



UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Adaptación de dominio no supervisada para segmentación de resonancias magnéticas cerebrales mediante redes adversarias de consistencia cíclica

Tesis de Licenciatura en Ciencias de la Computación

Julián Alberto Palladino

Director: Dr. Enzo Ferrante

Codirector: Dr. Diego Fernández Slezak

Buenos Aires, 2019

ADAPTACIÓN DE DOMINIO NO SUPERVISADA PARA SEGMENTACIÓN DE RESONANCIAS MAGNÉTICAS CEREBRALES MEDIANTE REDES ADVERSARIAS DE CONSISTENCIA CÍCLICA

La segmentación automática es uno de los problemas más importantes en el campo de visión por computadora aplicado al análisis de imágenes médicas. En particular, la segmentación de estructuras anatómicas y patológicas en imágenes de resonancia magnética cerebrales (MRI, por sus siglas en inglés) es una tarea fundamental en el ámbito de neuroimágenes (por ejemplo, para el análisis morfométrico de cerebro o el planeamiento de radioterapia).

Las redes neuronales convolucionales (CNN) específicamente diseñadas para segmentación de imágenes biomédicas (tales como U-Net o DeepMedic) han superado toda técnica previa en esta labor. Sin embargo, son extremadamente dependientes de los datos y sólo mantienen un buen rendimiento cuando no hay cambios entre las distribuciones de datos de entrenamiento y de evaluación. Este escenario ocurre frecuentemente en la práctica, ya que variar la marca, el modelo o los parámetros de adquisición del equipo de resonancia magnética utilizado tiene gran impacto en la distribución de las MRI que genera. En otras palabras, dichas variciones generan un cambio en el “dominio” del que provienen las imágenes. Para poder hacer frente a este problema y reutilizar modelos que han sido entrenados en dominios diferentes, se utilizan técnicas de “adaptación de dominio”.

En este trabajo se propone una estrategia no supervisada de adaptación de dominio basada en redes neuronales adversarias de consistencia cíclica (CycleGAN), cuyo objetivo es aprender una función “traductora” que transforme eficazmente imágenes de resonancia magnética volumétricas a través de distintos dominios. Para ello, se implementaron modelos de CycleGAN en 2D y 3D, y se verificó que ambos permiten reducir la divergencia de Jensen-Shannon entre dominios de MRI, lo cual posibilita una segmentación automática con modelos de CNN sobre dominios donde no se tienen datos etiquetados.

Palabras claves: Adaptación de dominio no supervisado, CycleGAN, segmentación de imágenes biomédicas, aprendizaje profundo.

UNSUPERVISED DOMAIN ADAPTATION FOR BRAIN MR IMAGE SEGMENTATION THROUGH CYCLE-CONSISTENT ADVERSARIAL NETWORKS

Image segmentation is one of the pilar problems in the fields of computer vision and medical imaging. Segmentation of anatomical and pathological structures in magnetic resonance images (MRI) of the brain is a fundamental task for neuroimaging (e.g brain morphometric analysis or radiotherapy planning).

Convolutional Neural Networks (CNN) specifically tailored for biomedical image segmentation (like U-Net or DeepMedic) have outperformed all previous techniques in this task. However, they are extremely data-dependent, and maintain a good performance only when data distribution between training and test datasets remains unchanged. When such distribution changes but we still aim at performing the same task, we incur in a domain adaptation problem (e.g. using a different MR machine or different acquisition parameters for training and test data).

In this work, we developed an unsupervised domain adaptation strategy based on cycle-consistent adversarial networks (CycleGAN). We aim at learning a mapping function to transform volumetric MR images between domains (which are characterized by different medical centers and MR machines with varying brand, model and configuration parameters). This technique allows us to reduce the Jensen-Shannon divergence between MR domains, enabling automatic segmentation with CNN models on domains where no labeled data was available.

Keywords: Unsupervised domain adaptation, CycleGAN, biomedical image segmentation, deep learning.

A mi familia.

Índice general

1.. Introducción	1
1.1. La resonancia magnética y su segmentación automática	1
1.2. Motivación del presente trabajo	1
1.3. Objetivo	3
1.4. Organización del trabajo	3
2.. Segmentación automática de imágenes médicas	5
2.1. Imágenes por resonancia magnética	5
2.2. Métodos clásicos de segmentación	7
2.2.1. Umbralado	7
2.2.2. Métodos orientados a la detección de bordes	7
2.2.3. Crecimiento de Regiones	8
2.2.4. Watershed	8
2.3. Redes neuronales artificiales	8
2.3.1. Perceptrón simple	9
2.3.2. Perceptrón multicapa	10
2.3.3. Descenso por el gradiente	11
2.4. Redes neuronales convolucionales	13
2.4.1. Capa de Pooling	15
2.4.2. Redes neuronales puramente convolucionales	16
2.5. Redes neuronales para la segmentación de imágenes	16
2.5.1. Capa de convolución transpuesta	17
2.5.2. Capa de dropout	17
2.5.3. Batch Normalization & Instance Normalization	18
2.6. Arquitectura U-Net 3D para segmentación de imágenes	19
3.. Adaptación de dominio mediante CycleGAN	21
3.1. Introducción	21
3.2. Adaptación de dominio supervisada	21
3.3. Adaptación de dominio no supervisada	22
3.3.1. Traducción de imágenes con datos pareados	22
3.3.2. Traducción de imágenes sin datos pareados	23
3.4. Redes adversarias generativas (GAN)	23
3.5. Redes adversarias de consistencia cíclica (Cycle-GAN)	25
3.5.1. Función de pérdida adversaria	26
3.5.2. Función de pérdida de consistencia cíclica	26
3.5.3. Función de pérdida de mapeo de identidad	28
3.6. Arquitectura de CycleGAN	29
3.6.1. Arquitectura de la red generadora	29
3.6.2. Arquitectura de la red discriminadora	30

4.. Metodología de trabajo para la experimentación	31
4.1. Librerías y hardware utilizado	31
4.2. Características del dataset	31
4.3. Implementaciones utilizadas de CycleGAN	32
4.3.1. Implementación 2D de CycleGAN	32
4.3.2. Implementación 3D de CycleGAN	34
4.3.3. Implementación de la capa de convolución transpuesta	34
4.4. Red segmentadora y su muestreo de datos	35
4.5. Validación cruzada y división de datos	36
4.6. Normalización	37
4.7. Data augmentation	37
4.8. Métricas	38
4.8.1. Coeficiente de Dice	38
4.8.2. Divergencia de Jensen-Shannon	38
5.. Resultados de la experimentación	41
5.1. Análisis por divergencia de Jensen-Shannon	42
5.1.1. Eficacia de F transformando el dominio Singapur	43
5.1.2. Eficacia de G transformando el dominio Utrecht	44
5.2. Análisis por coeficiente de Dice	45
5.2.1. Eficacia de F transformando el dominio Singapur	46
5.2.2. Eficacia de G transformando el dominio Utrecht	48
6.. Conclusiones y trabajo futuro	51
7.. Artículos científicos generados	53

1. INTRODUCCIÓN

Las imágenes por resonancia magnética (llamadas también MRI, por sus siglas en inglés) son consideradas como una de las mejores técnicas no invasivas para obtener una representación visual de la estructura y composición de diversas formaciones celulares y anatómicas del interior del cuerpo humano. Esta técnica se basa en la emisión de una serie de señales de radiofrecuencia ante las cuales se obtiene una respuesta diferente según el tipo de tejido. Todas las reacciones se capturan, y de esta manera se producen imágenes que diferencian detalladamente las distintas estructuras anatómicas. Existen múltiples tipos de resonancias magnéticas, destacándose las imágenes T1, T2, FLAIR, DWI, entre otras.

Debido a su efectividad y al bajo nivel de riesgo que implica su uso, las MRI son frecuentemente utilizadas en el análisis, diagnóstico y seguimiento de patologías ubicadas en diversos órganos y tejidos, aún en aquellos de más difícil acceso. En particular, son muy convenientes para realizar estudios cerebrales, ya que permiten conformar imágenes tridimensionales que se correspondan con el interior del cráneo presentando gran nivel de detalle y sin el uso de radiación.

Históricamente, la interpretación de imágenes médicas ha dependido tanto de factores objetivos (como la tecnología utilizada) como de los subjetivos del profesional tratante (tales como experiencia previa, cansancio, etc). Durante los últimos años se han desarrollado métodos computacionales para el análisis de dichas imágenes, con el fin de reducir el impacto de los factores subjetivos y mejorar su interpretación por medio de técnicas automatizadas. Los métodos desarrollados en esta tesis de licenciatura pretenden contribuir en esta dirección.

1.1. La resonancia magnética y su segmentación automática

La segmentación automática de imágenes médicas tiene como objetivo asistir a los profesionales de la salud en la interpretación de dichas imágenes. Consiste en la división de una imagen médica en “segmentos”, marcando cuáles píxeles corresponden a las distintas partes anatómicas o patológicas de la imagen (ejemplificado en la Figura 1.1).

Se realiza mediante una gran variedad de métodos de visión por computadora, desde la detección de bordes (nacida hace medio siglo con el pionero trabajo de Roberts [1]) hasta el aprendizaje automático (en auge actualmente).

Una de las ramas de aprendizaje automático que demuestra mayor potencial en el estado del arte son las redes neuronales, que tienen su origen en la génesis de la computación [3], y que hoy están desplegando su total potencial gracias a la creciente capacidad de cómputo en paralelo con el uso de unidades específicas de procesamiento gráfico (GPUs) y al mayor volumen de datos disponibles, los cuales juegan un rol clave en su entrenamiento.

