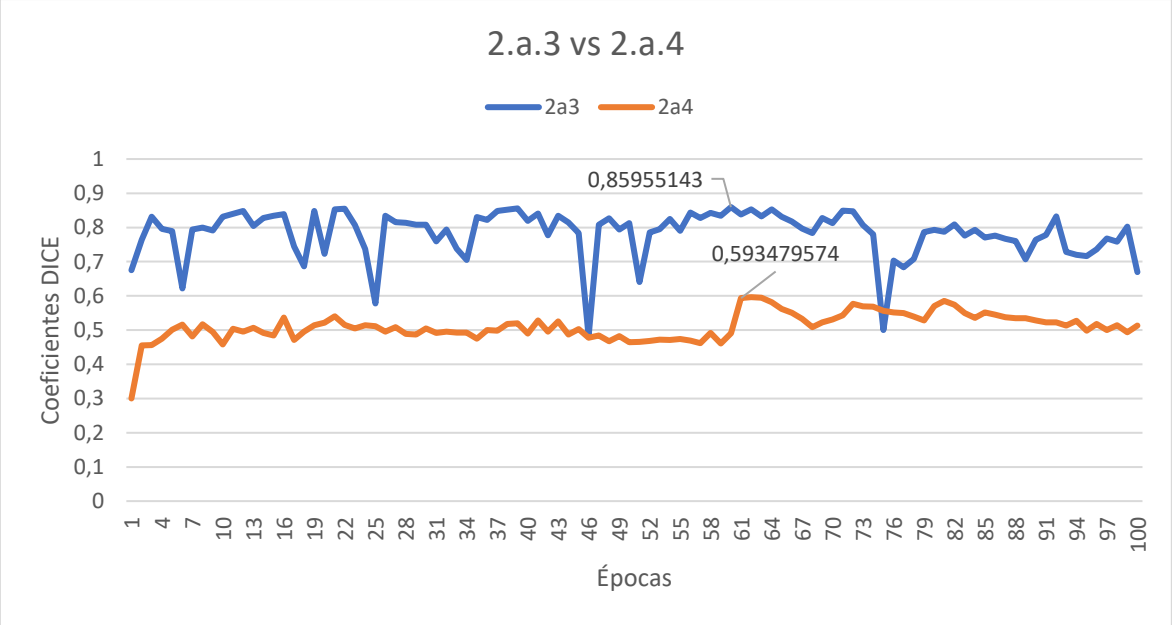

Figura 22. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 2.a.3 y 2.a.4.

La gráfica de la figura 22 habla por sí sola. Se observa que el modelo 2.a.3 tiene un promedio de DICE's mayor (0,784 frente a 0,511) y un máximo mayor (0,859 frente a 0,593). Se concluye que la tasa de aprendizaje de 1,00E-3 utilizada en el modelo 2.a.3 es mejor que la tasa de aprendizaje de 1,00E-4 utilizada en el modelo de 2.a.4 a pesar de tener menor varianza este último modelo (0,0016 frente a 0,0048 del modelo 2.a.3).

5.2.2. Comparativa de la tasa de aprendizaje con pesos iniciales aleatorios

Ahora se compara la tasa de aprendizaje 1,00E-3 frente a 1,00E-4 para los modelos de riñón entrenados con secuencia MECSE entrenados con 100 épocas, batch size de 10 e inicializados con los pesos aleatorios.

2ª comparativa (2.a.1 con 2.a.2, y 2.b.1)

Estos dos modelos tienen en común: entrenados con secuencia MECSE, 100 épocas, pesos iniciales aleatorios y batch size de 10.

Se diferencian en que el modelo 2.a.1 tiene una tasa de aprendizaje de 1,00E-3 y el 2.a.2 de 1,00E-4.

Id (2.a.1). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,72; Varianza DICE = 0,0307

Id (2.a.2). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,529; Varianza DICE = 0,0215

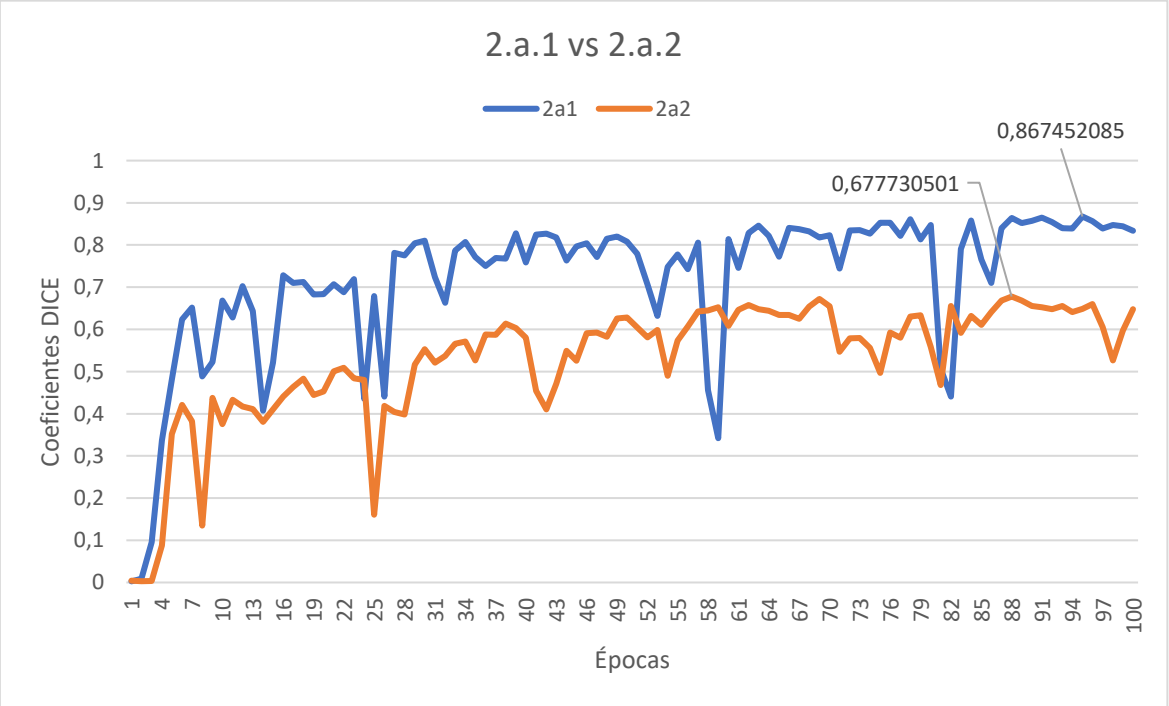

Figura 23. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 2.a.1 y 2.a.2.

En este caso, se tiene mejores coeficientes DICE también para la tasa de aprendizaje de 1,00E-3 (en promedio 0,72 frente a 0,53), aunque para la tasa de 1,00E-4 se observa una buena progresión, que con más épocas podría ser interesante, ya que tiene menor varianza (0,0215 frente a 0,0307) y tiene tendencia alcista. Se añade ese caso de 300 épocas:

Id (2.b.1). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-4 y batch size de 10.*

Media DICE = 0,658; Varianza DICE = 0,0199

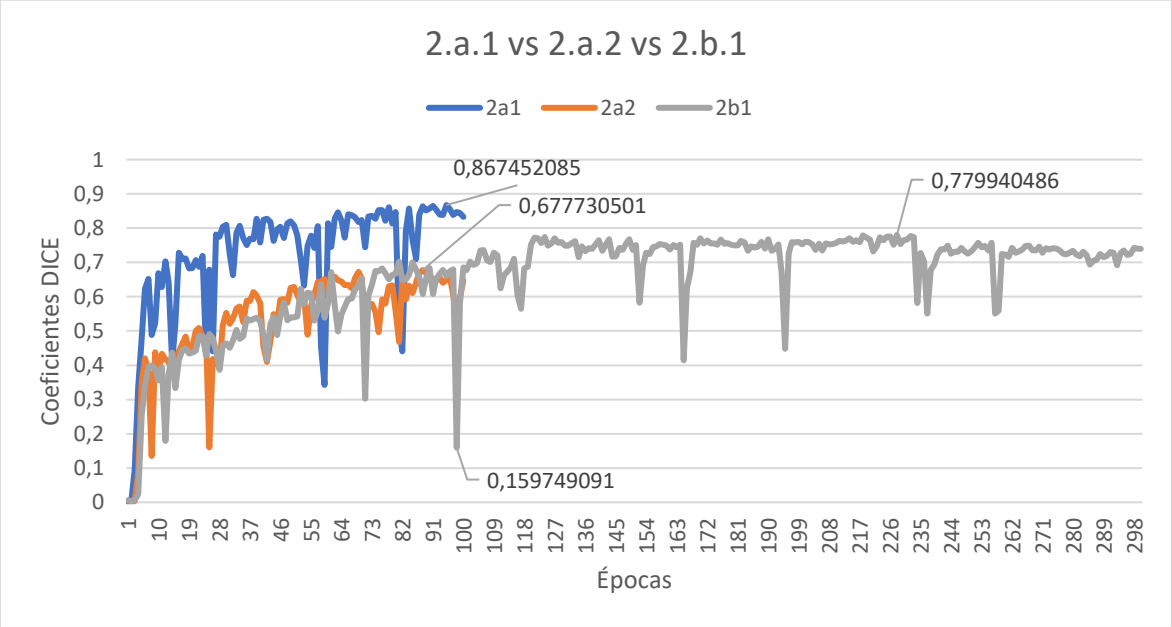

Figura 24. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 2.a.1, 2.a.2 y 2.b.1.

Se observa que mejora, pero igualmente no llega al promedio del modelo 2.a.1 (0,658 frente a 0,72) y también tiene algunas bajadas considerables como el mínimo de 0,16 sobre la época 100. Como ya se había observado antes, la tasa de aprendizaje de 1,00E-3 muestra mejores resultados, y por ello se utilizará para los siguientes entrenamientos, descartando la tasa de aprendizaje de 1,00E-4.

5.2.3. Comparativa con diferentes inicializaciones

En este apartado, se comparan las inicializaciones mediante el uso de transfer learning igual que se ha hecho anteriormente, pero para el modelo de riñón entrenado con secuencia MECSE, en lugar de con MECSE y THRIVE.

Primero se compara para 100 épocas, con batch size de 10 y tasa de aprendizaje de 1,00E-3, solamente cambiando la inicialización de los pesos.

<u>3ª comparativa (2.a.1 con 2.a.3)</u>

Estos dos modelos tienen en común: entrenados con secuencia MECSE, 100 épocas, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 2.a.1 tiene pesos iniciales aleatorios y el 2.a.3 pesos iniciales de hígado.

Id (2.a.1). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,72; Varianza DICE = 0,0307

Id (2.a.3). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,784; Varianza DICE = 0,0048

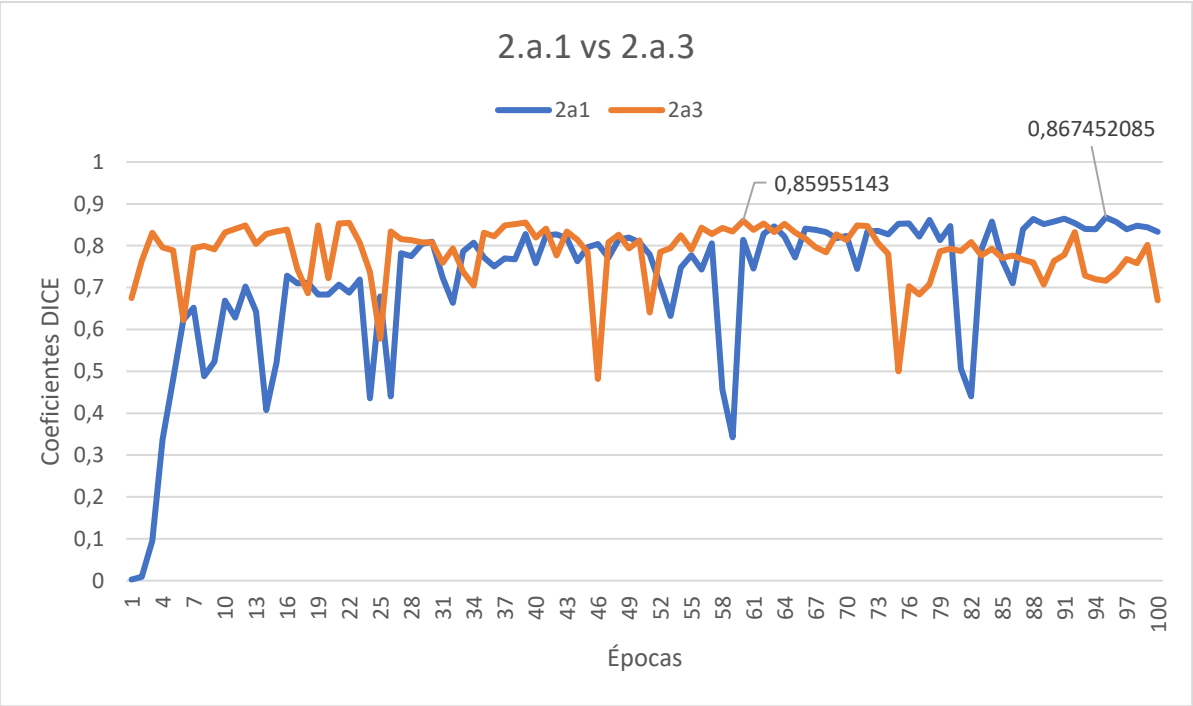

Figura 25. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 2.a.1 y 2.a.3.

Se observan mejores coeficientes en el modelo 2.a.3 en la mayor parte de las épocas, aunque sobre las últimas no tanto. En promedio, el modelo 2.a.3 inicializado con hígado tiene valores mayores (0,784 frente a 0,72 del modelo 2.a.1 inicializado aleatoriamente). A esto se le suma que también tiene menor varianza (0,0048 frente a 0,0307). Aun así, el modelo 2.a.1 inicializado aleatoriamente tiene un máximo mayor (0,867 frente a 0,859). Hay bastante similitud como para concluir algo significativo.

Entonces, se comparan los modelos con 300 épocas, batch size de 10 y tasa de aprendizaje de 1,00E-3, para las dos inicializaciones.

<u>4ª comparativa (2.b.2 con 2.b.3)</u>

Estos dos modelos tienen en común: entrenados con secuencia MECSE, 300 épocas, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 2.b.2 tiene pesos iniciales aleatorios y el 2.b.3 pesos iniciales de hígado.