1.2. Motivación del presente trabajo

Los métodos basados en aprendizaje automático para la segmentación de imágenes médicas en general, y MRI en particular, suelen seguir un paradigma supervisado, en el cual un conjunto de imágenes manualmente segmentadas píxel a píxel por un experto es

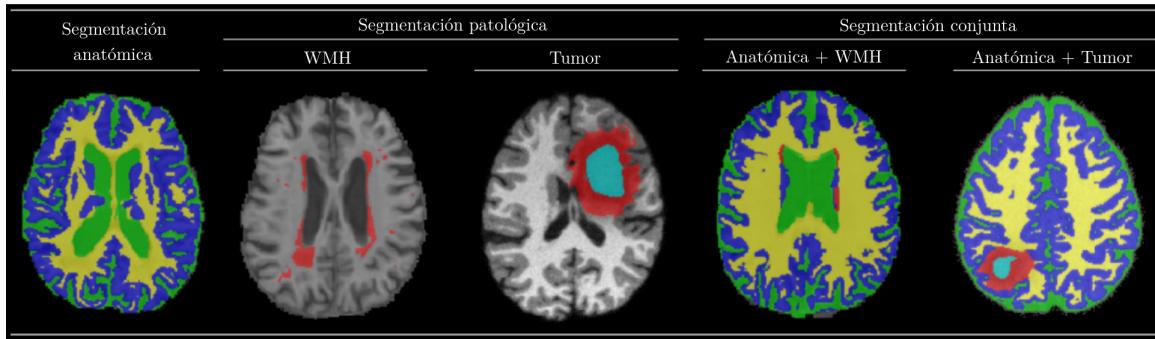


Fig. 1.1: Ejemplos de tipos de segmentaciones: anatómicas, patológicas y conjuntas [2].

utilizado para entrenar dicho modelo. Al entrenar una red neuronal, los datos de entrenamiento tienen igual (o incluso mayor) importancia que la arquitectura elegida [4], tanto en calidad como en cantidad. Esto conlleva una de las dificultades más grandes en la práctica: disponer de un conjunto de datos confiable y abundante del dominio sobre el que se va a utilizar la red, lo cual resulta muy difícil de lograr.

Por un lado, construir un dataset propio es una tarea ardua que requiere enorme esfuerzo manual. Por el otro, existen varios datasets públicos, que quizás sean numerosos y de calidad, pero muy probablemente tengan diferencias significativas con el dominio sobre el que se quiere trabajar. Esto se debe a una gran cantidad de factores, tales como los cientos de parámetros que puede tener la máquina que toma las imágenes, la marca y modelo de la misma, la calibración realizada, el tipo de MRI producida, el estilo de los profesionales al etiquetar los datos, etcétera. De esta manera, al entrenarse utilizando un dataset de cierto dominio, los modelos suelen funcionar muy bien para nuevos datos del mismo origen, pero si se emplea con un conjunto de imágenes obtenidas en otro dominio, su desempeño disminuye [5][6].

Una posible solución a este problema son las técnicas de “adaptación de dominio”, donde el desafío es entrenar un modelo con cierto dataset de un dominio “origen”, para luego utilizarlo de manera confiable sobre otro dataset de dominio “destino”. En el caso que se explora en este trabajo los dominios están dados por los centros médicos en que las imágenes han sido capturadas.

En el estado del arte existen diversas estrategias para lograr una correcta adaptación de dominio, que incluyen, entre otras, las supervisadas y no supervisadas. Ambas parten de la situación en la que se tiene un modelo entrenado sobre el dominio origen y su diferencia reside en que:

- Las supervisadas proponen utilizar una pequeña cantidad de datos etiquetados del dominio destino para reentrenar el modelo, perfeccionando su desempeño sobre el nuevo dominio. Es importante notar que en la estrategia supervisada es necesario tener cierta cantidad de datos etiquetados del dominio destino, suficiente para reentrenar el modelo.
- La estrategia no supervisada es empleada cuando no se dispone de datos etiquetados del dominio destino o no se cuenta con la capacidad de reentrenar el modelo. En estos casos, una de las técnicas que ha cobrado mayor relevancia en los últimos años es el uso de redes adversarias.

La capacidad de realizar una eficaz adaptación de dominio de manera no supervisada mediante una herramienta externa es deseable en los escenarios donde no se disponen de datos etiquetados. En el contexto de esta tesis de licenciatura, donde los dominios están dados por diferentes centros médicos que han utilizado diversos equipos de MRI para capturar las imágenes, dicha herramienta permitiría a los centros de salud que carecen de la capacidad de etiquetar sus datos sacar provecho de los datasets públicos, adaptándolos a su propio dominio, logrando así realizar análisis automáticos de sus imágenes médicas.

1.3. Objetivo

El objetivo de esta tesis de licenciatura es explorar la estrategia de adaptación de dominio no supervisada mediante redes adversarias de consistencia cíclica en el ámbito de la segmentación de lesiones cerebrales en imágenes 3D de resonancia magnética mediante redes neuronales profundas.

Se considera únicamente la segmentación de hiperintensidad de materia blanca (denominada WMH, por sus siglas en inglés), debido a que es una de las lesiones más difíciles de detectar en una MRI y de más variabilidad entre dominios. Asimismo, su asociación con distintas enfermedades, incluyendo diversos tipos de demencia, la convierte en una lesión importante a detectar de manera eficaz para la medicina moderna.

Específicamente, se propone investigar el comportamiento de la red convolucional U-Net 3D [7] al momento de trabajar con dominios diferentes y se pretende mejorar esta performance mediante el uso de CycleGAN como herramienta de adaptación de dominio no supervisada.

CycleGAN hace uso de redes adversarias, las cuales son entrenadas con elementos no etiquetados de dos dominios, y producen al final del entrenamiento dos funciones de “mapeo” (también denominadas “generadores”), las cuales “traducen” elementos de un dominio a otro. De esta manera, es posible adaptar imágenes 3D entre dominios, lo cual permite la segmentación sobre un dominio “objetivo” habiendo entrenado al modelo con datos etiquetados de otro dominio “origen”. En este trabajo se abarca la implementación de CycleGAN de dos maneras: de forma original en 2D y de forma alternativa modificada para su uso en 3D.

Los datasets utilizados deben ser de dominios diferenciados en diversos aspectos. A tal fin se elige trabajar con imágenes T1 y FLAIR del WMH Segmentation Challenge, organizado en MICCAI 2017 [8], las cuales se dividen en dos dominios de igual tamaño, donde la diferencia está dada por el centro médico de captura (y, por lo tanto, también se diferencian en el equipo utilizado, junto a sus parámetros de configuración).

1.4. Organización del trabajo

El presente documento está organizado en 6 secciones. En la Sección 2 se desarrollan los conceptos básicos requeridos para plantear el problema a abordar en este trabajo. Al principio, se exponen las técnicas clásicas de segmentación de imágenes médicas, las cuales involucran un manejo más “manual”. Luego, se aborda la explicación de métodos basados en redes neuronales profundas.

En la Sección 3 se plantea el problema de adaptación de dominio y se describen dos de sus posibles modalidades: supervisada en 3.2 y no supervisada en 3.3.

La Sección 3.5 abarca la estrategia de adaptación de dominio en la que se basa la propuesta realizada en este trabajo, la cual corresponde a la aplicación de CycleGAN como herramienta de transformación de MRIs de un dominio a otro de manera no supervisada. Se explica el modelo original de CycleGAN [9], examinando las distintas piezas que lo componen.

En la Sección 4 se describe la metodología de trabajo surgida de las ideas teóricas expuestas en las secciones anteriores. También se analizan y justifican las decisiones de implementación tomadas en el desarrollo del trabajo. Por último, se deja establecido el marco de experimentación en el cual se evaluaron de manera empírica las soluciones propuestas.

En la Sección 5 se exhiben e interpretan los resultados experimentales, buscando corroborar de manera empírica la utilidad de las ideas planteadas, midiendo su efectividad según distintos criterios en diversas situaciones.

Finalmente, en la Sección 6 se redactan las conclusiones extraídas de este trabajo y se plantean posibles trabajos futuros a ser desarrollados en la misma dirección.

2. SEGMENTACIÓN AUTOMÁTICA DE IMÁGENES MÉDICAS

2.1. Imágenes por resonancia magnética

En lugar de trabajar con radiación ionizante (como la radiografía o la tomografía), la resonancia magnética aprovecha el fenómeno magnético que ocurre sobre ciertos núcleos atómicos, cuyas propiedades cambian al ser expuestas a un campo externo. A través de estímulos magnéticos, los protones que se encuentran en el núcleo de los átomos del cuerpo humano provocan un pequeño momento o campo magnético como respuesta, el cual se manipula y censa para generar las imágenes. Este magnetismo nuclear tiene su origen en el spin nuclear.

El proceso de obtener una MRI se puede resumir en 4 pasos (ver Figura esquemática 2.1):

1. Se sitúa al paciente dentro de la máquina de MRI.
2. Se enciende el campo magnético del imán, con lo cual los núcleos de los átomos de hidrógeno del paciente se orientan de acuerdo a las líneas de fuerza del campo magnético.
3. Se aplica un estímulo de radiofrecuencia para cambiarlos de orientación. A este proceso se lo llama resonancia.
4. Cuando cesa el estímulo de radiofrecuencia, dichos núcleos liberan energía y vuelven a su situación inicial. Este proceso se conoce como relajación, y se mide de distintas maneras denominadas T1 y T2. Las diferencias de densidad nuclear y tiempos de relajación de los distintos tejidos del cuerpo determinan la intensidad de la señal de cada voxel.

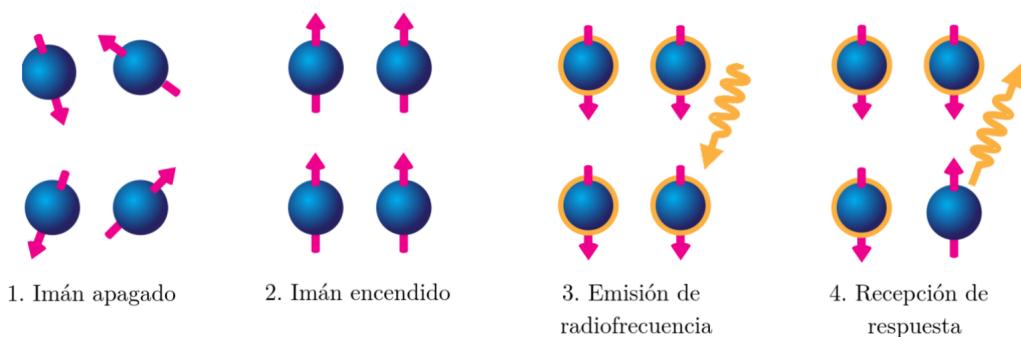


Fig. 2.1: Obtención de una imagen por resonancia magnética resumida en 4 pasos.

Existen diversos parámetros de adquisición de la imagen que determinan el estímulo magnético aplicado por la máquina. Algunos de ellos son Tiempo de Repetición (TR), Tiempo de Eco (TE) y Tiempo de Inversión (TI), entre otros. La variación de dichos parámetros tiene gran impacto en las imágenes producidas, como se puede visualizar en la Figura 2.2.

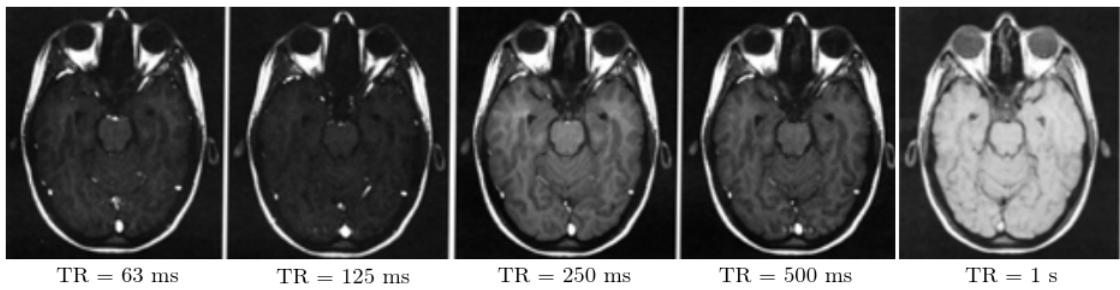


Fig. 2.2: Variación del parámetro de adquisición “Tiempo de Repetición” (TR) y su impacto en la MRI producida [10].

La elección de los parámetros determina qué modalidad de MRI se está utilizando. Cada variante genera imágenes en las que se resaltan distintos tejidos y sustancias, como las grasas, los agentes de contraste o el agua, entre otros, como se puede ver en la Figura 2.3. Vale la pena mencionar que dentro de cada modalidad existe cierto rango de libertad para los parámetros, lo cual implica la variabilidad entre las imágenes de una misma modalidad.

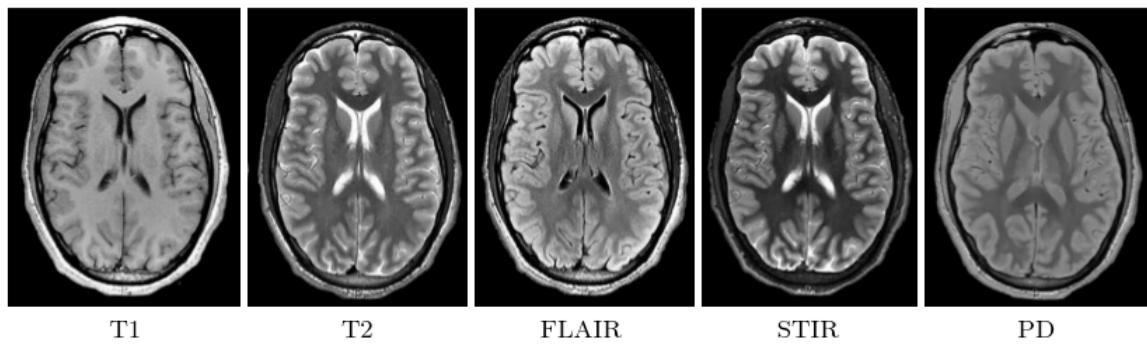


Fig. 2.3: Algunas de las diversas modalidades distintas de adquisición de MRI [11].

De esta manera, cada modalidad se especializa en distinguir ciertos aspectos anatómicos y patológicos. Un ejemplo de ello se verifica en la figura 2.4, donde FLAIR es de mayor utilidad para detectar y medir lesiones de Esclerosis Múltiple que sus contrapartes PD y T2.

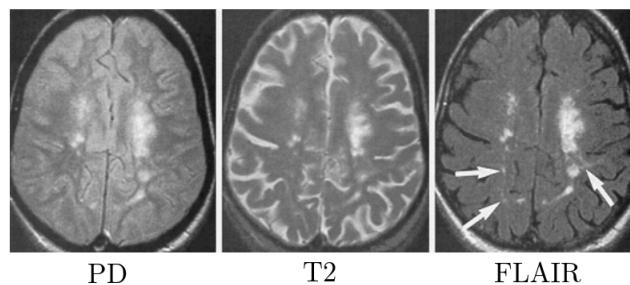


Fig. 2.4: Tres modalidades de MRI del mismo corte axial de un paciente. Se evidencia mucho mejor la presencia de Esclerosis Múltiple en la modalidad FLAIR que en PD y T2. [12]

Las tres maneras de realizar cortes 2D a partir de una MRI 3D se muestran en la figura 2.5. Los cortes se denominan axiales 2.5(a), sagitales 2.5(b) y coronales 2.5(c). A lo largo del presente documento se utilizan cortes axiales para exhibir los planos relevantes de las imágenes volumétricas.

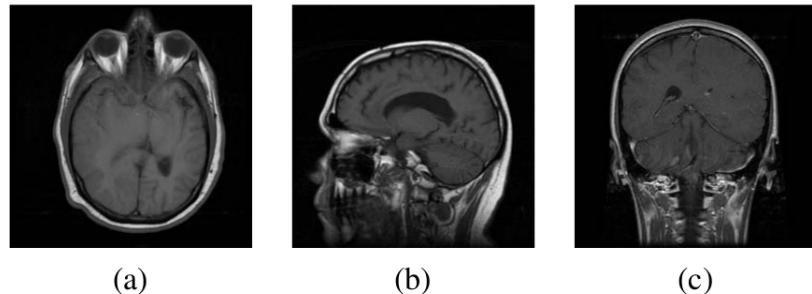


Fig. 2.5: Cortes de una MRI cerebral. (a) Corte axial, (b) corte sagital, (c) corte coronal [13].

2.2. Métodos clásicos de segmentación

A continuación se presentan algunos de los métodos clásicos de segmentación de imágenes médicas, los cuales no se basan en un proceso de aprendizaje sino que involucran una exploración “manual” de las imágenes para reconocer las diferentes características de las estructuras anatómicas y patológicas de la imagen.

2.2.1. Umbralado

El umbralado (thresholding en inglés) es una técnica de segmentación simple y fácil de implementar. Consiste en etiquetar como parte de la región de interés únicamente a aquellos píxeles (o véxeles) de la imagen con un valor de intensidad mayor a cierto umbral (threshold). Dicho umbral puede ser preestablecido por el usuario, o bien ser calculado automáticamente en función de la imagen. Además, puede extenderse permitiendo varios valores de umbral, lo cual se denomina técnica de multi-umbralado. Debido a su simplicidad es muy susceptible al ruido y raramente suele utilizarse de manera independiente, por lo que se combina con otros métodos.

2.2.2. Métodos orientados a la detección de bordes

En este grupo recaen aquellos métodos basados en la detección de bordes por medio de filtros convolucionales (Roberts, Prewitt o Sobel, entre otros). Su funcionamiento consiste en calcular el gradiente de cada punto mediante una operación convolucional (explicado en la Sección 2.4), donde el resultado es un número alto para el caso de un gradiente de gran valor y bajo en el caso contrario. De esta manera, la imagen resultante tiene distintos colores en aquellos píxeles donde hay saltos grandes con respecto a sus píxeles adyacentes, marcando así bordes en la imagen.

Esta técnica se remonta a los principios de la visión por computadora y se puede combinar con el procedimiento de detección de bordes de Canny [14], el cual consiste en:

1. Aplicar un filtro Gaussiano para eliminar ruido y suavizar la imagen en cada pixel.

2. Calcular los gradientes de cada píxel mediante alguna convolución especializada.
3. Suprimir los valores no máximos para conseguir el adelgazamiento del ancho de los bordes obtenidos con el gradiente hasta lograr bordes de un pixel de ancho.
4. Aplicar la función de histéresis basada en dos umbrales para remover los bordes “débiles”.

2.2.3. Crecimiento de Regiones

También llamado Region Growing, el enfoque de Crecimiento de Regiones propone una idea similar a los algoritmos de clustering aglomerativos. Se inicia a partir de uno o más puntos por región (llamados “semillas”). Luego, iterativamente, se va determinando si cada vecino a una región debería añadirse según un criterio determinado. Dicho criterio es planteado de antemano y se basa en características de la imagen tales como intensidad, brillo, color, gradiente y propiedades geométricas, entre otras. Cuando no quedan puntos que cumplan con el criterio de aceptación en la frontera de la región el proceso termina. [15]

2.2.4. Watershed

Similar al método de crecimiento de regiones, Watershed toma a la imagen como un mapa topográfico, donde las intensidades de los píxeles representan las diferentes alturas de los puntos. Se simula la “inundación” de dicho mapa desde sus puntos de menor altura, creando lagunas a partir de ellos. Las curvas formadas por las intersecciones de las lagunas forman las fronteras entre las distintas regiones de la imagen. [16]

2.3. Redes neuronales artificiales

Las redes neuronales artificiales (notadas RRNN) funcionan de manera alternativa a los métodos anteriormente descriptos. A diferencia de los métodos “clásicos”, donde los criterios para realizar la segmentación son pre-establecidos explícitamente de forma manual, un modelo de segmentación basado en RRNN “aprende” a partir de analizar múltiples ejemplos o “instancias”.