Id (2.b.2). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 300 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,761; Varianza DICE = 0,0154

Id (2.b.3). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,791; Varianza DICE = 0,0045

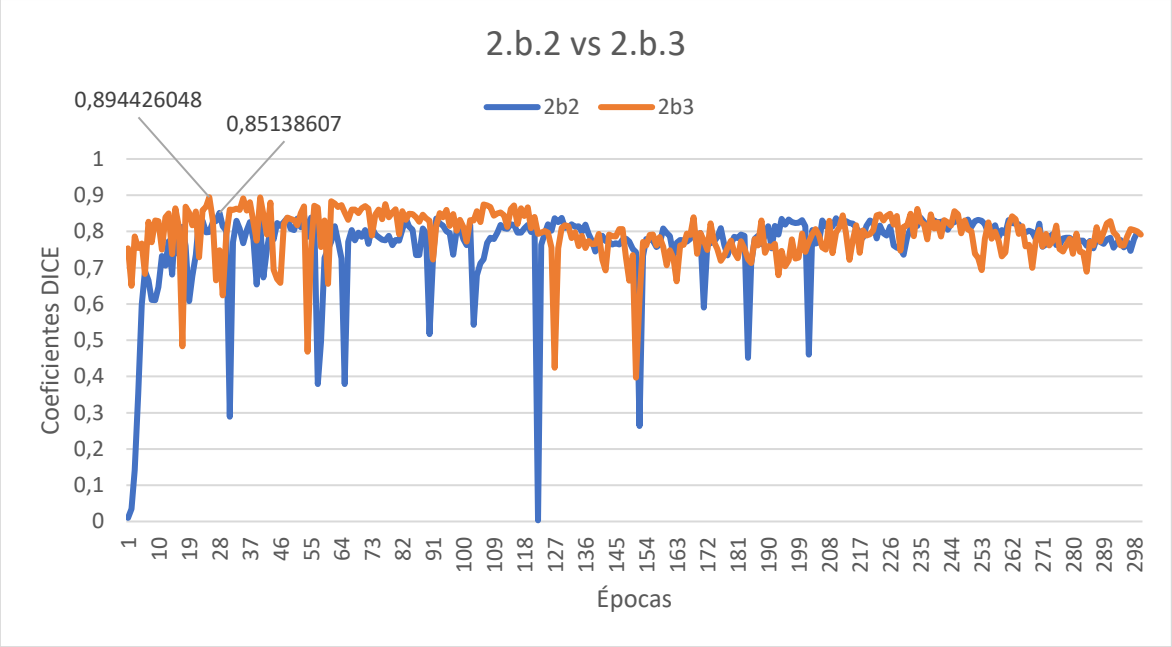

Figura 26. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 2.b.2 y 2.b.3.

Analizando el promedio de los coeficientes, se observa un mayor promedio cuando se hace uso del transfer learning con pesos iniciales de hígado en el modelo 2.b.3 (0,791 frente a 0,761), algo que también se observa gráficamente en la figura 26. Asimismo, el modelo 2.b.3 también tiene menor varianza (0,0045 frente a 0,0154) y mayor valor máximo (0,89 frente a 0,85). Se concluye, que para el modelo de riñón entrenado con secuencia MECSE, se obtienen mejores resultados con los pesos iniciales de hígado que con los pesos iniciales aleatorios.

5.2.4. Comparativa en función de las secuencias de segmentación

Ahora se van a comparar los modelos de riñones entrenados con MECSE y THRIVE frente a los entrenados solo con MECSE. Para estas comparativas, se utilizan los modelos de 100 épocas, 10 de batch size y tasa de aprendizaje de 1,00E-3.

5ª comparativa (1.b.2 con 2.a.3)

Estos dos modelos tienen en común: 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.b.2 está entrenado con secuencias MECSE y THRIVE y el 2.a.3 sobre secuencia MECSE.

Id (1.b.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,776; Varianza DICE = 0,005

Id (2.a.3). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,784; Varianza DICE = 0,0048

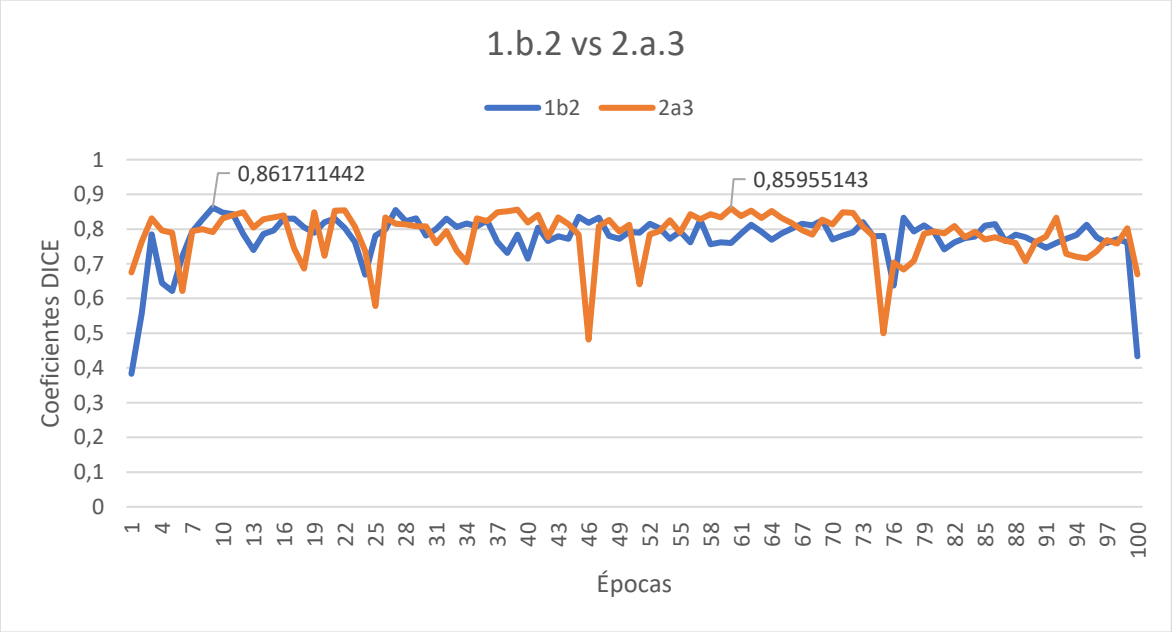

Figura 27. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.b.2 y 2.a.3.

Se observan dos modelos muy igualados, que tienen coeficientes promedios de 0,776 para el modelo 1.b.2 y 0,784 para el modelo 2.a.3. También tienen prácticamente la misma varianza (aproximadamente 0,005) y los valores máximos son muy similares (0,861 en 1.b.2 y 0,859 en 2.a.3). Con estos resultados no se puede concluir nada significativo.

Para analizar mejor esta comparativa se analizan los mismos modelos, pero con 300 épocas.

6ª comparativa (1.c.3 con 2.b.3)

Estos dos modelos tienen en común: 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.c.3 está entrenado con secuencias MECSE y THRIVE y el 2.b.3 sobre secuencia MECSE.

Id (1.c.3). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,772; Varianza DICE = 0,0043

Id (2.b.3). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,791; Varianza DICE = 0,0045

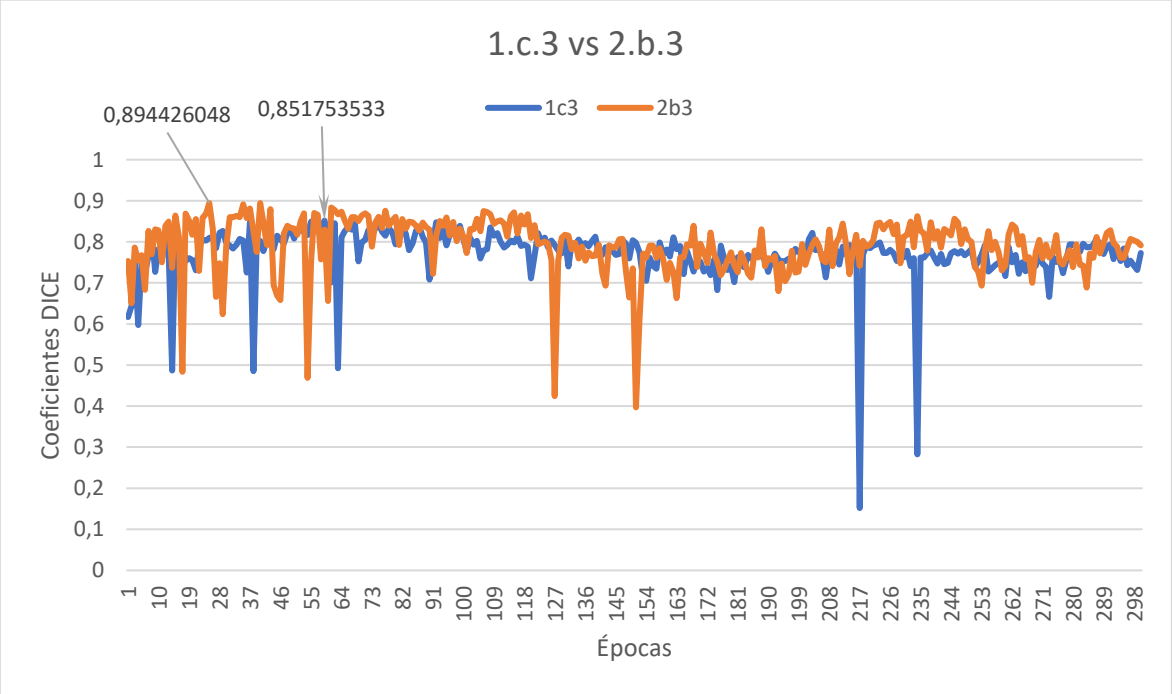

Figura 28. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.c.3 y 2.b.3.

Se observa que el modelo 2.b.3 entrenado con secuencia MECSE ofrece resultados mejores que el modelo 1.c.3 entrenado con secuencias MECSE y THRIVE. Tiene mayor promedio de coeficientes (0,791 frente a 0,772) y un máximo mayor (0,894 frente a 0,852). La varianza es prácticamente la misma (≈0,0045).

7ª comparativa (1.b.1 con 2.a.1)

Estos dos modelos tienen en común: 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.b.1 está entrenado con secuencia MECSE y THRIVE y el 2.a.1 sobre secuencia MECSE.

Id (1.b.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,761; Varianza DICE = 0,0176

Id (2.a.1). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,72; Varianza DICE = 0,0307

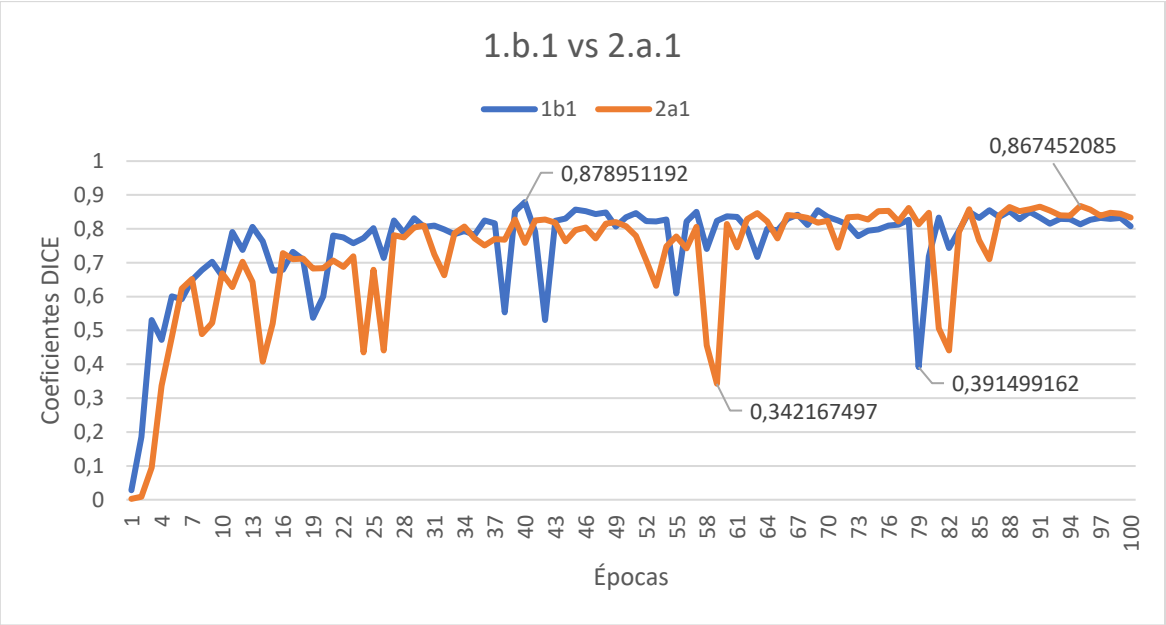

Figura 29. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.b.1 y 2.a.1.

Se observa que el modelo 1.b.1 es un poco superior, pues está durante más épocas por encima. En cuanto a los promedios, el modelo 1.b.1 es superior (0,761 frente a 0,72) y tiene menor varianza (0,0176 frente a 0,0307). También tiene un máximo mayor (0,878 frente a 0,867). En general, parece que están bastante igualados, aunque el modelo 1.b.1 entrenado con secuencias MECSE y THRIVE obtiene resultado un tanto mejores.

De estos resultados, se observa que el modelo entrenado con secuencias MECSE y THRIVE obtiene mejores resultados que el entrenado con secuencia MECSE para pesos iniciales aleatorios. En cambio, para pesos iniciales de hígado, ocurre lo contrario, es decir, el modelo entrenado con MECSE obtiene mejores resultados que el entrenado con MECSE y THRIVE. La explicación de esto es que los pesos de hígado fueron obtenidos en un modelo entrenado con MECSE, entonces el dominio inicial y final es el mismo, y esto hace que se obtengan mejores resultados.

5.3. Modelo de riñón entrenado con secuencia THRIVE

Una vez ya establecidos los hiperparámetros que mejor funcionan en los entrenamientos anteriores: 100 épocas (aunque puntualmente se alarga a 300 cuando hay dudas), batch size de 10 y tasa de aprendizaje de 1,00E-3. Se fijan estos hiperparámetros y se compararán las diferentes inicializaciones mediante el tranfer learning y el uso de las diferentes secuencias.

5.3.1. Comparativa con diferentes inicializaciones

Ahora se compara para 100 épocas, con batch size de 10 y tasa de aprendizaje de 1,00E-3, solamente cambiando la inicialización de los pesos.

1ª comparativa (3.a.1 con 3.a.2)

Estos dos modelos tienen en común: entrenados con secuencia THRIVE, 100 épocas, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 3.a.1 está inicializado con pesos aleatorios y el modelo 3.a.2 con pesos de hígado mediante transfer learning.