El entrenamiento consiste en presentarle al sistema una gran cantidad de instancias cuidadosamente seleccionadas tal que sean representativas de la tarea a realizar, y el sistema intenta ajustar una estructura estadística de modo que la lógica de dicha estructura prediga correctamente el comportamiento de dichas instancias. Este proceso de entrenamiento permite crear, eventualmente, un modelo que automatice el procesamiento de la tarea en cuestión.

En general, es necesario cierto preprocessamiento sobre los datos de entrada que le especifique “artesanalmente” a la red neuronal las características de las imágenes que deben ser analizadas, facilitándole así la tarea de aprendizaje. A esto se lo denomina “extracción de características” (“feature extraction”), y se realiza utilizando diversos métodos para extraer características relativas a la intensidad, textura y estructura de las imágenes, entre muchas otras. Sin embargo, usualmente la extracción “artesanal” de características es necesaria únicamente al utilizar modelos de redes neuronales de poca capacidad (como el perceptrón simple). Como se evidencia en este trabajo, en caso de las redes neuronales profundas y convolucionales dicha extracción de características no es necesaria, ya que los

modelos “aprenden” qué features utilizar y de qué manera durante el entrenamiento. A esto se lo denomina “aprendizaje de características” (“feature learning”).

La ventaja principal de las redes neuronales es su capacidad de generalización, la cual permite encontrar soluciones a problemas sin la necesidad de explicitar una técnica manual. Además, el costo computacional se concentra en el entrenamiento. Una vez entrenada, una red ofrece gran velocidad al momento de la aplicación sobre datos de test. Por último, se destaca la facilidad de reentrenar con datos nuevos una red ya entrenada, con el objetivo de mejorar la generalización o de ser utilizada en nuevos dominios.

2.3.1. Perceptrón simple

La red neuronal más básica es el **perceptrón simple**, el cual fue ideado como un modelo simplificado de una neurona biológica. Tiene como entrada un conjunto de variables x_1, \dots, x_n con sus respectivos pesos reales w_1, \dots, w_n , los cuales representan “la importancia” de cada x_i .

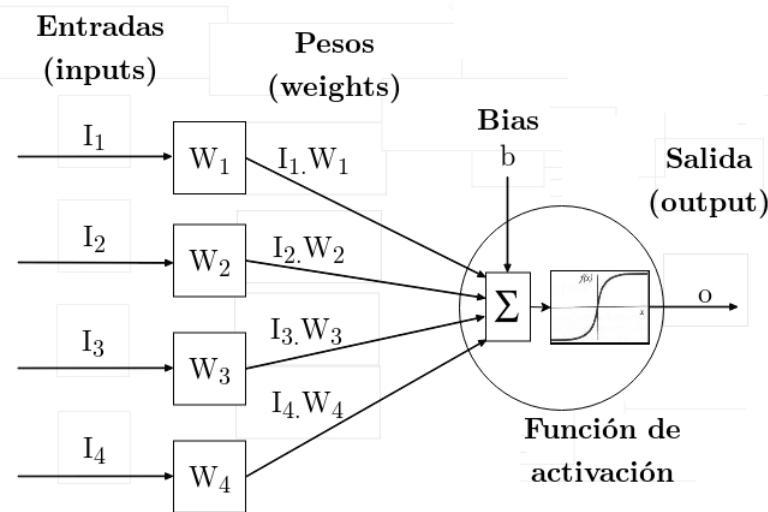


Fig. 2.6: Modelo de perceptrón simple de 4 entradas.

Formalmente, la salida de la neurona se define de la siguiente manera:

$$\text{output} = f(v) \quad (2.1)$$

donde f es la “función de activación” y v se define de la forma:

$$v = \left(\sum_{i=1}^n x_i w_i \right) + b \quad (2.2)$$

Para emular la sinapsis de la neurona biológica del cerebro humano, la función de activación tiene como salida un valor entre 0 y 1. De esta manera, cuando la salida tiene valor 1, se dice que la neurona “se activa”, y con valor 0 que “se desactiva”. Algunas de las funciones de activación básicas son:

- **Heaviside.** Ilustrada en la Figura 2.7(a), se calcula:

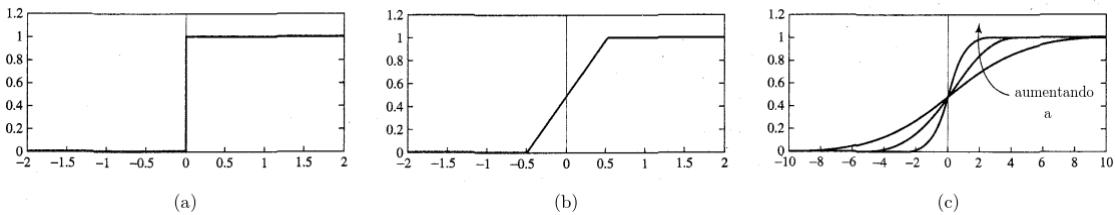
$$f(v) = \begin{cases} 1 & \text{si } v \geq 0 \\ 0 & \text{caso contrario} \end{cases} \quad (2.3)$$

- **Lineal.** Ilustrada en la Figura 2.7(b), se calcula:

$$f(v) = \begin{cases} 1 & \text{si } v \geq \frac{1}{2} \\ v & \text{si } \frac{1}{2} > v > -\frac{1}{2} \\ 0 & \text{si } -\frac{1}{2} \geq v \end{cases} \quad (2.4)$$

- **Sigmoide.** Correspondiente a la Figura 2.7(c), donde el parámetro “ a ” varía la pendiente de la función. Se define de la forma:

$$f(v) = \frac{1}{1 + e^{-av}} \quad (2.5)$$



Se puede pensar al perceptrón simple como la representación de un hiperplano de decisión en el espacio n-dimensional de instancias (por ejemplo, puntos). El perceptrón tiene como salida un 1 para instancias de un lado del hiperplano, y 0 para instancias del otro, como se puede visualizar en las Figuras 2.8(a) y 2.8(b).

Sin embargo, existen conjuntos de instancias que no son posibles de separar con ningún hiperplano, como lo demuestra la Figura 2.9. A aquellos que sí tienen dicha capacidad se los califican como “linealmente separables”.

2.3.2. Perceptrón multicapa

Al conectar múltiples neuronas entre sí, se adquiere la capacidad de separar conjuntos más complejos que los linalmente separables.

Se denomina “arquitectura” a la topología, estructura o patrón de conexiones de una red neuronal. En una red neuronal multicapa los nodos se conectan por medio de conexiones sinápticas, donde la estructura de las conexiones determina el comportamiento de la red. Las neuronas se suelen agrupar en unidades estructurales llamadas “capas”, y el conjunto de varias capas constituye la arquitectura de la red neuronal. En el presente trabajo se utilizan únicamente redes *feedforward*, donde las conexiones sinápticas son direccionales, es decir, la información solamente puede propagarse en un único sentido.

Así, cada arquitectura comienza con una capa de entrada, seguida de n capas intermedias llamadas “ocultas”, finalizando con una capa final. Dos ejemplos de arquitecturas simples se ilustran en la Figura 2.10. Las capas de dicha figura se denominan “densas” o “fully connected”, en las que cada neurona se conecta con toda neurona de la capa siguiente.

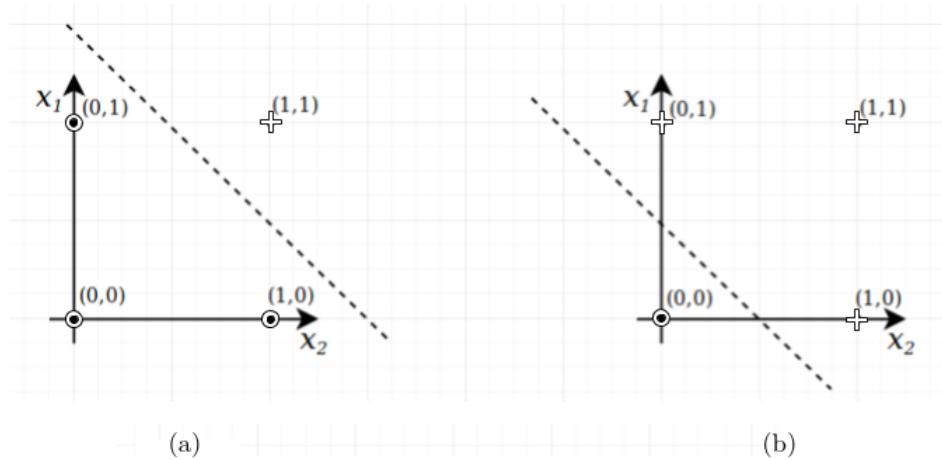


Fig. 2.8: Dos ejemplos de conjuntos linealmente separables. (a) corresponde a la función booleana AND. (b) corresponde a la función booleana OR.

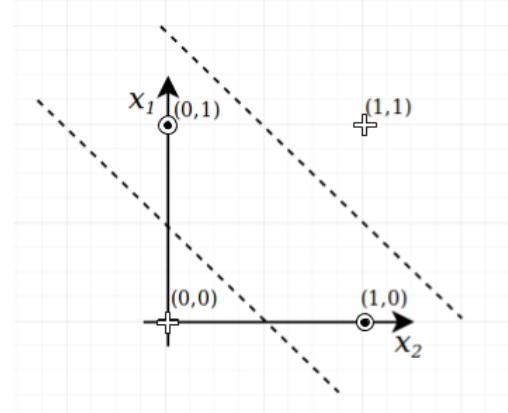


Fig. 2.9: Ejemplo de un conjunto no linealmente separable, correspondiente a la función XOR.