Id (3.a.1). *Gráfica obtenida del modelo de riñón entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,604; Varianza DICE = 0,0262

Id (3.a.2). *Gráfica obtenida del modelo de riñón entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,722; Varianza DICE = 0,0073

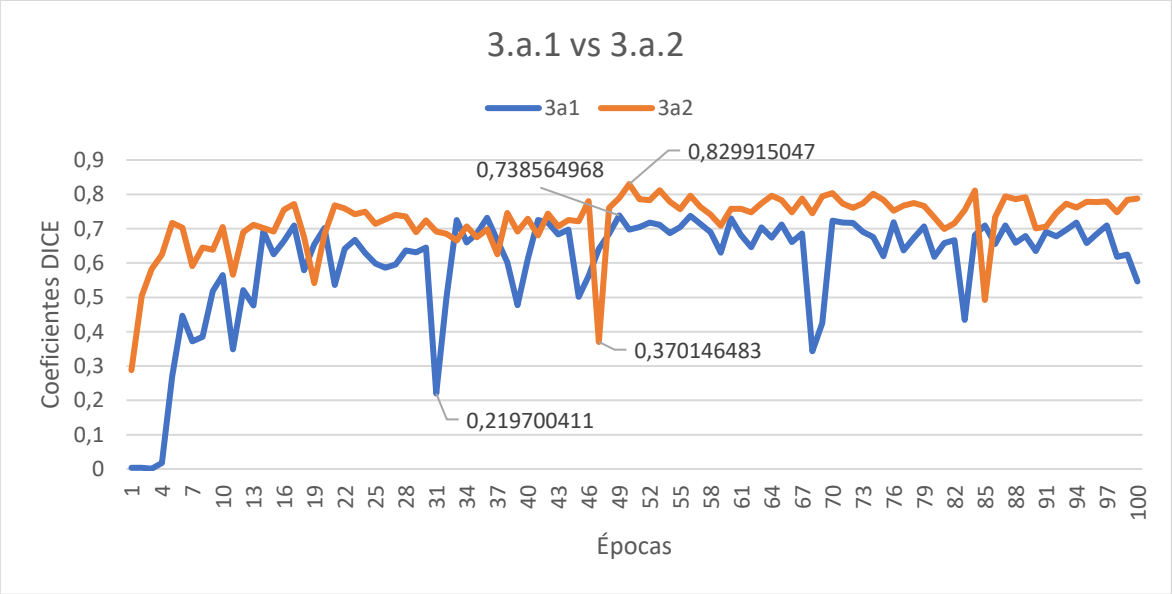

Figura 30. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 3.a.1 y 3.a.2.

Se observa que el modelo 3.a.2 inicializado con hígado mediante transfer learning tiene mejores resultados que el modelo 3.a.1 inicializado aleatoriamente, con un promedio más alto (0,722 frente a 0,604) y con menor varianza (0,0073 frente a 0,0262). Asimismo, tiene un máximo más alto (0,829 frente a 0,738). Con estos datos calculados y observando como en la figura 29 está por encima, se concluye que se obtienen mejores resultados aplicando transfer learning con pesos iniciales de hígado.

5.3.2. Comparativa en función de las secuencias de segmentación

Ahora se comparan los modelos de riñones entrenados con MECSE y THRIVE frente a los entrenados solo con THRIVE. Para estas comparativas, se utilizan los modelos de 100 épocas, 10 de batch size y tasa de aprendizaje de 1,00E-3.

2ª comparativa (1.b.1 con 3.a.1)

Se comparan para pesos iniciales aleatorios, variando el modelo entrenado con ambas secuencias o solo sobre THRIVE.

Estos dos modelos tienen en común: 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.b.1 está entrenado con secuencias MECSE y THRIVE y el 3.a.1 sobre secuencia THRIVE.

Id (1.b.1). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,761; Varianza DICE = 0,0176

Id (3.a.1). *Gráfica obtenida del modelo de riñón entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,604; Varianza DICE = 0,0262

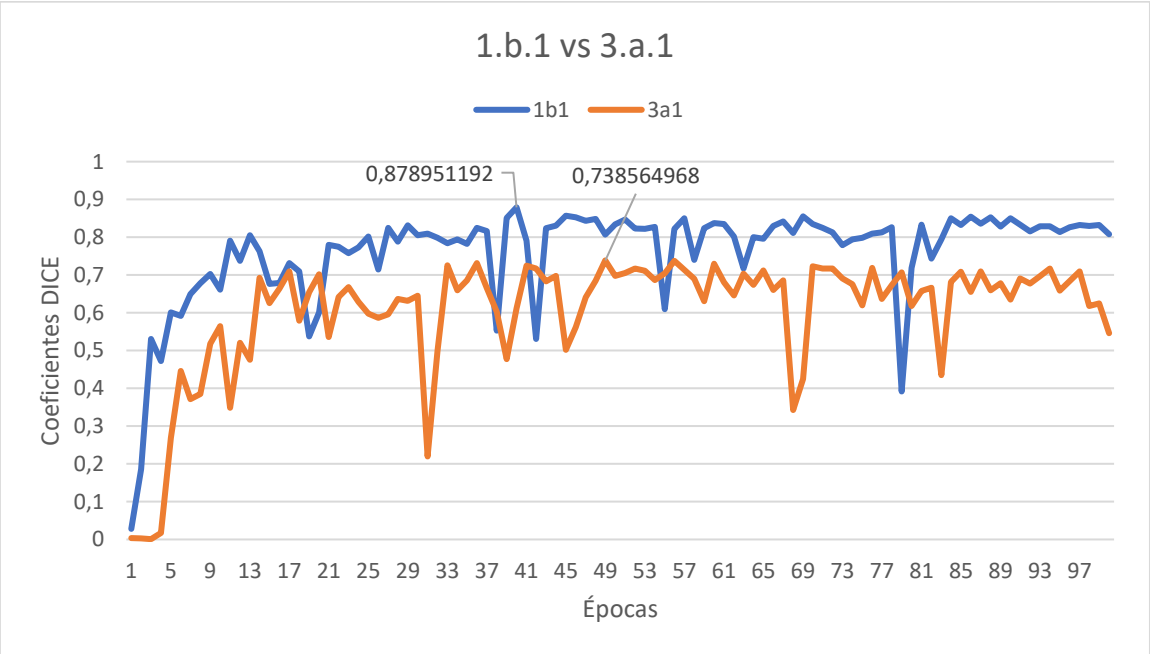

Figura 31. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.b.1 y 3.a.1.

Se observa que el modelo 1.b.1 entrenado con secuencias MECSE y THRIVE obtiene mejores coeficientes que el modelo 3.a.1 entrenado con secuencia THRIVE. De hecho, el modelo 1.b.1 tiene un promedio mayor (0,761 frente a 0,604) y menor varianza (0,0176 frente a 0,0262). Además, también ofrece un máximo mayor (0,878 frente a 0,738).

<u>3ª comparativa (1.b.2 con 3.a.2)</u>

Se comparan para pesos iniciales de hígado, variando el modelo entrenado con ambas secuencias o solo sobre THRIVE.

Estos dos modelos tienen en común: 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 1.b.2 está entrenado con secuencias MECSE y THRIVE y el 3.a.2 sobre secuencia THRIVE.

Id (1.b.2). *Gráfica obtenida del modelo de riñón entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,776; Varianza DICE = 0,005

Id (3.a.2). *Gráfica obtenida del modelo de riñón entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,722; Varianza DICE = 0,0073

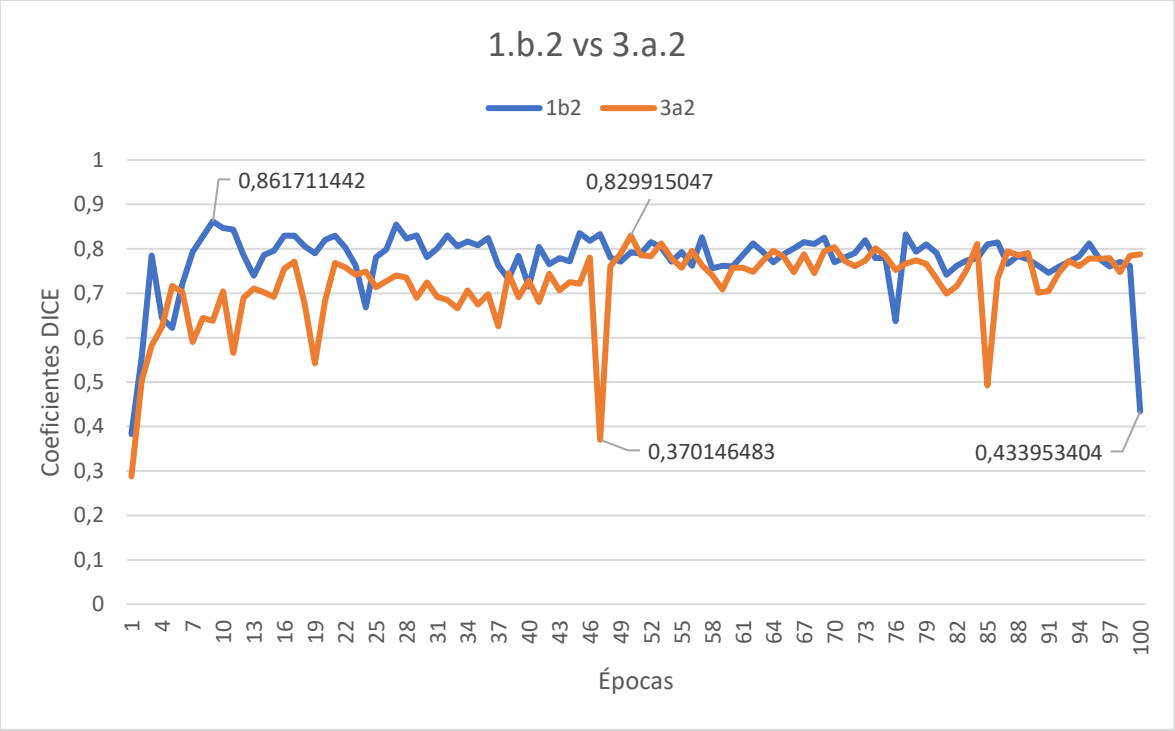

Figura 32. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 1.b.2 y 3.a.2.

Se observa que el modelo 1.b.2 entrenado con secuencias MECSE y THRIVE ofrece mejores coeficientes que el modelo entrenado con secuencia THRIVE, sobre todo en la primera mitad del entrenamiento. El modelo 1.b.2 tiene un promedio de 0,776 y el modelo 3.a.2 de 0,722, por tanto, el modelo 1.b.2 tiene un mayor promedio. Asimismo, el modelo 1.b.2 tiene menor varianza (0,005 frente a 0,0073) y un máximo más alto (0,862 frente a 0,829). Se concluye que el modelo 1.b.2 obtiene mejores resultados, pero no habría que seleccionar los pesos de la última época porque tienen un bajón del DICE que llega a 0,43.

<u>4ª comparativa (2.a.3 con 3.a.2)</u>

Estos dos modelos tienen en común: 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 2.a.3 está entrenado con secuencia MECSE y el 3.a.2 sobre secuencia THRIVE.

Id (2.a.3). *Gráfica obtenida del modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,784; Varianza DICE = 0,0048

Id (3.a.2). *Gráfica obtenida del modelo de riñón entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,722; Varianza DICE = 0,0073

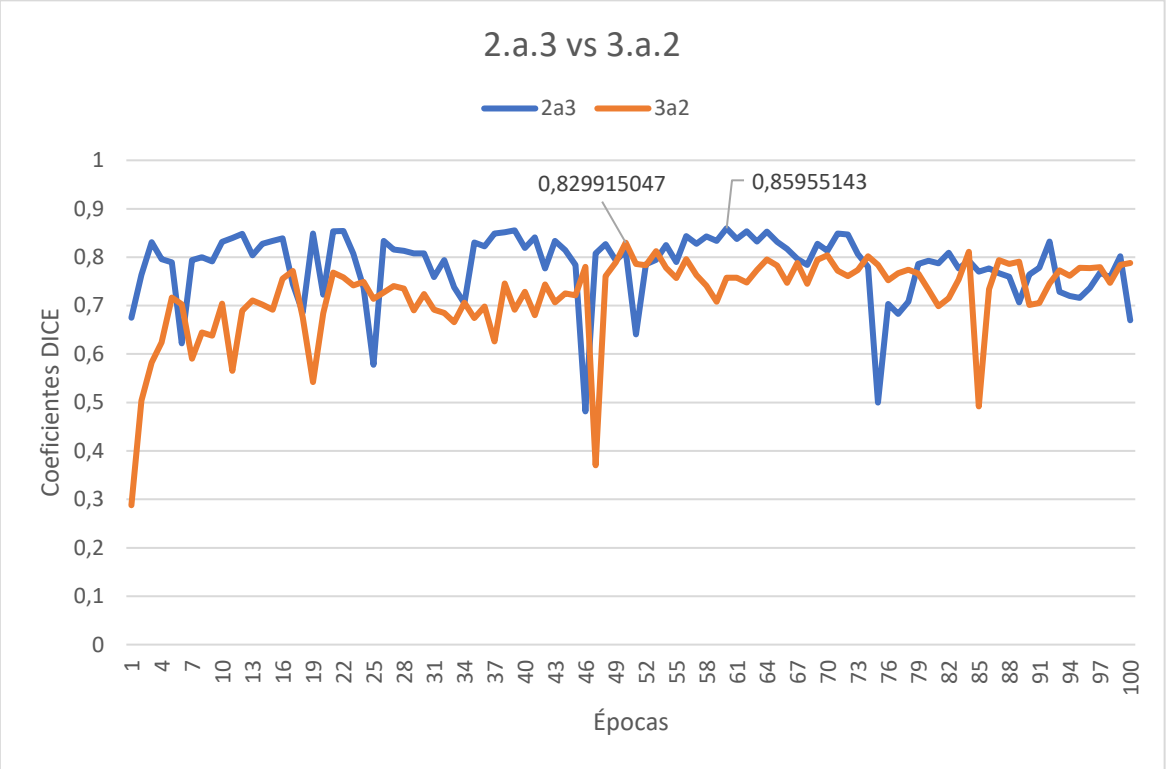

Figura 33. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 2.a.3 y 3.a.2.

Se observa que el modelo 2.a.3 entrenado con secuencia MECSE tiene mejores coeficientes que el modelo 3.a.2 entrenado con secuencia THRIVE (en promedio, 0,784 frente a 0,722, respectivamente). Asimismo, el modelo 2.a.3 tiene menor varianza (0,0048 frente a 0,0073) y mayor DICE máximo (0,859 frente a 0,829). Con esto, se concluye que el modelo 2.a.3 ofrece mejores resultados que el modelo 3.a.2.

En general, se ha observado que, para las secuencias en el modelo de riñón, segmentar sobre secuencias MECSE y THRIVE obtiene resultados un poco superiores a segmentar sobre secuencia MECSE cuando se inicializan aleatoriamente los modelos. En cambio, con pesos iniciales de hígado, segmentar sobre secuencia MECSE obtiene mejores resultados que segmentar sobre secuencias MECSE y THRIVE. Después, se ha observado que segmentar sobre secuencia THRIVE ofrece resultados peores que segmentar sobre secuencia MECSE y que segmentar sobre secuencias MECSE y THRIVE.

Recapitulando los resultados y conclusiones obtenidas hasta el momento:

Los mejores coeficientes se obtienen utilizando 300 épocas*, un batch size de 10, una tasa de aprendizaje de 1,00E-3, segmentando sobre secuencia MECSE e inicializando con pesos de hígado mediante transfer learning. Estos dos últimos hiperparámetros: secuencias y pesos iniciales (para ver el efecto del transfer learning) se siguen comparando en próximos entrenamientos.

**Se habían obtenido dos mejores modelos con resultados muy igualados con 100 épocas, entonces para salir de dudas y tener más datos para analizar, se ha ampliado el entrenamiento a 300 épocas, pero por lo general con 100 es suficiente excepto en casos puntuales.*

5.4. Modelo de bazo entrenado con secuencias MECSE y THRIVE

Ahora se comparan modelos con 100 épocas, batch size de 10 y tasa de aprendizaje de 1,00E-3, solamente cambiando la inicialización de los pesos.

Para analizar el transfer learning, se seleccionará el modelo de riñón con mejores resultados entrenado hasta el momento para introducir sus pesos como pesos iniciales del modelo de bazo.

Para seleccionar los pesos del mejor modelo, se debe seleccionar la época donde el DICE sea más alto, pero también donde haya cierta estabilidad (no haya subidas y bajas bruscas alrededor de esa época). Por ello, el modelo elegido ha sido el 2.a.3 (modelo de riñón entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10) y la época 60 con un DICE de 0,859 rodeado de épocas con coeficientes similares y estables.

<u>1ª comparativa (4.a.1, 4.a.2 y 4.a.3)</u>

Estos tres modelos tienen en común: 100 épocas, entrenados con secuencias MECSE y THRIVE, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 4.a.1 se inicializa con pesos aleatorios y el 4.a.2 con pesos de hígado y el 4.a.3 con pesos de riñón.

Id (4.a.1). *Gráfica obtenida del modelo de bazo entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,534; Varianza DICE = 0,0073

Id (4.a.2). *Gráfica obtenida del modelo de bazo entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,541; Varianza DICE = 0,0042

Id (4.a.3). *Gráfica obtenida del modelo de bazo entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10. Nota: Para los pesos iniciales, se eligen los pesos del modelo de riñones entrenado con secuencia MECSE, 100 épocas, 10 de batch size, 1,00E-3 de tasa de aprendizaje e inicializado con hígado; concretamente, los pesos de la época 60 que posee un Dice de 0.86.*

Media DICE = 0,646; Varianza DICE = 0,0019

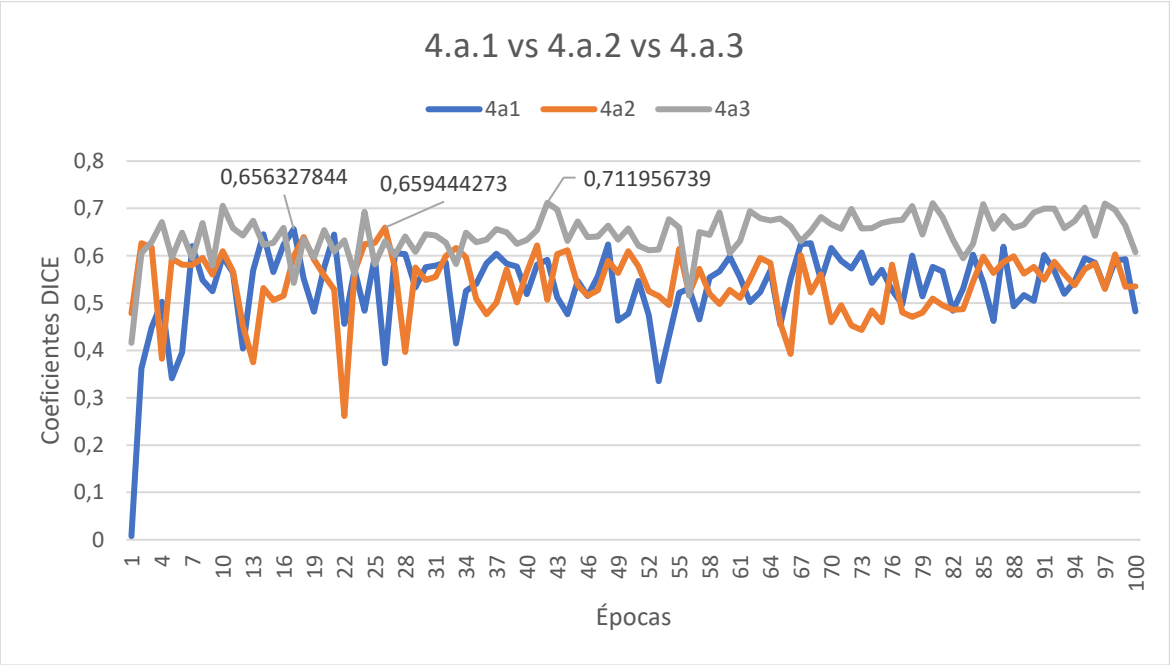

Figura 34. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 4.a.1, 4.a.2 y 4.a.3.

En general, se observan coeficientes menores que en los modelos de riñones para las tres inicializaciones. De todos modos, se analizan las diferentes inicializaciones para ver el efecto del transfer learning.

Se observa que el modelo 4.a.3 inicializado con riñón presenta los mejores coeficientes. El modelo 4.a.3 tiene el mayor promedio (0,646 frente a 0,541 del modelo 4.a.2 inicializado con hígado y frente a 0,534 del modelo 4.a.1 inicializado aleatoriamente). También el modelo 4.a.3 tiene menor varianza (0,0019 frente a 0,0042 del modelo 4.a.2 y 0,0073 del modelo 4.a.1) y mayor valor máximo (0,712 frente a 0,659 del modelo 4.a.2 y 0,656 del modelo 4.a.1). Con estos datos se observa que el mejor modelo es el inicializado con riñón, después el de hígado y por último el aleatorio; aunque entre estos dos últimos la diferencia es muy poca.

5.5. Modelo de bazo entrenado con secuencia MECSE

Ahora se comparan modelos con 100 épocas, batch size de 10 y tasa de aprendizaje de 1,00E-3, solamente cambiando la inicialización de los pesos.

1ª comparativa (5.a.1, 5.a.2 y 5.a.3)

Estos tres modelos tienen en común: 100 épocas, entrenados con secuencia MECSE, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 5.a.1 se inicializa con pesos aleatorios y el 5.a.2 con pesos de hígado y el 5.a.3 con pesos de riñón.

Id (5.a.1). *Gráfica obtenida del modelo de bazo entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,566; Varianza DICE = 0,0152

Id (5.a.2). *Gráfica obtenida del modelo de bazo entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,597; Varianza DICE = 0,0056

Id (5.a.3). *Gráfica obtenida del modelo de bazo entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10. Nota: Para los pesos iniciales, se eligen los pesos del modelo de riñones entrenado con secuencia MECSE, 100 épocas, 10 de batch size, 1,00E-3 de tasa de aprendizaje e inicializado con hígado; concretamente, los pesos de la época 60 que posee un Dice de 0.86.*

Media DICE = 0,64; Varianza DICE = 0,0032

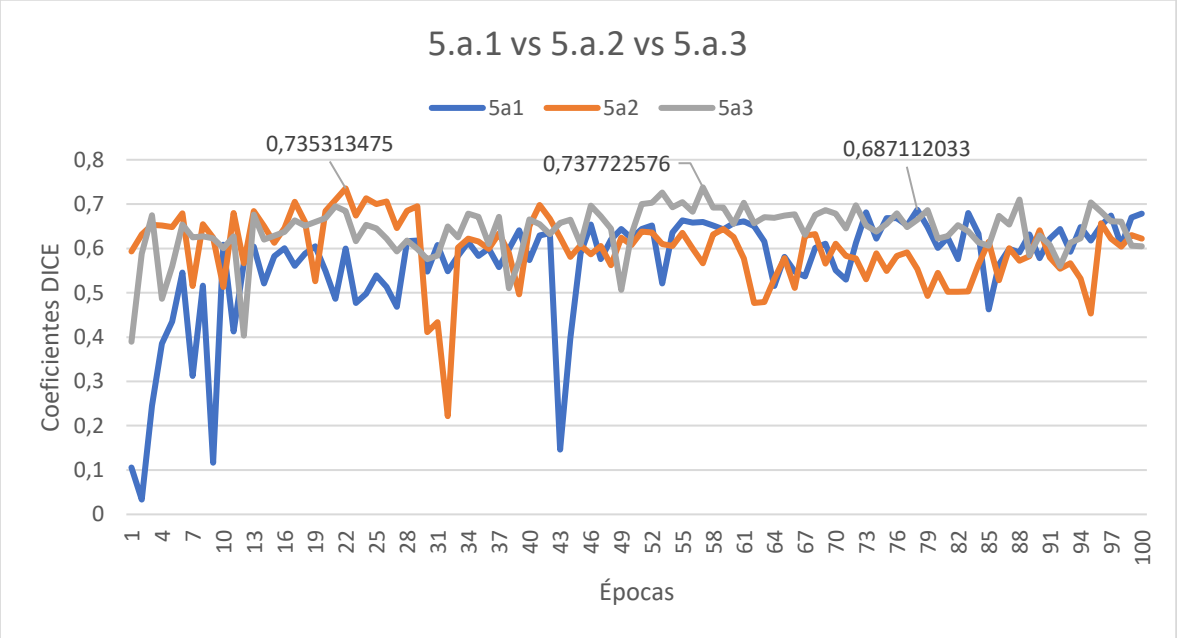

Figura 35. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 5.a.1, 5.a.2 y 5.a.3.

Se observa que el modelo 5.a.3 inicializado con riñón tiene mejores coeficientes en la mayor parte de las épocas, excepto entre las épocas 20 y 30 donde el modelo 5.a.2 inicializado con hígado obtiene mejores resultados y obtiene su máximo (0,735). Aun así, el promedio del modelo 5.a.3 es mayor (0,64 frente a 0,597 del modelo 5.a.2 y 0,566 del modelo 5.a.1). En cuanto a la varianza, el modelo 5.a.3 tiene la menor varianza (0,0032 frente a 0,0056 del modelo 5.a.2 y 0,0152 del modelo 5.a.1). El modelo 5.a.3 tiene un máximo mayor por poco (0,737 frente a 0,735 del modelo 5.a.2 y 0,687 del modelo 5.a.1). En resumen, el modelo 5.a.2 inicializado con hígado está más igualado con el modelo 5.a.3 inicializado con riñón, pero el 5.a.3 sigue dando mejores resultados. Por debajo de estos, se encuentra el modelo 5.a.1 inicializado aleatoriamente con resultados peores.

5.6. Modelo de bazo entrenado con secuencia THRIVE

Ahora se comparan modelos con 100 épocas, batch size de 10 y tasa de aprendizaje de 1,00E-3, solamente cambiando la inicialización de los pesos.

1ª comparativa (6.a.1, 6.a.2 y 6.a.3)

Estos tres modelos tienen en común: 100 épocas, entrenados con secuencia THRIVE, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 6.a.1 se inicializa con pesos aleatorios y el 6.a.2 con pesos de hígado y el 6.a.3 con pesos de riñón.

Id (6.a.1). *Gráfica obtenida del modelo de bazo entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,534; Varianza DICE = 0,0151

Id (6.a.2). *Gráfica obtenida del modelo de bazo entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10.*

Media DICE = 0,504; Varianza DICE = 0,01

Id (6.a.3). *Gráfica obtenida del modelo de bazo entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10. Nota: Para los pesos iniciales, se eligen los pesos del modelo de riñones entrenado con secuencia MECSE, 100 épocas, 10 de batch size, 1,00E-3 de tasa de aprendizaje e inicializado con hígado; concretamente, los pesos de la época 60 que posee un Dice de 0.86.*

Media DICE = 0,527; Varianza DICE = 0,0065

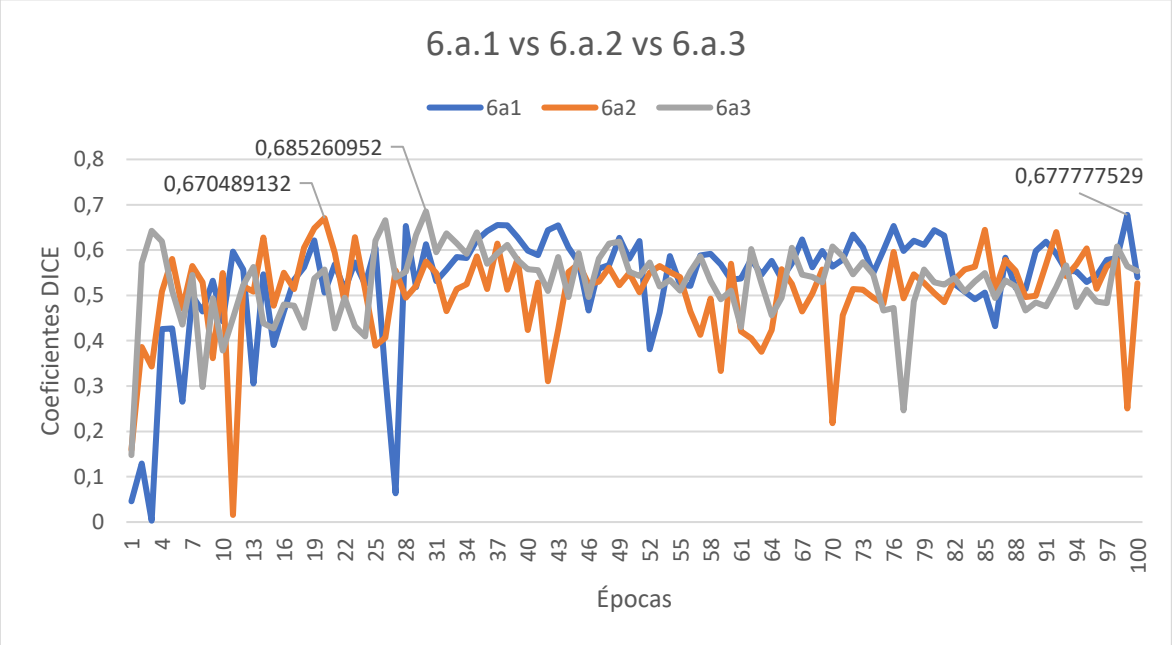

Figura 36. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 6.a.1, 6.a.2 y 6.a.3.