2.3.3. Descenso por el gradiente

Para medir la calidad de la separación en clases realizada por el hiperplano generado por la red neuronal se utiliza lo que se llama una “función de costo” o “función de pérdida”. Durante el entrenamiento, la red actualiza iterativamente sus pesos buscando reducir el valor de la función de pérdida.

Una función de pérdida muy popular consiste en tomar las diferencias cuadradas entre los resultados esperados y los obtenidos:

$$E(\bar{w}) = \frac{1}{|D|} \sum_{d \in D} (t_d - o_d)^2 \quad (2.6)$$

donde \bar{w} es el vector de pesos, D es el conjunto de instancias de entrenamiento, t_d la salida esperada para la instancia d , y o_d la salida de la red.

Para encontrar los valores de \bar{w} que minimicen $E(\bar{w})$ se suele utilizar un método popular de optimización llamado “Descenso por el gradiente”, el cual consiste en la utilización del

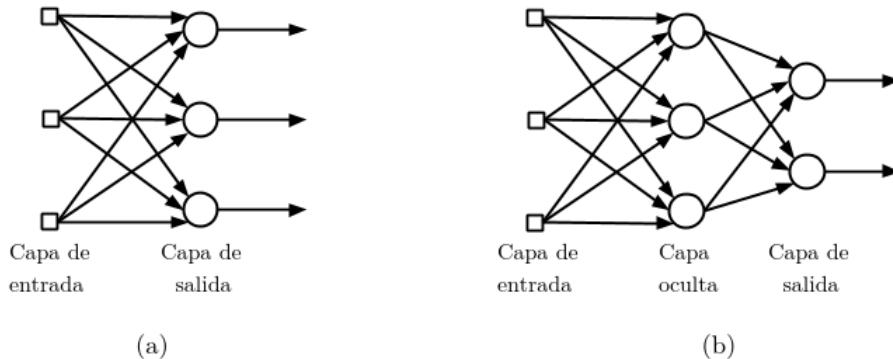


Fig. 2.10: Dos arquitecturas *feedforward* simples. (a) de 2 capas y (b) de 3 capas.

gradiente de la función de pérdida para buscar iterativamente el valor mínimo de $E(\bar{w})$.

Para el caso de una red de dos entradas, se expone una visualización de los posibles \bar{w} y sus valores de error asociados en la Figura 2.11. Se llama “hipótesis” a \bar{w} , y “espacio de hipótesis” al espacio en el que vive.

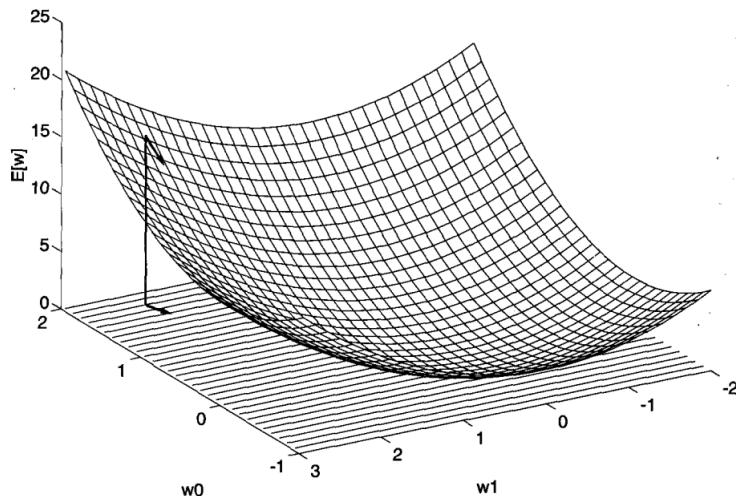


Fig. 2.11: Error para las distintas hipótesis. Para dos pesos, el espacio de hipótesis H es el plano $< w_0, w_1 >$. El eje vertical indica el error asociado a la hipótesis, relativo al conjunto de instancias de entrenamiento. La flecha indica el menor gradiente en un punto particular, marcando la dirección en el plano w_0, w_1 con mayor pendiente descendente en la superficie del error.[18]

Descenso por el gradiente consiste en avanzar de manera iterativa en la dirección en la que disminuye el gradiente. La elección de cuánto avanzar en cada iteración está controlada por un factor denominado “learning rate”, que puede ser constante o adaptativo, dependiendo de la variante del método utilizada (algunos ejemplos son Adam [19], Adagrad [20], entre otros).

En caso del Perceptrón Multicapa resulta necesario encontrar el gradiente de error para los pesos de las capas intermedias, no sólo de aquellos utilizados en la última capa.

Para obtener el gradiente correspondiente a los pesos de las neuronas en dichas capas intermedias es necesario propagar los errores de la capa de salida “hacia atrás” en la red. Este procedimiento se conoce como “Back-propagation”, y está basado en la aplicación sucesiva de la regla de la cadena. Para una explicación detallada del método, ver el trabajo que lo propone [21].

2.4. Redes neuronales convolucionales

La convolución es una operación matemática definida como la integral del producto de dos funciones, donde una es desplazada y reflejada. En el caso de las imágenes digitales, que pueden ser consideradas funciones discretas de dos variables, **la convolución** puede implementarse como la multiplicación matricial entre distintas submatrices de la entrada y una matriz fija denominada “filtro” o “kernel”. La elección de dichas submatrices depende de los saltos (“stride”) de la convolución. Finalmente, al resultado puede agregársele determinado espaciado (“padding”).

Para dos dimensiones, se define formalmente al resultado de una convolución g entre una imagen f y un kernel h de la manera:

$$g(i, j) = \sum_{k,l} f(i - k, j - l)h(k, l) \quad (2.7)$$

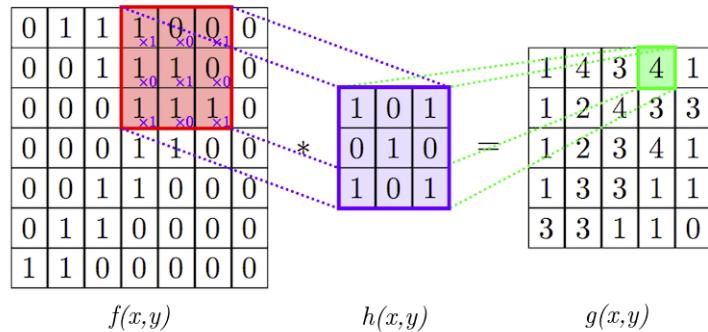


Fig. 2.12: Ejemplo de convolución en dos dimensiones.

Las convoluciones tienen un rango de utilidad muy amplio. Un uso muy común es la convolución con filtros Gaussianos para suavizar una imagen con ruido. Otro ejemplo es utilizar los filtros de Roberts, Prewitt o Sobel para detectar saltos grandes en el gradiente de la imagen marcando los bordes de la misma.

Las convoluciones pueden ser utilizadas para procesar imágenes con varios canales (RGB por ejemplo). Se muestran dos ejemplos de ello en las Figuras 2.13 y 2.14.

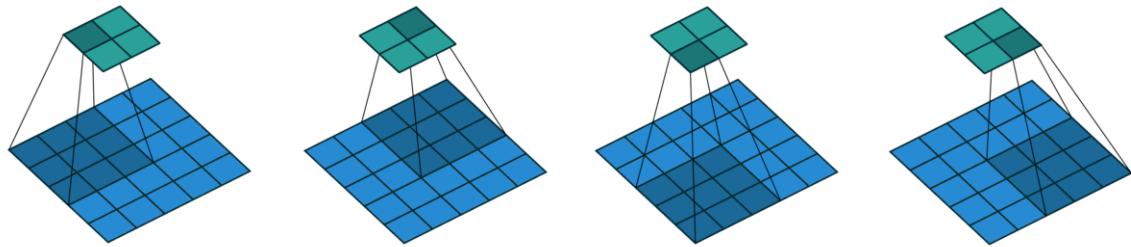


Fig. 2.13: Ejemplo de convolución con un filtro de dimensiones 2×2 sobre una imagen de un canal.

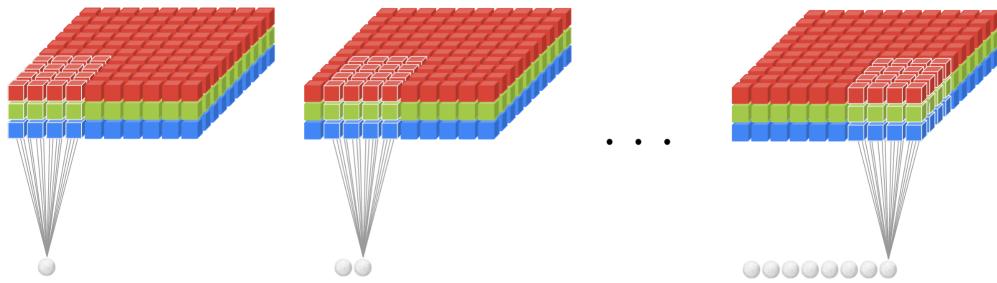


Fig. 2.14: Ejemplo de convolución con un filtro de dimensiones $4 \times 4 \times 3$ sobre una imagen de 3 canales (rojo, verde y azul).

En el contexto de las redes neuronales, las redes neuronales convoluciones (CNN por sus siglas en inglés) utilizan **capas convolucionales** donde dicha operación es aprovechada para aprender filtros capaces de extraer características útiles para resolver el problema en cuestión. Sus principales hiperparámetros son el tamaño del kernel, padding y stride, mientras que los valores del kernel se infieren durante el entrenamiento. De esta forma, un aprendizaje exitoso resulta en una capa convolucional capaz de detectar features útiles para el objetivo de la red, aprovechando el amplio rango de utilidad de las convoluciones.