Se observan resultados dispersos. Los tres modelos tienen promedios similares, aunque el del modelo inicializado aleatoriamente es un poco más alto (0,534 frente a 0,527 del modelo 6.a.3 inicializado con riñón y 0,504 del modelo 6.a.2 inicializado con hígado). El valor máximo se obtiene con el modelo 6.a.3 (0,685 frente a 0,678 del modelo 6.a.1 y 0,67 del modelo 6.a.2). El modelo 6.a.3 tiene menor varianza (0,0065 frente a los otros dos modelos que tienen 0,01 aproximadamente). Por tanto, no se puede concluir nada significativo de esta comparativa. A pesar de esto, el modelo 6.a.3 tiene la menor varianza, el mayor máximo y casi el mismo promedio que el promedio más alto, entonces se podría intuir que es el mejor modelo de los tres.

5.6.1. Comparativa de los mejores modelos de bazo para diferentes secuencias

2ª comparativa (4.a.3, 5.a.3 y 6.a.3)

Ahora se comparan los mejores modelos de bazo obtenidos sobre secuencias MECSE y THRIVE, sobre secuencia MECSE solamente y sobre secuencia THRIVE solamente.

Estos tres modelos tienen en común: 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10.

Se diferencian en que el modelo 4.a.3 se segmenta sobre secuencias MECSE y THRIVE, el 5.a.3 sobre secuencia MECSE y el 6.a.3 sobre secuencia THRIVE.

Id (4.a.3). *Gráfica obtenida del modelo de bazo entrenado con secuencias MECSE y THRIVE entrenado con 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10. Nota: Para los pesos iniciales, se eligen los pesos del modelo de riñones entrenado con secuencia MECSE, 100 épocas, 10 de batch size, 1,00E-3 de tasa de aprendizaje e inicializado con hígado; concretamente, los pesos de la época 60 que posee un Dice de 0.86.*

Media DICE = 0,646; Varianza DICE = 0,0019

Id (5.a.3). *Gráfica obtenida del modelo de bazo entrenado con secuencia MECSE entrenado con 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10. Nota: Para los pesos iniciales, se eligen los pesos del modelo de riñones entrenado con secuencia MECSE, 100 épocas, 10 de batch size, 1,00E-3 de tasa de aprendizaje e inicializado con hígado; concretamente, los pesos de la época 60 que posee un Dice de 0.86.*

Media DICE = 0,64; Varianza DICE = 0,0032

Id (6.a.3). *Gráfica obtenida del modelo de bazo entrenado con secuencia THRIVE entrenado con 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10. Nota: Para los pesos iniciales, se eligen los pesos del modelo de riñones entrenado con secuencia MECSE, 100 épocas, 10 de batch size, 1,00E-3 de tasa de aprendizaje e inicializado con hígado; concretamente, los pesos de la época 60 que posee un Dice de 0.86.*

Media DICE = 0,527; Varianza DICE = 0,0065

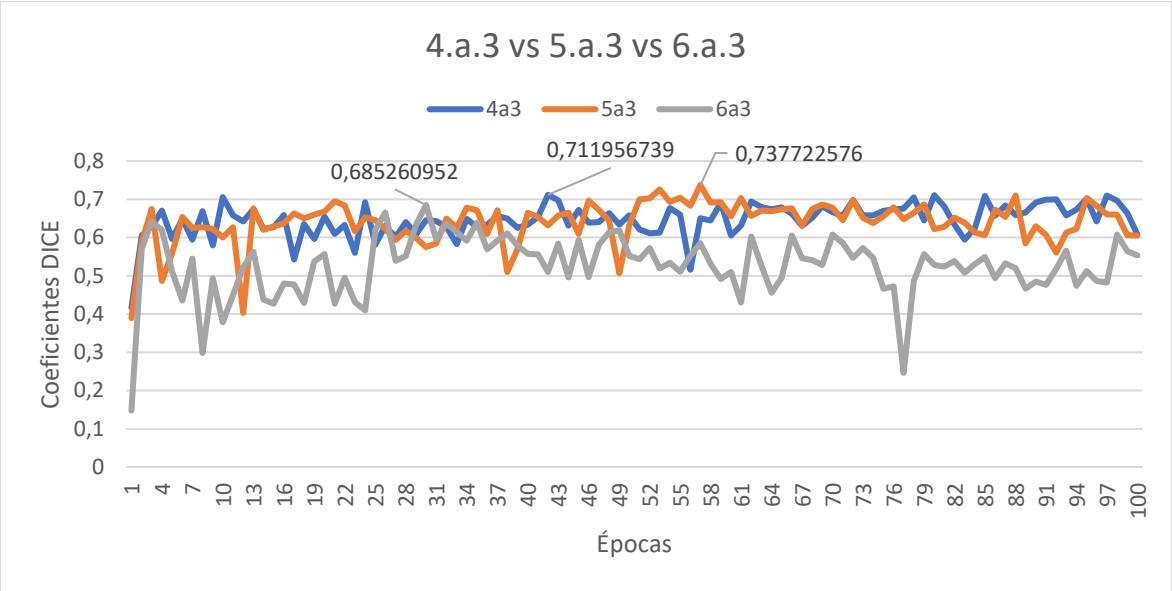

Figura 37. Gráfica comparativa de coeficientes DICE respecto las épocas entre los modelos 4.a.3, 5.a.3 y 6.a.3.

Se observa que el modelo 6.a.3 entrenado con secuencia THRIVE queda por debajo de los otros dos obteniendo peores resultados. En cuanto al modelo 4.a.3 entrenado con secuencias MECSE y THRIVE y el modelo 5.a.3 entrenado con secuencia MECSE, tienen promedios prácticamente iguales (0,646 y 0,640 respectivamente). El modelo 4.a.3 tiene menor varianza que el 5.a.3 por poco (0,0019 frente a 0,0032) pero el modelo 5.a.3 tiene mayor máximo (0,738 frente a 0,712). Con estos resultados es difícil extraer una conclusión estadísticamente significativa.

5.7. Validación de los resultados

Una vez entrenados todos los modelos, se seleccionan los pesos del que mejor métrica tiene y se cargan junto a la arquitectura del modelo para realizar las predicciones. Las predicciones se aplican sobre los datos de test que aún no han sido pasados por ningún modelo.

Se ha creado un Excel que recoge los coeficientes DICE de cada predicción y sobre el cual se han eliminado los valores 1 correspondientes a cuando no existe máscara ni real ni predicha. Después, se han graficado los coeficientes resultantes y se ha indicado el máximo en la propia gráfica, que corresponde a la predicción con mayor coeficiente DICE, es decir, con mayor similitud entre máscara real y predicha. Posteriormente, se muestran para cada modelo, tanto la gráfica de coeficientes de las predicciones como la predicción con mejor coeficiente DICE.

Por otro lado, un factor interesante en las predicciones del modelo de riñón/bazo entrenado con secuencias MECSE y THRIVE es saber cómo está influyendo cada una de las secuencias en el modelo conjunto. Para conocer este dato, se dispone de los NPY's de test de cada y se carga el modelo entrenado con secuencias MECSE y THRIVE (la arquitectura y los mejores pesos obtenidos con el modelo entrenado conjuntamente) para después realizar las predicciones sobre los casos de test de cada secuencia por separado. Se hace primero con la secuencia MECSE y después con la THRIVE; se calcula el promedio de los coeficientes de cada una de las secuencias y para obtener el promedio del modelo entrenado conjuntamente: se suman ambas y se dividen entre dos. Matemáticamente sería así:

$$\frac{Media\ \ DICE\ secuencia\ MECSE + Media\ \ DICE\ secuencia\ THRIVE}{2} = Media\ DICE\ secuencia\ MECSE\ y\ THRIVE \quad (5.7)$$

Esto es así, porque, al diseñar los entrenamientos, los casos de entrenamiento y test se han asignado así, es decir, en los modelos entrenados con secuencias MECSE y THRIVE se ha utilizado para un mismo paciente/caso tanto una secuencia MECSE como una THRIVE (relación 1:1).

5.7.1. Validación modelos de riñón

Para los modelos de riñón se ha seleccionado como mejor modelo, un modelo entrenado con ambas secuencias (para que generalice bien los resultados en todas las secuencias), el 1.b.2 (Modelo de riñón entrenado con secuencias MECSE y THRIVE, 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10) con una media de coeficientes de 0,776.

Concretamente se han seleccionado los pesos de la época 9 que ofrecen el máximo DICE del modelo, 0,862.

5.7.1.1. Modelo de riñón entrenado con secuencias MECSE y THRIVE validado sobre secuencias MECSE y THRIVE.

A continuación, se muestra la gráfica obtenida de las predicciones al utilizar los pesos mencionados anteriormente.

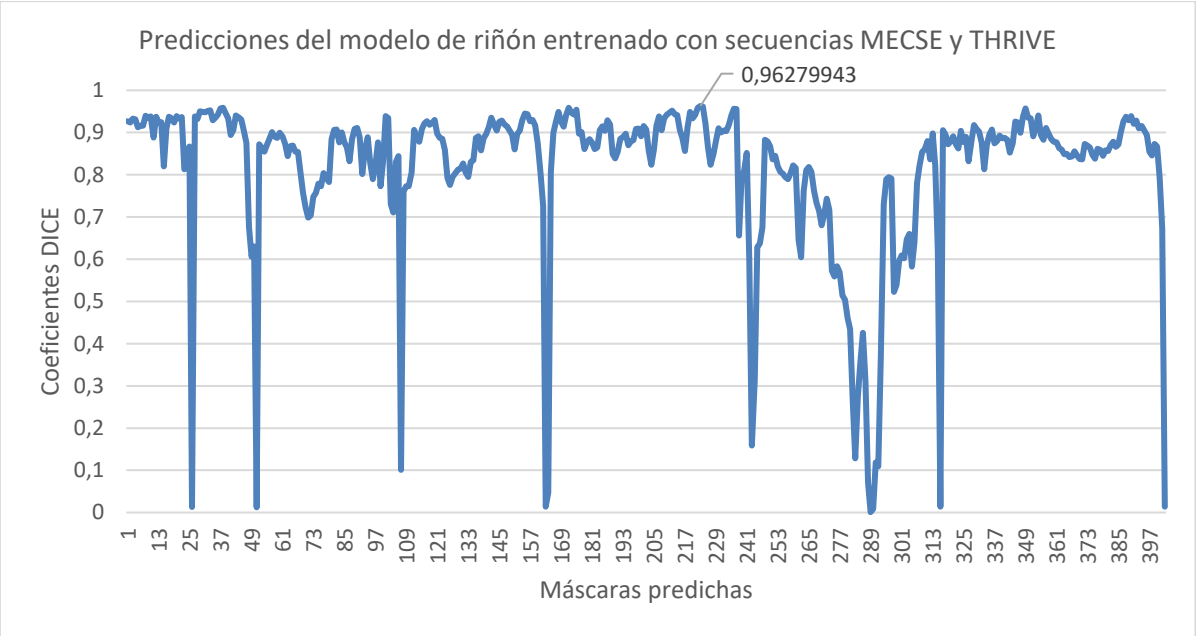

Figura 38. Gráfica de coeficientes DICE de las máscaras predichas con el modelo de riñón entrenado con secuencias MECSE y THRIVE sobre secuencias MECSE y THRIVE.

Los coeficientes DICE de las predicciones tienen un valor promedio de 0,817 y un valor máximo de 0,962.

La mejor predicción obtenida con este modelo es la siguiente:

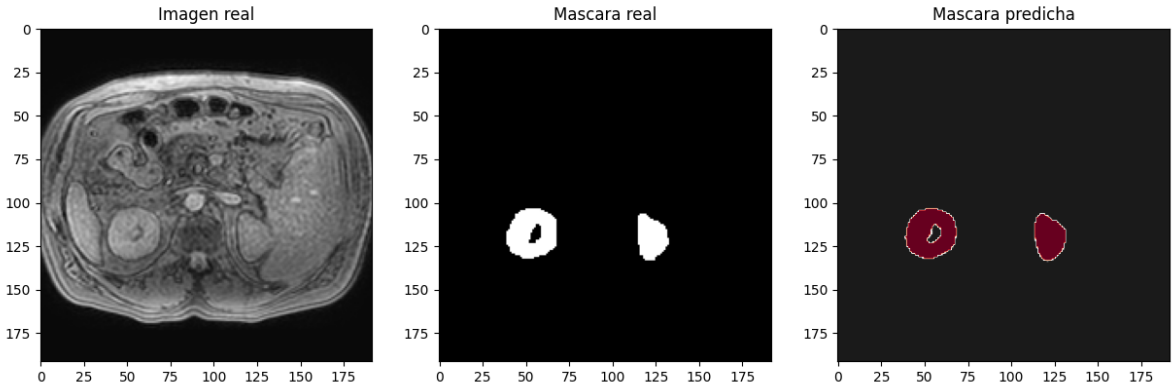

Figura 39. Predicción con mejor coeficiente DICE (0,962) del modelo de riñón entrenado con secuencias MECSE y THRIVE sobre MECSE y THRIVE.

5.7.1.2. Modelo de riñón entrenado con secuencias MECSE y THRIVE validado sobre secuencia MECSE.

Se muestra la gráfica obtenida con las predicciones del modelo de riñón entrenado con secuencia MECSE.

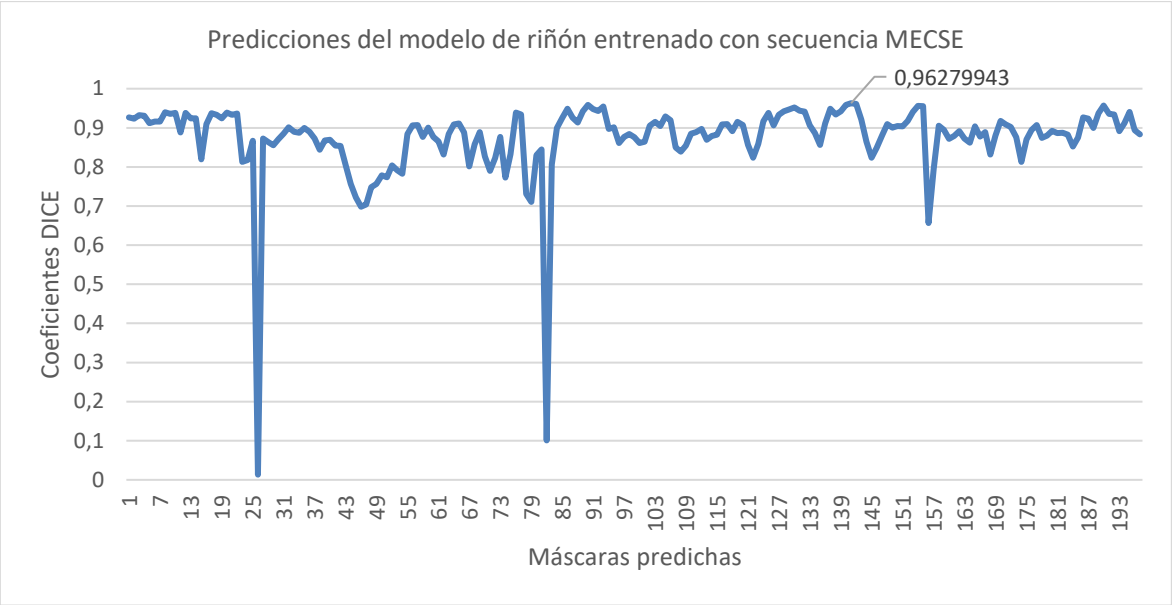

Figura 40. Gráfica de coeficientes DICE de las máscaras predichas con el modelo de riñón entrenado con secuencias MECSE y THRIVE sobre secuencia MECSE.

Se observa que tiene el mismo máximo (0,962) que el modelo entrenado con secuencias MECSE y THRIVE, esto es porque el máximo del modelo mixto proviene de la secuencia MECSE. Los coeficientes DICE de las predicciones tienen un valor promedio de 0,875.

La mejor predicción obtenida con este modelo es la siguiente:

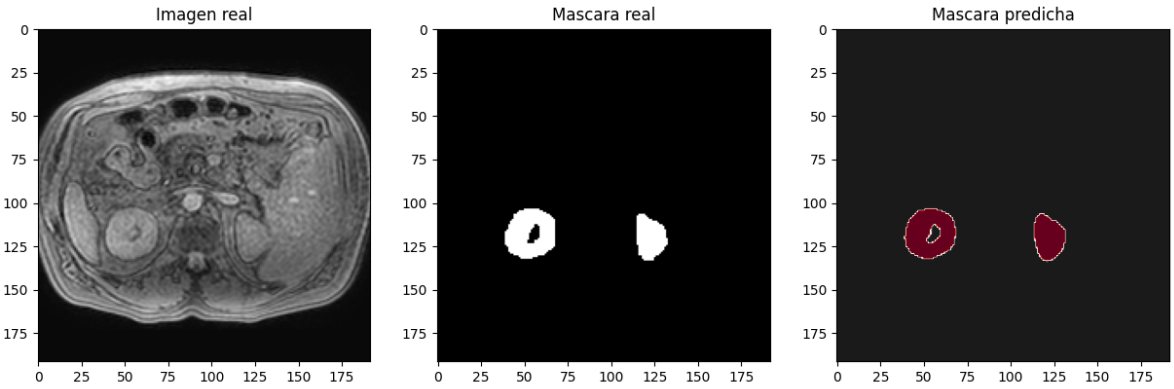

Figura 41. Predicción con el mejor coeficiente DICE (0,962) del modelo de riñón entrenado con secuencias MECSE y THRIVE sobre secuencia MECSE.

Se observa que es la misma predicción que con el modelo entrenado con secuencias MECSE y THRIVE. Esto es por lo explicado anteriormente, de que el modelo conjunto se alimenta de ambas secuencias.

5.7.1.3. Modelo de riñón entrenado con secuencias MECSE y THRIVE validado sobre secuencia THRIVE.

Se muestra la gráfica obtenida con las predicciones del modelo de riñón entrenado con secuencia THRIVE.

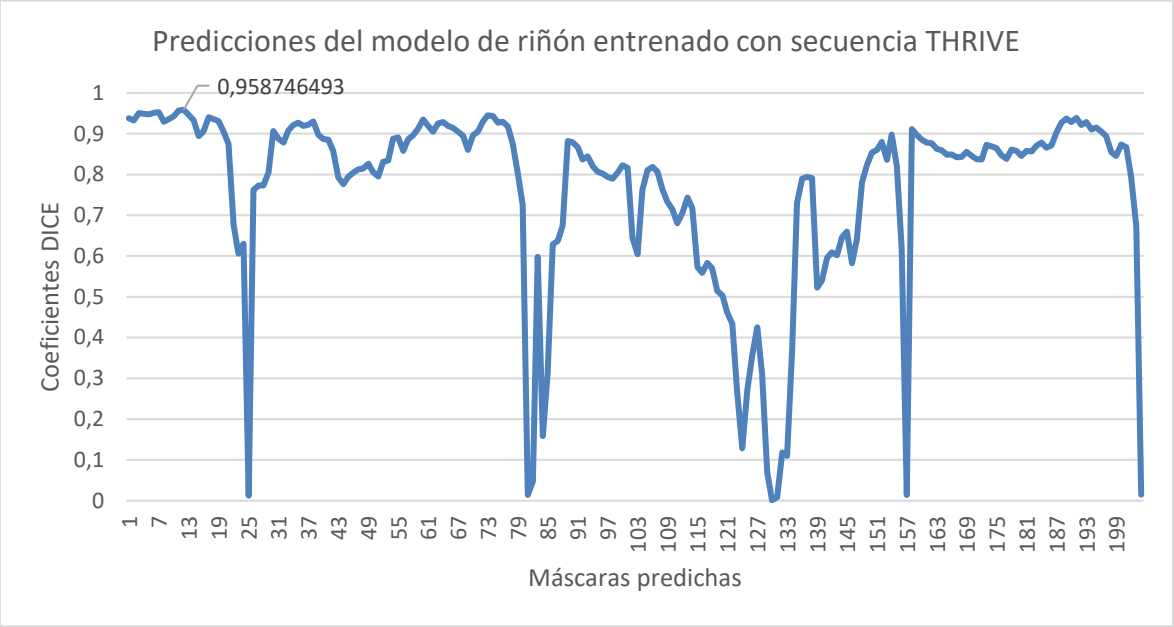

Figura 42. Gráfica de coeficientes DICE de las máscaras predichas con el modelo de riñón entrenado con secuencias MECSE y THRIVE sobre secuencia THRIVE.

Los coeficientes DICE de las predicciones tienen un valor promedio de 0,762 y un valor máximo de 0,958.

La mejor predicción obtenida con este modelo es la siguiente:

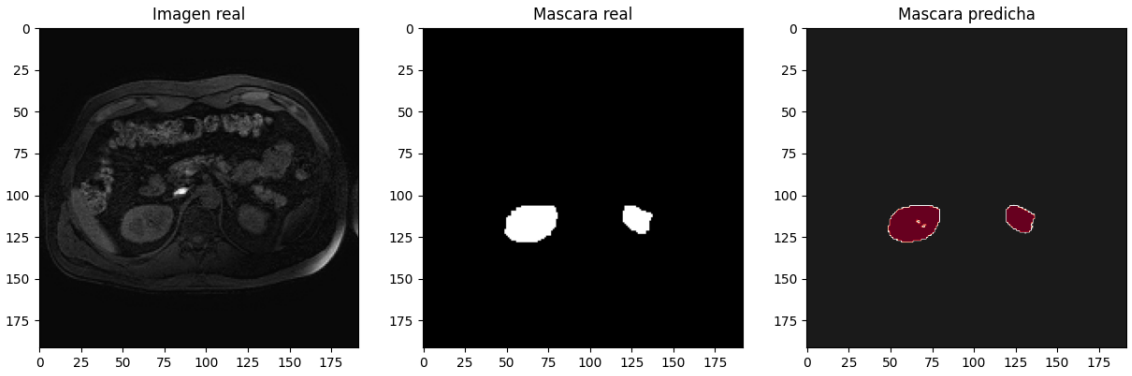

Figura 43. Predicción con el mejor coeficiente DICE (0,958) del modelo de riñón entrenado con secuencias MECSE y THRIVE sobre secuencia THRIVE.

Ahora que ya se tienen los promedios de los coeficientes de las predicciones de ambas secuencias, se comprueba si los datos son correctos. Se utiliza la fórmula (5.7) con una media de DICE para secuencia MECSE de 0,875 y una media de DICE para secuencia THRIVE de 0,762.

$$\frac{0{,}875 + 0{,}762}{2} = 0{,}8185$$

Se observa que da aproximadamente el valor esperado, que es un valor promedio de 0,817 para el modelo entrenado con secuencias MECSE y THRIVE.

5.7.2. Validación modelos de bazo.

Para los modelos de bazo se ha seleccionado como mejor modelo, un modelo entrenado con ambas secuencias (para que generalice bien los resultados en todas las secuencias), el 4.a.3 (Modelo de bazo entrenado con secuencias MECSE y THRIVE, 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10) con una media de coeficientes de 0,646.

Concretamente se han seleccionado los pesos de la época 42 que ofrecen el máximo DICE del modelo, 0,712.

5.7.2.1. Modelo de bazo entrenado con secuencias MECSE y THRIVE validado sobre secuencias MECSE y THRIVE.

Se muestra la gráfica obtenida con las predicciones del modelo de bazo entrenado con secuencias MECSE y THRIVE.

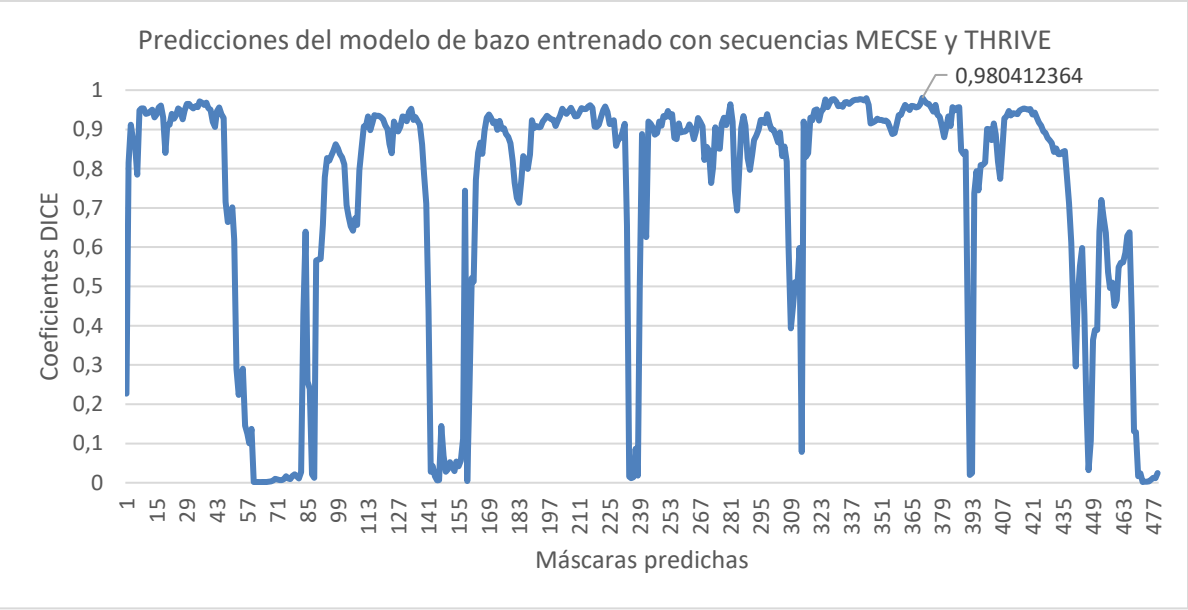

Figura 44. Gráfica de coeficientes DICE de las máscaras predichas con el modelo de bazo entrenado con secuencias MECSE y THRIVE sobre secuencias MECSE y THRIVE.

Los coeficientes DICE de las predicciones tienen un valor promedio de 0,727 y un valor máximo de 0,98.

La mejor predicción obtenida con este modelo es la siguiente:

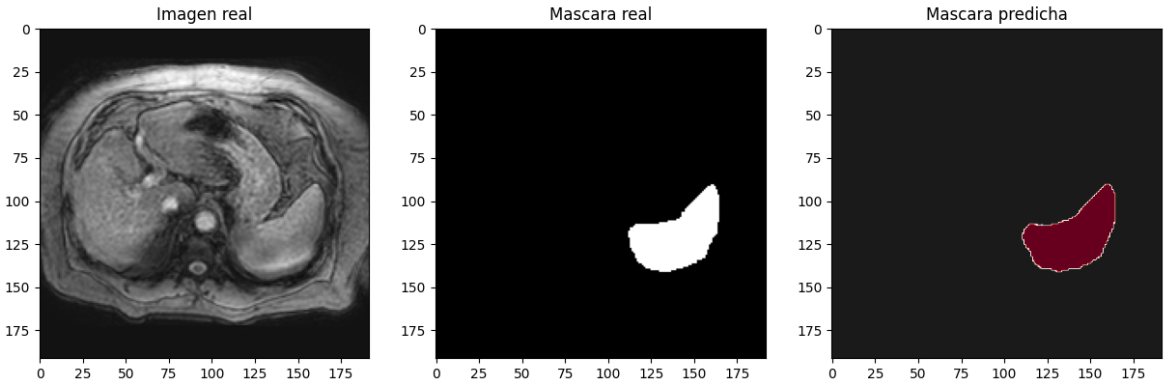

Figura 45. Predicción con el mejor coeficiente DICE (0,98) del modelo de bazo entrenado con secuencias MECSE y THRIVE sobre secuencias MECSE y THRIVE.