Una cualidad valiosa de las capas convolucionales es la invariancia a la traslación: como la convolución es realizada sobre toda la imagen utilizando el mismo filtro con soporte local, las características resaltadas por dicho filtro son detectadas independientemente de su posición, a diferencia de las capas densas (ver Figura 2.17). De esta manera, por ejemplo, una CNN que detecte tumores cerebrales puede ser entrenada con un dataset que contenga únicamente tumores en el hemisferio derecho y aún así poder detectar tumores en el izquierdo.

Otra cualidad notable de las capas convolucionales es que, a diferencia de las capas densas, las neuronas de una capa convolucional comparten los mismos pesos, lo cual disminuye significativamente la cantidad de parámetros a ajustar. De esta manera, se reduce la complejidad del modelo además del tiempo necesario para el entrenamiento. Se puede visualizar la comparación entre capas densas y convolucionales en la Figura 2.17.

Las redes neuronales convolucionales resultan particularmente útiles para el procesamiento de imágenes y video, debido a que las capas convolucionales aprovechan la distribución espacial de la entrada. Además, suelen incluir otras capas específicas, tales como Convolución Transpuesta (ver Sección 2.5.1) y Pooling (Sección 2.4.1).

Las CNN suelen utilizar convoluciones intercaladas con poolings, incrementando con cada capa la región de la imagen de entrada que procesa cada neurona. A dicha región se le llama “campo receptivo” y se puede visualizar un ejemplo en la Figura 2.15. De esta forma:

- Las primeras capas funcionan como herramientas básicas de Visión por Computadora (detectando bordes y esquinas de manera local).
- Las intermedias detectan características más complejas (formas, figuras).
- Las finales detectan features particulares del dominio en grandes regiones de la imagen (orejas de gatos y perros en clasificación de animales, por ejemplo)

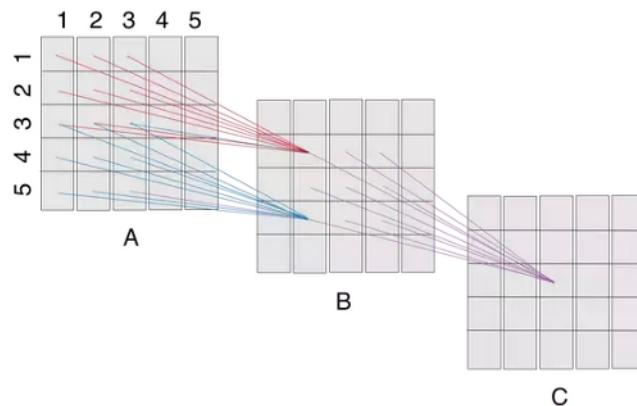


Fig. 2.15: Ejemplo visual del campo receptivo. Cada neurona procesa un mayor tamaño de la imagen de entrada.

2.4.1. Capa de Pooling

Las capas de “pooling” reducen las dimensiones de la entrada mediante la combinación de la salida de múltiples neuronas de una capa en la entrada de una única neurona de la capa siguiente. Dichas combinaciones pueden ser locales para una vecindad de píxeles (por ejemplo 2×2) o globales (a partir de la imagen completa).

El objetivo de las capas de pooling es filtrar activaciones ruidosas de capas previas mediante la abstracción de las activaciones en el campo receptivo resultando en un único valor representativo. Dicho valor suele ser el máximo o el promedio de dicho campo receptivo. Por tanto, las combinaciones más comunes de la capa de pooling son *MaxPooling* (el máximo) y *AvgPooling* (el promedio), como lo indica la Figura 2.16.

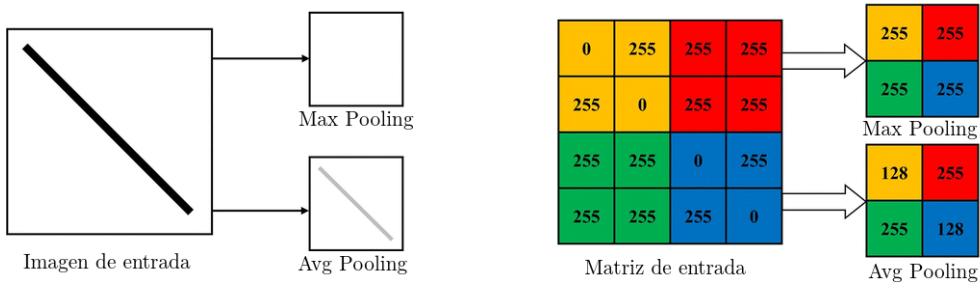


Fig. 2.16: Representación visual y numérica de Pooling promedio (AvgPooling) y máximo (MaxPooling) [22].

2.4.2. Redes neuronales puramente convolucionales

Se llama red puramente convolucional a aquella que no tiene capas densas en su arquitectura. La ventaja que tienen las redes puramente convolucionales es la capacidad de procesar entradas de cualquier tamaño, tanto en entrenamiento como en evaluación.

Esto se debe a que no utilizan capas densas para el aprendizaje, las cuales fijan el tamaño de la entrada al determinar que cada neurona se conecte con todas las de la capa anterior. Como se visualiza en la Figura 2.17, la cantidad de conexiones de la capa densa depende directamente de la cantidad de inputs, mientras que en la capa convolucional la cantidad de conexiones depende del tamaño de kernel.

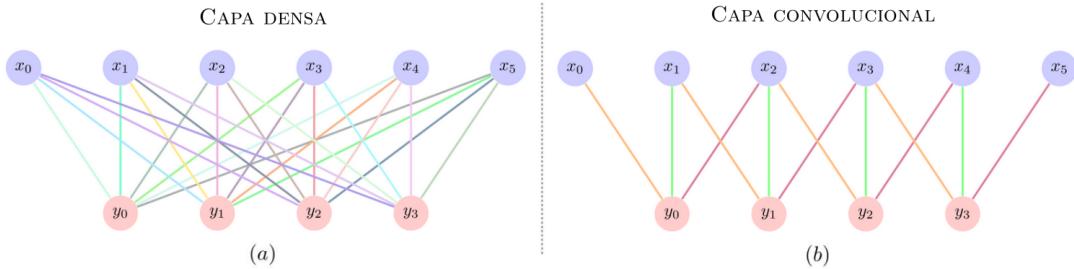


Fig. 2.17: Comparación entre una capa densa y una convolucional, donde cada conexión se representa con un color distinto. La capa densa tiene una conexión distinta por cada par de neuronas (en este caso $6 \times 4 = 24$), mientras que la capa convolucional utiliza las mismas 3 conexiones para toda la entrada independientemente de la cantidad de neuronas.

Algunas arquitecturas puramente convolucionales a destacar son U-Net [7] y las implementaciones de VGG [23] y GoogLeNet [24] propuestas en [25].

2.5. Redes neuronales para la segmentación de imágenes

La segmentación de imágenes es uno de los problemas más importantes del área de visión por computadora. Es una tarea de alto nivel esencial en la comprensión de escenas, lo cual se destaca por el gran número de aplicaciones que surgen en la actualidad, tales como vehículos autónomos, interacciones entre humanos y computadoras, realidad virtual, entre otras.

Es posible pensar la segmentación como una extensión de la clasificación, la cual consiste en la predicción de etiquetas para una entrada. Esto significa que, dada una imagen, la red clasificadora responde qué etiquetas le corresponden a la imagen. El siguiente paso lógico es la detección y localización, que provee no solamente las clases presentes en la imagen sino también información adicional sobre la ubicación espacial de dichas clases. Finalmente, la segmentación realiza predicciones para cada píxel, asignando a cada uno sus etiquetas correspondientes.

La segmentación de una imagen debe tener las mismas dimensiones que la imagen de entrada. Considerando que las capas convolucionales y de pooling reducen las dimensiones, es necesario agregar en la arquitectura de la red segmentadora capas que reviertan la reducción de dimensiones. Para ello, se busca obtener una imagen de alta resolución a partir de una de baja. Esto se logra con una operación de “convolución transpuesta”, la cual permite al modelo “pintar” varios vóxeles de salida a partir de cada vóxel de entrada.

2.5.1. Capa de convolución transpuesta

La convolución transpuesta consiste en una operación convolucional especial, cuyo objetivo es revertir la concentración de información realizada por las capas de convolución y pooling. Para ello se invierte el sentido del stride: en lugar de aplicar saltos sobre la entrada, se aplican saltos sobre la salida, rellenando con 0 en los elementos intermedios para lograr el tamaño de salida deseado, y luego utilizando una convolución estándar.

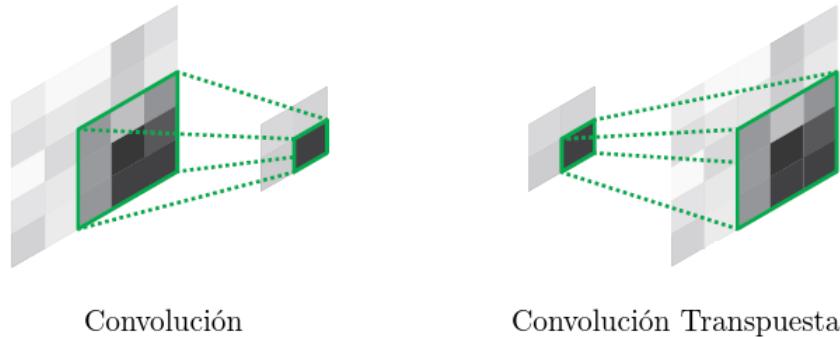


Fig. 2.18: Comparación entre capa de convolución y de convolución transpuesta. La convolución transpuesta no es exactamente la inversa de la convolución, pero simula serlo. [26].

2.5.2. Capa de dropout

En aprendizaje automático suele ocurrir que un algoritmo de aprendizaje sea “sobreentrenado” para cierto conjunto de datos. Este fenómeno, llamado “sobreajuste” (“overfitting”), resulta en un modelo ajustado a características muy específicas de los datos de entrenamiento que no tienen relación causal con la función objetivo. Así, un modelo sobreentrenado tiene gran capacidad de predicción sobre instancias del dataset de entrenamiento, pero muy poca para operar sobre datos nuevos.

Un motivo común de overfitting en CNN radica en la concentración del aprendizaje en unas pocas neuronas que realizan todo el trabajo. Esto genera modelos de alta variabilidad muy susceptibles a cambios en los datos de entrada.

La capa de Dropout [27] solventa este problema mediante la desactivación de un porcentaje de las neuronas en cada iteración del aprendizaje. Dicho porcentaje es un hiperparámetro de la capa de Dropout y la elección de las neuronas a desactivar se hace de manera aleatoria y varía en cada iteración.

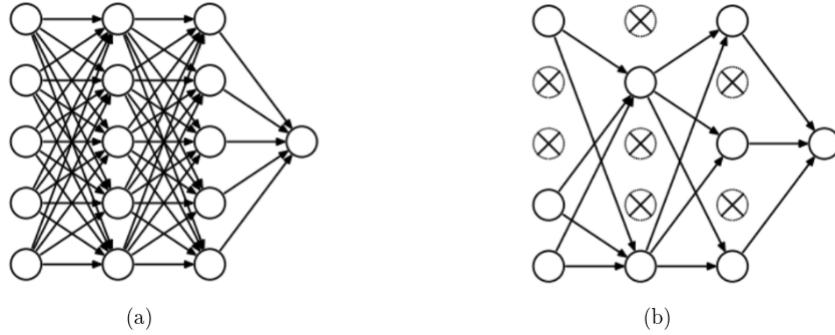


Fig. 2.19: Ejemplo del funcionamiento de Dropout. (a) representa una red en su arquitectura original y (b) la red en una iteración aplicando Dropout.

Es importante agregar que Dropout únicamente desactiva neuronas durante el proceso de entrenamiento y no durante la evaluación. La gran popularidad de Dropout se debe al bajo costo que requiere su implementación en contraste con su efectividad para reducir el sobreajuste.

2.5.3. Batch Normalization & Instance Normalization

Al utilizar múltiples capas ocultas en la topología de la red, los pesos de la misma son particularmente sensibles a la variación de la distribución de los datos entre cada “batch” de entrenamiento. Gran variabilidad de los datos puede causar que la red nunca logre converger en una buena generalización.

Una solución popular y eficiente a dicho problema es utilizar la capa “Batch Normalization” [28] luego de cada Convolución, la cual ajusta y escala las activaciones de una capa, normalizándola con media 0 y varianza 1. De esta manera, Batch Normalization mejora la velocidad, eficiencia y estabilidad de una red neuronal durante su entrenamiento.

Existen otras alternativas menos populares a Batch Normalization con usos más específicos, tales como Instance Normalization (que normaliza con respecto a una instancia en lugar del batch entero), Layer Normalization (que normaliza con respecto a los canales de entrada) y Group Normalization (que normaliza con respecto a subconjuntos de los canales de entrada) [29].

2.6. Arquitectura U-Net 3D para segmentación de imágenes

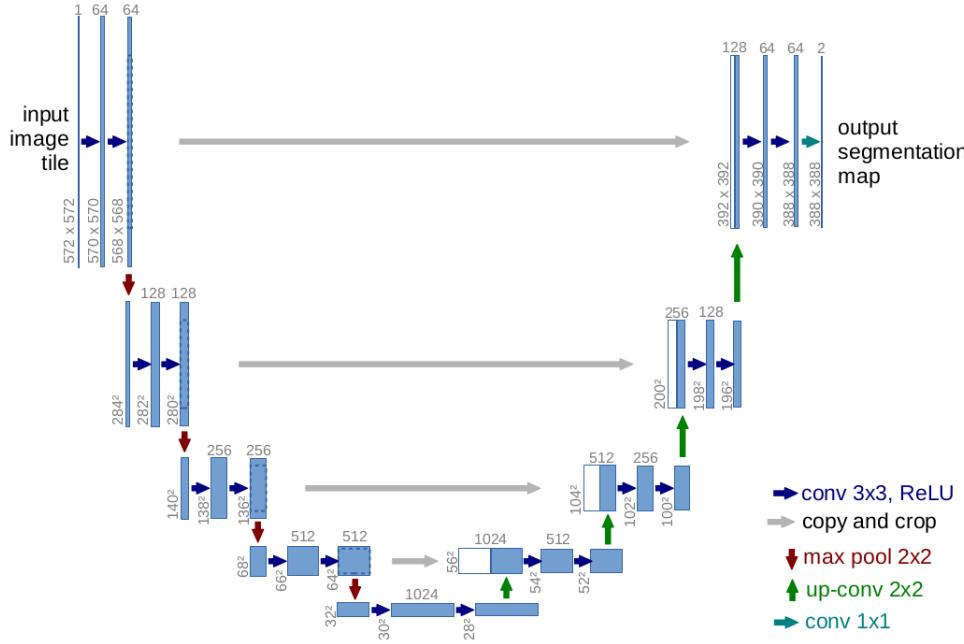


Fig. 2.20: Esquema de la arquitectura U-Net, dividido en un camino de contracción (izquierda) y uno de expansión (derecha). Cada bloque azul corresponde a un volumen de características. El número de canales se nota en la parte superior de cada bloque, mientras que las dimensiones se anotan en la parte inferior izquierda. Las flechas representan las diferentes operaciones [7].

La arquitectura U-Net consiste en un camino de contracción (encoder) y uno de expansión (decoder). La contracción sigue la típica arquitectura de una red convolucional. Consiste en la aplicación reiterada de dos convoluciones de 3×3 seguidas de una activación ReLU y una capa de *Max Pooling* de 2×2 con stride 2. Estas secuencias de componentes realizan el “downsampling”, es decir, la condensación de la entrada en un mapa de características de menor dimensión, encapsulando la información cada vez más en cada iteración.

Por otro lado, cada paso del camino de expansión consiste en el “upsampling” del mapa de características, seguido de una convolución de 2×2 . Luego, se aplica una concatenación con el mapa de características correspondiente del camino de contracción, lo cual tiene como objetivo combinar información de alto nivel del camino de expansión con información localizada del camino de contracción. Alternativamente, esta concatenación puede ser reemplazada por una suma elemento a elemento, lo cual reduce la complejidad del modelo [30]. Finalmente, se realizan dos convoluciones de 3×3 , cada una con activación ReLU.

La capa final es una convolución de 1×1 , la cual funciona como reemplazo de una capa densa. A la salida de dicha capa se aplica una función Softmax [31], generando un vector de probabilidades de cada clase para cada pixel. [7]

3. ADAPTACIÓN DE DOMINIO MEDIANTE CYCLEGAN

3.1. Introducción

Las redes neuronales profundas han demostrado ser de amplia utilidad en el análisis de imágenes médicas, con mayor eficacia que los métodos convencionales sobre tareas específicas tales como clasificación y detección. Por ejemplo, en análisis de MRI, las CNN han demostrado alcanzar rendimiento excepcional en diversas tareas tales como segmentación de hiperintensidad de materia blanca (WMH) [6], segmentación de tumores [32] y detección de microsangrado [33].

Si bien muchos estudios muestran resultados excelentes sobre dominios específicos, no es común encontrarse con investigaciones que evalúen la generalizabilidad de estos modelos variando la distribución de los datos de test. Esto genera la necesidad de investigar el campo de adaptación de dominio para garantizar el uso adecuado de los modelos y su aplicabilidad sobre datos reales, los cuales comúnmente varían en distintos aspectos, como ser marca del escáner, protocolo y parámetros. Esto se acentúa al trabajar sobre MRI, donde la distribución de intensidades, apariencia y contraste de las zonas de tejido blando suele presentar gran variabilidad cuando se utilizan distintos equipos de captura o configuración de los mismos.

Formalmente, un dominio D puede ser expresado por un espacio de características (features) χ y una distribución de probabilidad $P(X)$, donde $X = \{x_1, \dots, x_n\} \in \chi$. Dado un dominio específico $D = \{\chi, P(X)\}$, una tarea de aprendizaje sobre dicho dominio consiste en dos componentes: un espacio de etiquetas Y y una función objetivo de predicción $f(\cdot)$. Se nota a la tarea como $T = \{Y, f(\cdot)\}$. La función objetivo $f(\cdot)$ puede ser “entrenada” a partir de los datos de entrenamiento, los cuales consisten en pares $\{x_i, y_i\}$, donde $x_i \in X$ y $y_i \in Y$. Al finalizar el proceso de entrenamiento, el modelo entrenado $f(\cdot)$ se utiliza para predecir la etiqueta de una nueva instancia x [34].

3.2. Adaptación de dominio supervisada

En el contexto de la adaptación de dominio supervisada se dispone de un modelo entrenado sobre un dominio S , al cual se lo quiere adaptar para su utilización sobre elementos de otro dominio U , del cual también se tienen datos etiquetados. La adaptación consiste en transferir los conocimientos aprendidos en el dominio original para aplicarlos en el etiquetado de elementos del dominio destino. A esto se lo denomina “transferencia de aprendizaje” o “transfer learning” (TL).

Formalmente, dado un dominio fuente S con una tarea de aprendizaje T_S y un dominio objetivo U con una tarea de aprendizaje T_U , se define “aprendizaje por transferencia” como el proceso de mejorar el aprendizaje de la función predictiva objetivo $f_U(\cdot)$ sobre U utilizando la información de S y T_S , donde $S \neq U$ y $T_S \neq T_U$ [34]. Se nota como $\tilde{f}_{SU}(\cdot)$ al modelo predictivo inicialmente entrenado con los datos de S y adaptado al dominio objetivo U .

Una técnica común de TL consiste en realizar dicha transferencia de conocimientos mediante un reentrenamiento sobre el dominio destino, variando los hiperparámetros para hilar fino (se lo llama “fine-tuning”) y adaptarse a las sutiles diferencias del nuevo dominio.