5.7.2.1. Modelo de bazo entrenado con secuencias MECSE y THRIVE validado sobre secuencia MECSE.

Se muestra la gráfica obtenida con las predicciones del modelo de bazo entrenado con secuencia MECSE

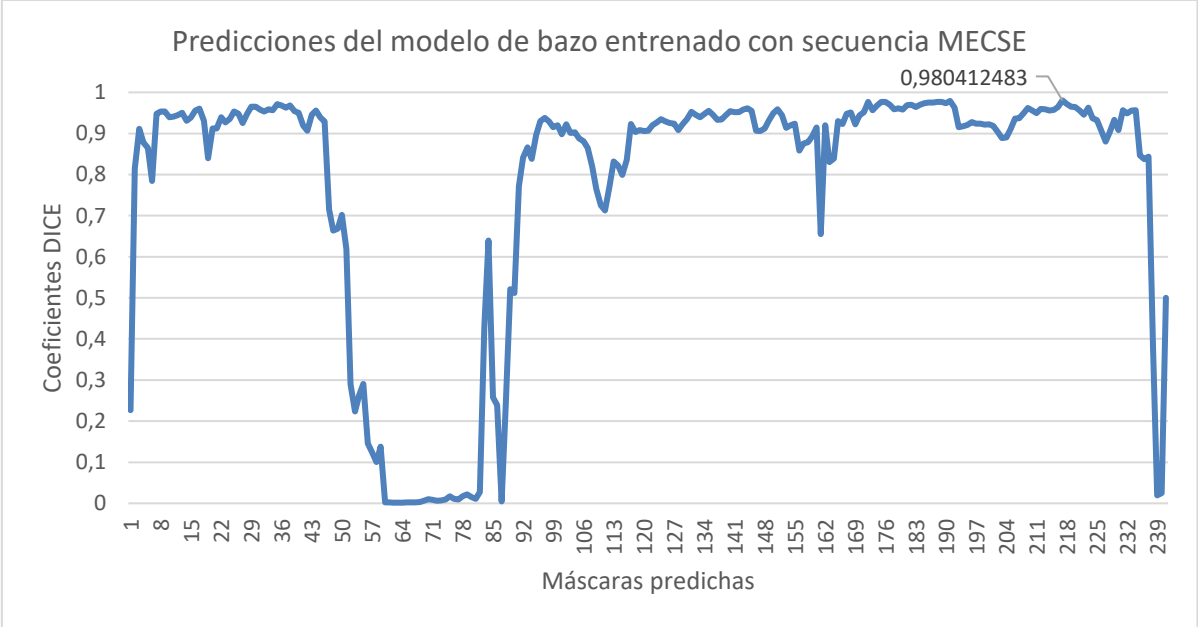

Figura 46. Gráfica de coeficientes DICE de las máscaras predichas con el modelo de bazo entrenado con secuencias MECSE y THRIVE sobre secuencia MECSE.

Los coeficientes DICE de las predicciones tienen un valor promedio de 0,772 y un valor máximo de 0,98, el mismo que en el modelo entrenado con secuencias con MECSE y THRIVE. Ocurre lo mismo que lo explicado en el modelo de riñón.

La mejor predicción obtenida con este modelo es la siguiente:

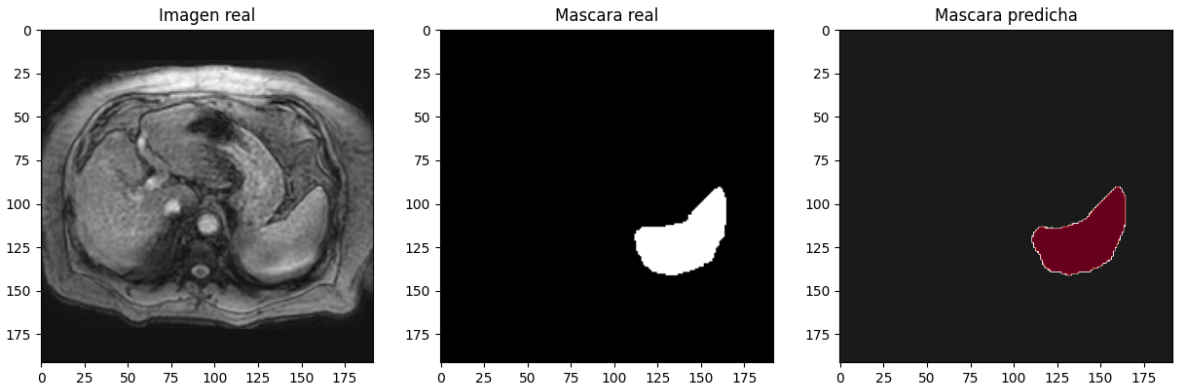

Figura 47. Predicción con el mejor coeficiente DICE (0,98) del modelo de bazo entrenado con secuencias MECSE y THRIVE sobre secuencia MECSE.

Se observa que es la misma predicción que con el modelo entrenado con secuencias MECSE y THRIVE, igual que ocurría en el modelo de riñón.

5.7.2.1. Modelo de bazo entrenado con secuencias MECSE y THRIVE validado sobre secuencia THRIVE.

Se muestra la gráfica obtenida con las predicciones del modelo de bazo entrenado con secuencia THRIVE.

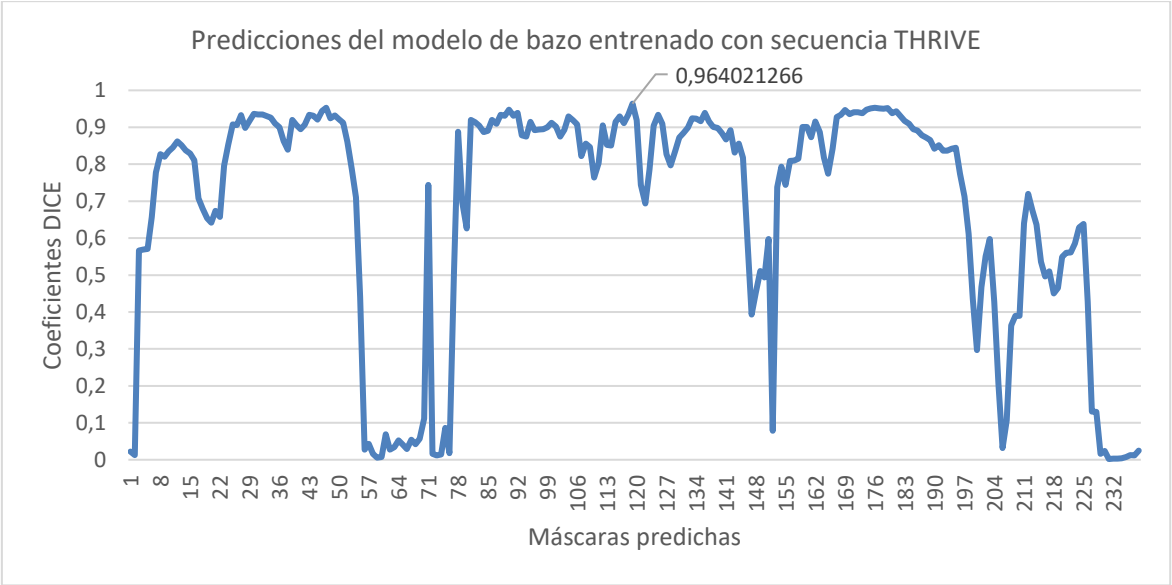

Figura 48. Gráfica de coeficientes DICE de las máscaras predichas con el modelo de bazo entrenado con secuencias MECSE y THRIVE sobre secuencia MECSE.

Los coeficientes DICE de las predicciones tienen un valor promedio de 0,681 y un valor máximo de 0,964.

La mejor predicción obtenida con este modelo es la siguiente:

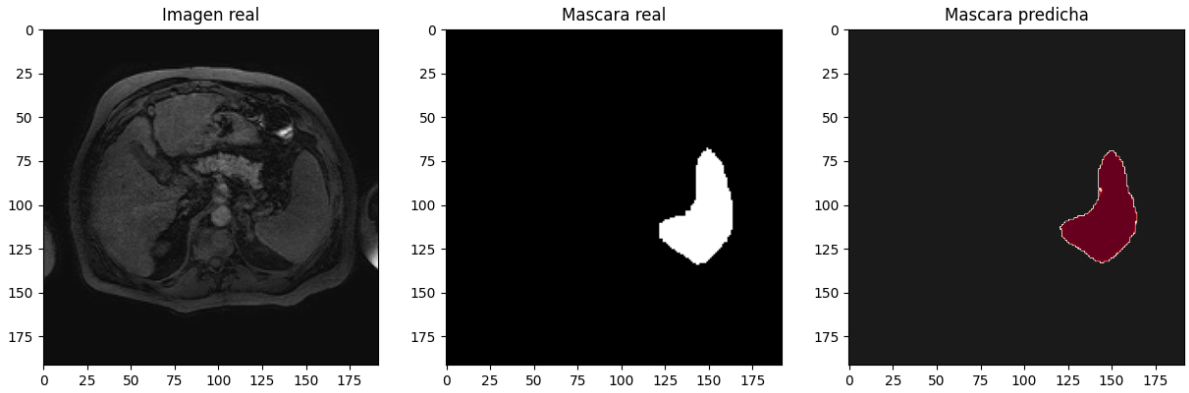

Figura 49. Predicción con el mejor coeficiente DICE (0,964) del modelo de bazo entrenado con secuencias MECSE y THRIVE sobre secuencia THRIVE.

5.8. Discusión de los resultados

En este apartado se discuten los resultados obtenidos, relacionando resultados de los diferentes modelos entrenados para poder sacar las conclusiones correspondientes.

5.8.1 Discusión de los entrenamientos

Se ha observado que cuando se aplica transfer learning se obtienen buenos resultados en épocas más tempranas (10-20 épocas primeras) que si se inicializa un modelo con pesos aleatorios. Esto es útil cuando no se tiene mucha capacidad computacional y no se pueden realizar muchos entrenamientos.

Cuando se entrena el modelo de riñón con MECSE y THRIVE, 100 épocas, batch size de 10 y tasa de aprendizaje 1,00E-3 (mejores hiperparámetros) para ambas inicializaciones (aleatoria y con pesos de hígado) se observa que el máximo DICE (0,879) se obtiene para el modelo inicializado aleatoriamente (1.b.1) en la época 40 y es mayor que el máximo (0,862) del modelo inicializado con pesos de hígado (1.b.2) obtenido en la época 10. Esto se debe a que al aplicar transfer learning, el modelo parte con ventaja ya que tiene ciertas características del modelo pre-entrenado ya aprendidas en sus capas.

El máximo coeficiente DICE y el mayor promedio de coeficientes obtenidos durante todos los entrenamientos son 0,894 y 0,791 respectivamente. Se obtienen con el modelo 2.b.3 (modelo de riñón entrenado con secuencia MECSE, 300 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10). Esto es debido a que el modelo del cual proceden los pesos iniciales de hígado utilizados se entrenó con el dominio MECSE, que es el mismo dominio sobre el cual se ha entrenado este modelo. Es decir, el dominio original y objetivo es el mismo, MECSE, y solo cambia el órgano a segmentar (riñón en lugar de hígado). Además, al entrenarse para 300 épocas permite que el DICE pueda mejorar ligeramente sobre los modelos de 100 épocas.

Por otro lado, cuando se compara el modelo 3.a.2 (modelo de riñón entrenado con secuencia THRIVE, 100 épocas, pesos iniciales de hígado, tasa de aprendizaje de 1,00E-3 y batch size de 10) con el modelo 1.b.1 (modelo de riñón entrenado con secuencias MECSE y THRIVE, 100 épocas, pesos iniciales aleatorios, tasa de aprendizaje de 1,00E-3 y batch size de 10) se observa que obtiene peores resultados (3.a.2 frente 1.b.1: Media = 0,722<0,761; máximo = 0,829<0,879) con pesos

iniciales de hígado respecto al otro con pesos iniciales aleatorios. Esto se explica porque el modelo entrenado con MECSE y THRIVE añade más complejidad/variabilidad, pero también más casos y mayor capacidad de generalización al ser menos dependiente del dominio (combina MECSE+THRIVE). Esto explica la importancia del dominio inicial respecto al final, ya que los pesos de hígado vienen de un modelo entrenado con MECSE, por eso no dan tan buenos resultados en este modelo entrenado con THRIVE.

Por otra parte, si nos fijamos en los modelos de bazo entrenados con MECSE, el modelo 5.a.3 (modelo de bazo entrenado con secuencia MECSE, 100 épocas, pesos iniciales de riñón, tasa de aprendizaje de 1,00E-3 y batch size de 10) inicializado con pesos de riñón es el que mejores resultados ofrece respecto a los modelos con pesos aleatorios y de hígado. Se quiere resaltar este resultado, ya que a priori, se pensaba que se obtendrían mejores resultados con la inicialización de hígado, ya que son morfológicamente más parecidos y no son dos órganos como los riñones.

En cuanto a los modelos de bazo entrenados con THRIVE, el que mejor media obtiene es el modelo inicializado aleatoriamente (6.a.1) debido a que los otros dos han sido inicializados con pesos que provienen de un modelo entrenado con secuencia MECSE, entonces para este modelo funcionan peor.

5.8.2 Discusión de las predicciones

Las predicciones que se han realizado han conseguido obtener unos buenos coeficientes DICE, que se han visto perjudicados por la caída del coeficiente DICE a prácticamente cero o valores muy bajos, en ciertos cortes de las imágenes segmentadas. En estos cortes, la máscara predicha no segmentaba nada, solo un trozo de órgano o incluso un solo órgano en el caso de los dos riñones.

Estos errores en ciertos cortes pueden ser causados por diferentes causas, desde errores de adquisición en las imágenes hasta errores de la propia red de segmentación. La extensión del propio trabajo no permite profundizar en el análisis y resolución de dichos errores.

Capítulo 6
Conclusiones

En el presente trabajo se ha tratado de desarrollar una red neuronal convolucional para segmentación automática y evaluar cómo afectan sus diferentes hiperparámetros al entrenamiento. Asimismo, también se ha analizado la importancia del transfer learning cuando no se dispone de muestras suficientes.

Para ello, se ha realizado una revisión bibliográfica con el objetivo de conocer la arquitectura y el funcionamiento de las redes neuronales y concretamente se ha visto que para segmentación morfológica funcionan muy bien las redes neuronales convolucionales con arquitectura U-Net.

Después, se han realizado segmentaciones manuales sobre los NIFTI's obtenidos en formato DICOM del proyecto previo 2021-595-1 TIAbdoSeg "DESARROLLO DE TÉCNICAS BASADAS EN TRANSFER LEARNING PARA LA SEGMENTACIÓN MÁSIVA MULTIÓRGANO" con el objetivo de crear una serie de muestras que sirvieran de *groundtruth* para la red neuronal de segmentación.

Seguidamente, se ha desarrollado e implementado un modelo *encoder-decoder* basado en la arquitectura U-Net para realizar las segmentaciones de forma automática. En este punto, teniendo en cuenta la literatura científica actual, se han probado diferentes valores para los hiperparámetros, así como diferentes inicializaciones para analizar la utilidad del transfer learning.

Asimismo, se han comparado los resultados obtenidos en los diferentes entrenamientos, y se ha observado que los hiperparámetros que mejor funcionan para la red neuronal propuesta son un *batch size* de 10 y una tasa de aprendizaje de 1,00E-3, para entrenamientos realizados con 100 épocas y, en caso de duda, ampliando a 300.

En los modelos de riñón, el que mejores resultados ha dado ha sido el modelo entrenado con secuencia MECSE, entrenado con estos hiperparámetros mencionados e inicializado con pesos de hígado mediante transfer learning. De hecho, al aplicar transfer learning, se ha observado que se mejoran las métricas de validación, concretamente, el coeficiente DICE, respecto a una inicialización aleatoria. En los modelos de bazo, se ha aplicado transfer learning, a partir de los propios modelos entrenados de riñón, utilizando los mejores pesos obtenidos para inicializar los modelos de bazo. Como resultado, se ha obtenido que la inicialización con riñón ha sido mejor que con pesos aleatorios y de hígado, debido a que se ha entrenado con el mismo modelo, pero con un órgano diferente.

Finalmente, se ha analizado la generalización de la red mediante sus predicciones frente al *groundtruth* para validar los resultados, y se ha visto, que para la mayor parte de las predicciones se obtienen resultados buenos.

Bibliografía

Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*.

Artificial Intelligence A Modern Approach Fourth Edition. (2020).

Ashenden, S. K., Bartosik, A., Agapow, P. M., & Semenova, E. (2021). Introduction to artificial intelligence and machine learning. *The Era of Artificial Intelligence, Machine Learning, and Data Science in the Pharmaceutical Industry*, 15–26. https://doi.org/10.1016/B978-0-12-820045-2.00003-9

Bali, J., Garg, R., & Bali, R. (2019). Artificial intelligence (AI) in healthcare and biomedical research: Why a strong computational/AI bioethics framework is required? *Indian Journal of Ophthalmology*, *67*(1), 3. https://doi.org/10.4103/IJO.IJO_1292_18

Batch normalization: theory and how to use it with Tensorflow | by Federico Peccia | Towards Data Science. (2018). https://towardsdatascience.com/batch-normalization-theory-and-how-to-use-it-with-tensorflow-1892ca0173ad

Blitzer, J., Mcdonald, R., & Pereira, F. (2006). *Domain Adaptation with Structural Correspondence Learning*. 120–128.

Cardenas, C. E., Yang, J., Anderson, B. M., Court, L. E., & Brock, K. B. (2019). Advances in Auto-Segmentation. *Seminars in Radiation Oncology*, *29*(3), 185–197. https://doi.org/10.1016/J.SEMRADONC.2019.02.001

Chandrakumar, T., & Kathirvel, R. (2016). *Classifying Diabetic Retinopathy using Deep Learning Architecture*. www.ijert.org

Complete Guide to Adam Optimization | by Layan Alabdullatef | Towards Data Science. (2020). https://towardsdatascience.com/complete-guide-to-adam-optimization-1e5f29532c3d

Deep Learning with Python FRANÇOIS CHOLLET M A N N I N G SHELTER ISLAND. (2018).

Dice, L. R. (1945). *MEASURES OF THE AMOUNT OF ECOLOGIC ASSOCIATION BETWEEN SPECIES*.

Dropout y Batch Normalization. (2018). https://vincentblog.xyz/posts/dropout-y-batch-normalization

Hubel, D. H., & Wiesel, T. N. (1959). RECEPTIVE FIELDS OF SINGLE NEURONES IN THE CAT'S STRIATE CORTEX. *J. Physiol. (I959)*, *148*, 574–591.

Ioffe, S., & Szegedy, C. (2015). *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*.

ITK-SNAP Home. (2018). http://www.itksnap.org/pmwiki/pmwiki.php

Jimenez-Pastor, A., Alberich-Bayarri, A., Lopez-Gonzalez, R., Marti-Aguado, D., França, M., Bachmann, R. S. M., Mazzucco, J., & Marti-Bonmati, L. (2021). Precise whole liver automatic

segmentation and quantification of PDFF and R2* on MR images. *European Radiology*, *31*(10), 7876–7887. https://doi.org/10.1007/S00330-021-07838-5/FIGURES/6

Jorge Matich, D. (2001). *Cátedra: Informática Aplicada a la Ingeniería de Procesos-Orientación I Redes Neuronales: Conceptos Básicos y Aplicaciones*.

Ker, J., Wang, L., Rao, J., & Lim, T. (2017). Deep Learning Applications in Medical Image Analysis. *IEEE Access*, *6*, 9375–9379. https://doi.org/10.1109/ACCESS.2017.2788044

Kriegeskorte, N., & Golan, T. (2019). Neural network models and deep learning. *Current Biology*, *29*(7), R231–R236. https://doi.org/10.1016/J.CUB.2019.02.034

Lecun et al., 1998. (1998). *Gradient-based learning applied to document recognition*. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=726791&casa_token=Re_5yDg9OF8AAAAA:gb_HTrbzSxXgAl_-b1VoilLZ8g6X4nThmSoiBes0OG2y8alxGQwfN6e_fv3qkmPclKrFBGwt&tag=1

Lecun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. In *Nature* (Vol. 521, Issue 7553, pp. 436–444). Nature Publishing Group. https://doi.org/10.1038/nature14539

Lee, C.-Y., Xie, S., Gallagher, P., Zhang, Z., & Tu, Z. (2014). *Deeply-Supervised Nets*.

Long, M., Cao, Y., Wang, J., Jordan, M. I., & Edu, J. (2015). *Learning Transferable Features with Deep Adaptation Networks*.

Martí-Aguado, D., Alberich-Bayarri, Martín-Rodríguez, J. L., França, M., García-Castro, F., González-Cantero, J., González-Cantero, & Martí-Bonmatí, L. (2020). Differences in multi-echo chemical shift encoded MRI proton density fat fraction estimation based on multifrequency fat peaks selection in non-alcoholic fatty liver disease patients. *Clinical Radiology*, *75*(11), 880.e5-880.e12. https://doi.org/10.1016/J.CRAD.2020.07.031

Maxpooling vs minpooling vs average pooling | by Madhushree Basavarajaiah | Medium. (2019). https://medium.com/@bdhuma/which-pooling-method-is-better-maxpooling-vs-minpooling-vs-average-pooling-95fb03f45a9

Mccarthy, J. (2007). *WHAT IS ARTIFICIAL INTELLIGENCE?* http://www-formal.stanford.edu/jmc/

Medical Imaging Interaction Toolkit: Main Page. (2022). https://docs.mitk.org/2022.04/

Mo, K., Li, S., Zhang, Y., Li, J., & Yang, Q. (2016). *Personalizing a Dialogue System with Transfer Learning*. www.aaai.org

MRI - T1W High Resolution Isotropic Volume Examination - MR-TIP: Database. (2018). https://www.mr-tip.com/serv1.php?type=db1&dbs=T1W%20High%20Resolution%20Isotropic%20Volume%20Examination

Nielsen. (2015). *Neural Networks and Deep Learning*. http://neuralnetworksanddeeplearning.com

Ravi et al., 2017. (2017). *Deep learning for health informatics*. https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=7801947

Redes Neuronales Convoluciones — Introducción al Aprendizaje Automático. (2021). https://dcain.etsin.upm.es/~carlos/bookAA/05.7_RRNN_Convoluciones_CIFAR_10_INFORMATIVO.html

Ronneberger, O., Fischer, P., & Brox, T. (n.d.). *U-Net: Convolutional Networks for Biomedical Image Segmentation*. Retrieved August 15, 2022, from http://lmb.informatik.uni-freiburg.de/

Rosenblatt, F. (1958). THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN 1. *Psychological Review*, *65*(6), 19–27.

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature 1986 323:6088*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Sankar, M., Batri, K., & Parvathi, R. (2016). EARLIEST DIABETIC RETINOPATHY CLASSIFICATION USING DEEP CONVOLUTION NEURAL NETWORKS. In *International Journal of Advanced Engineering Technology E-ISSN*.

Segmentación de Imágenes con Redes Convolucionales. (2022). https://www.iartificial.net/segmentacion-imagenes-redes-convolucionales/

Sharma, S., Sharma, S., & Athaiya, A. (2020a). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, *4*, 310–316. http://www.ijeast.com

Sharma, S., Sharma, S., & Athaiya, A. (2020b). ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*, *4*, 310–316. http://www.ijeast.com

Skrifter, B., ind, B. v, Thorvald, B. S., & København, R. (1948). *DET KONGELIGE DANSKE VIDENSKABERNES SELSKAB A METHOD OF E ST A B L ISH IN G GROUPS OF EQUAL AMPLITUDE IN PLANT SOCIOLOGY B A SE D ON SIM ILARITY OF S P E C IE S CONTENT AND ITS APPLICATION TO ANALYSES OF THE VEGETATION ON DANISH COMMONS*.

Tarificación oficial Plataforma de Radiología Experimental y Biomarcadores de Imagen SERVICIOS OFRECIDOS CÓDIGO DESCRIPCIÓN DEL SERVICIO INTERNOS 1 OPIs* 2 EMPRESAS* 3 RESONANCIA MAGNÉTICA Y SERVICIOS ASOCIADOS*. (2019). www.iislafe.es

Torrey, L., & Shavlik, J. (n.d.). *Transfer Learning*.

Torrey, L., & Shavlik, J. (2009). Transfer learning. *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, 242–264. https://doi.org/10.4018/978-1-60566-766-9.CH011

Transfer Learning: A Beginner's Guide | DataCamp. (2022). https://www.datacamp.com/tutorial/transfer-learning

Wang, H., Yu, D., Chen, P., & Wei, Z. (2016). *Mixed Pooling for Convolutional Neural Networks Video content analysis View project Mixed Pooling for Convolutional Neural Networks*. https://doi.org/10.1007/978-3-319-11740-9_34

Wei, Y., Zheng, Y., & Yang, Q. (2016). *Transfer Knowledge between Cities*. https://doi.org/10.1145/2939672.2939830

What is Artificial Intelligence (AI)? | IBM. (2020). https://www.ibm.com/cloud/learn/what-is-artificial-intelligence

What is Deep Learning? | IBM. (2020). https://www.ibm.com/cloud/learn/deep-learning

Zhou, L. Q., Wang, J. Y., Yu, S. Y., Wu, G. G., Wei, Q., Deng, Y. bin, Wu, X. L., Cui, X. W., & Dietrich, C. F. (2019). Artificial intelligence in medical imaging of the liver. *World Journal of Gastroenterology*, *25*(6), 672. https://doi.org/10.3748/WJG.V25.I6.672

Parte II

Presupuesto

Capítulo 7

Presupuesto

En este capítulo se aporta una valoración económica de forma estructurada del Trabajo de Fin de Grado. Primero, se describen los costes de mano de obra, maquinaria y materiales. Finalmente, se desarrolla el presupuesto total teniendo en cuenta el presupuesto de ejecución de material y el presupuesto de ejecución por contrata.

7.1. Coste de mano de obra

En esta sección se tienen en cuenta los costes de mano de obra o recursos humanos, en función del tiempo que ha sido dedicado por cada persona participante. Concretamente, ha participado: Ignacio Bosch Roig, Dr. ingeniero de telecomunicaciones como tutor; Amadeo Ten Esteve, ingeniero de telecomunicaciones especializado en electrónica y biomedicina como cotutor; José Ramón Aviñó Nouselles estudiante de ingeniería biomédica como autor; Diana Veiga Canuto, radióloga como realizadora de las pruebas de imagen y revisora de las segmentaciones manuales realizadas.

Tabla 7.1. Cuadro de precios de mano de obra.

Denominación	**Uds.**	**Cantidad**	**Precio Unitario (€)**	**Total (€)**
Tutor	h	15	20	300
Cotutor	h	60	18	1080
Autor	h	350	12	4200
Radióloga	h	25	25	625
			Total	**6205**

7.2. Coste de maquinaria y materiales

En esta sección se contabilizan los costes asociados al software y al hardware necesarios para la realización del proyecto.

En cuanto a software, para la segmentación manual se han utilizado ITK-SNAP y MITK. Se ha utilizado PyCharm 2021.3.3 como entorno de programación, Python 3.8 como leguaje y se ha hecho uso de la librería Keras sobre el framework TensorFlow. Asimismo, las gráficas se han realizado con Microsoft Excel y la redacción con Micrsoft Word.

Tabla 7.2. Cuadro de precios de software.

Denominación	Uds.	Cantidad	Precio Unitario (€)	Periodo de amortización (años)	Intervalo amortizado (meses)	Total (€)
Paquete Microsoft Office	u	1	80	1	5	33,33
Python 3.8	u	1	0	1	5	0
PyCharm	u	1	0	1	5	0
Keras y TensorFlow	u	1	0	1	5	0
Microsoft Windows 11	u	1	32	1	5	13,33
					Total	**46,66**

En cuanto al hardware, se ha empleado un ordenador Victus by HP Laptop 11th Gen Intel® Core™ i7 11800H 8 núcleos 2.3 GHz 16 GB de RAM con tarjeta gráfica NVIDIA GeForce RTX 3050 Laptop GPU. Además, para realizar las adquisiciones de las imágenes se ha utilizado una Resonancia Magnética (RM) de 3 Tesla (T).

Tabla 7.3. Cuadro de precios de hardware.

Denominación	Uds.	Cantidad	Precio Unitario (€)	Periodo de amortización (años)	Intervalo amortizado (meses)	Total (€)
Victus by HP Laptop 11th Gen Intel® Core™ i7	u	1	1150	4	8	191,66
*RM de Alta Complejidad	sujetos	16	261	-	-	4176
					Total	**4367,66**

*No se amortiza el coste debido a que este precio corresponde a la Tarifación oficial de la Plataforma de Radiología Experimental y Biomarcadores de Imagen (*Tarificación Oficial Plataforma de Radiología Experimental y Biomarcadores de Imagen SERVICIOS OFRECIDOS CÓDIGO DESCRIPCIÓN DEL SERVICIO INTERNOS* 1 OPIs* 2 EMPRESAS* 3 RESONANCIA MAGNÉTICA Y SERVICIOS ASOCIADOS*, 2019).

7.3. Presupuesto total

A los cuadros de precios anteriores, se debe añadir el porcentaje de gastos generales (13%) y el asociado al beneficio industrial (6%). Finalmente, al precio total bruto se debe añadir el impuesto del IVA (21%). Aplicando todo esto en la siguiente tabla, se obtiene el presupuesto total del trabajo.

Tabla 7.4. Presupuesto total.

Capítulos		**Importe (€)**
Coste mano de obra Coste maquinaria y materiales		6205 4414,32
Presupuesto de ejecución de material		10619,32
13% de gastos generales 6% de beneficio industrial		1380,51 637,16
Presupuesto de ejecución por contrata		12636,99
21% de IVA		2653,77
PRESUPUESTO TOTAL		**15290,76**