[35]

Además, es posible variar este reentrenamiento mediante la modificación de la red. Por ejemplo, en [36] se propone el congelamiento de las capas iniciales de la red, preservando más aún el conocimiento adquirido del dominio origen. Alternativamente, el trabajo [37] agrega una función de pérdida de “confusión de dominio” para restringir el aprendizaje de la red a únicamente atributos invariantes al dominio. Por último, resulta importante destacar que el potencial de TL se extiende más allá de la adaptación de dominio dentro de una tarea específica: el trabajo [38] demuestra que es posible entrenar una red multi-propósito capaz de segmentar tejido cerebral, músculo pectoral y arterias coronarias.

En las pruebas preliminares del presente trabajo se abordó la adaptación de dominio supervisada correspondiente al escenario donde se tienen unos pocos datos etiquetados en el dominio objetivo. Dichos experimentos demostraron que es posible adaptar un modelo en el mismo marco de experimentación de la Sección 5 realizando simplemente fine-tuning de los pesos, tal como lo sugiere la literatura [36]. Por esa razón, se decidió explorar un escenario más complejo, en el cual no se dispone de datos etiquetados en el dominio objetivo.

3.3. Adaptación de dominio no supervisada

La adaptación de dominio no supervisada se define de la misma manera que la supervisada, con la diferencia clave que no se tienen datos etiquetados en el dominio objetivo. En consecuencia, no se puede construir un modelo $\tilde{f}_{SU}(\cdot)$ a partir de $f_S(\cdot)$ utilizando “aprendizaje por transferencia”, al no disponerse de datos etiquetados del dominio U .

3.3.1. Traducción de imágenes con datos pareados

Uno de los enfoques más sencillos para realizar DA no supervisada es aquel con el requerimiento más fuerte: tener las correspondencias entre los datos de entrenamiento de S y U (es decir, “tener datos pareados”).

Formalmente, un conjunto de datos pareados consiste en instancias $\{x_{S_i}, x_{U_i}\}_{i=1}^N$, donde se conoce una correspondencia entre cada x_{S_i} y x_{U_i} .

Ciertas técnicas de DA pareado se asemejan a las de TL supervisado. Por ejemplo, el trabajo [39] propone reentrenar $f_S(\cdot)$ sobre U penalizando las diferencias entre cada par $(f(x_{S_i}), f(x_{U_i}))$ mediante una “pérdida de consistencia”.

Otro enfoque consiste en construir una función de traducción $G : U \rightarrow S$ aprovechando las parejas para evaluar la calidad de la transformación. De esta manera, se puede aplicar $f_S(\cdot)$ sobre elementos de $G(U)$, obteniendo predicciones del dominio del cual no se tienen datos etiquetados. Esta transformación realizada por G requiere trabajar con suficiente precisión para modificar aquello que depende del dominio mientras que conserva la información de la imagen importante para la tarea a resolver.

Dichas funciones de traducción pueden producirse mediante “redes adversarias generativas” (GAN), las cuales han demostrado su amplio rango de aplicaciones en el procesamiento de imágenes y visión por computadora, como mejorar la resolución de una imagen [40], colorizar imágenes en escala de grises [41], entre muchas otras.

3.3.2. Traducción de imágenes sin datos pareados

En la práctica, resulta inusual disponer de datos pareados, ya que en general esta posibilidad existe únicamente cuando un dominio puede construirse a partir del otro. Por ejemplo, al tener los dominios “imágenes en color” e “imágenes en escala de grises”, el segundo domino puede producirse a partir del primero.

En el ámbito de la segmentación de MRI es evidente lo difícil que sería producir un dataset pareado: se deberían tener dos imágenes de cada paciente, capturadas en el mismo momento, donde cada imagen pertenezca a uno de los dos dominios.

En consecuencia, en este trabajo se aborda el enfoque de traducción imagen a imagen sin datos pareados.

Las técnicas de DA no supervisado sin datos pareados más populares en el estado del arte se basan en el uso de redes adversarias. Por ejemplo, [42] propone como adversaria de la red segmentadora a una red discriminadora de dominio, que induce el aprendizaje de características invariantes al dominio en la red segmentadora.

Alternativamente, se puede buscar una función de traducción $G : U \rightarrow S$ de igual manera que en las estrategias de datos pareados. El problema es: ¿Cómo evaluar $G(x_{U_i})$, si no se conoce su pareja x_{S_i} en el dominio objetivo? La solución mediante redes adversarias son las redes adversarias generativas (GAN). Un ejemplo de su potencial se demuestra en [43], que utiliza GAN para adaptar un dominio de objetos renderizados sintéticos a un dominio de objetos en fotografías reales. Otro ejemplo es [44], que utiliza CycleGAN para transformar imágenes urbanas de videojuegos al dominio de fotografías urbanas (específicamente del dataset CityScapes [45]).

En este trabajo se eligió abordar la adaptación de dominio no supervisada sin datos pareados mediante las redes adversarias cíclicas (Cycle-GAN), de forma similar a la estrategia adoptada en [46] para el caso de imágenes oftalmológicas 2D, pero con el foco sobre lesiones cerebrales en MRI tridimensionales.

3.4. Redes adversarias generativas (GAN)

Las redes adversarias generativas (llamadas GAN, por sus siglas en inglés) se basan en un modelo cuyo entrenamiento funciona mediante la simulación de un juego de dos participantes: un “generador” y un “discriminador”.

- **El generador** aprende a generar instancias (imágenes, por ejemplo), que se parezcan a las instancias verdaderas de la distribución subyacente de los datos de entrenamiento.
- **El discriminador** es su adversario. Aprende a distinguir entre las imágenes falsas producidas por el generador y las instancias reales de los datos.

De esta manera, el generador intenta engañar al discriminador, mientras que el discriminador intenta determinar cuáles instancias son reales y cuáles falsas. A medida que progresa el entrenamiento, ambos jugadores “aprenden” incrementalmente a ser mejores: el discriminador a detectar imágenes falsas y, lo que es más importante, el generador a producir imágenes más realistas (lo cual es el objetivo final de dicho entrenamiento).

La primera arquitectura de GAN fue propuesta en el trabajo pionero de Ian Goodfellow et al. [47], donde el generador tiene como entrada un vector de ruido, el cual determina la imagen producida. Se visualiza un esquema de la arquitectura en la Figura 3.1 mientras que algunos resultados de su trabajo se presentan en la Figura 3.2.

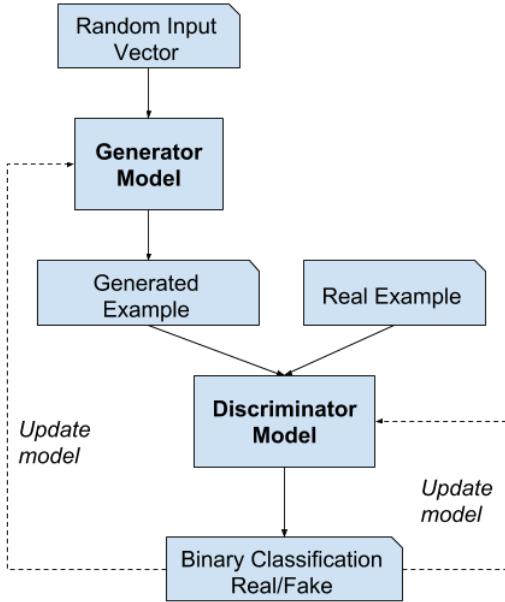


Fig. 3.1: Esquema de arquitectura de GAN del trabajo de Goodfellow et al. [47].

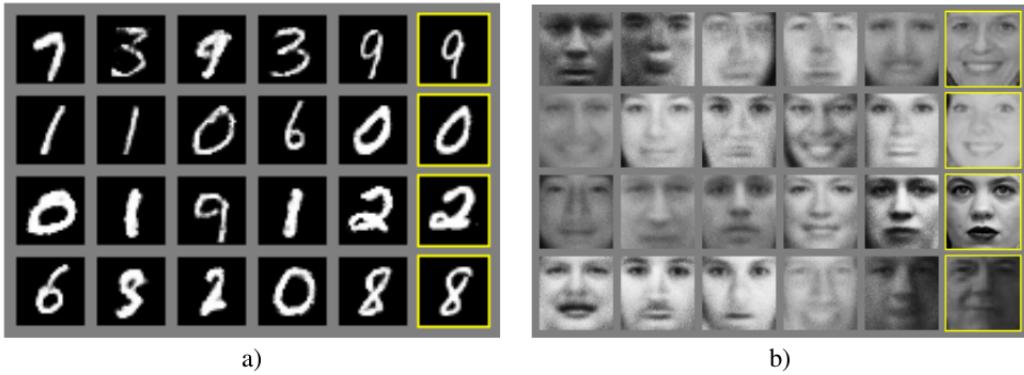


Fig. 3.2: Algunos resultados del trabajo de Goodfellow et al. [47]. Para ambos datasets, las 5 columnas de la izquierda son imágenes generadas con GAN. Por otro lado, la columna de la derecha muestra la instancia real más similar a su vecino más cercano, para demostrar que el modelo no memoriza el dataset sino que genera instancias nuevas. Los datasets son a) MNIST [48] y b) TFD [49].

La técnica de GAN puede ser adaptada de múltiples maneras para solucionar diversos problemas, entre los cuales se encuentran adaptación de dominio supervisada y no supervisada (ver ejemplos en las Secciones 3.3 y 3.2 respectivamente).

3.5. Redes adversarias de consistencia cíclica (Cycle-GAN)

Cycle-GAN es una adaptación del modelo original de GAN con el objetivo específico de traducir instancias entre dominios sin tener los datasets pareados. Al no tener las instancias pareadas, Cycle-GAN debe abordar la función de pérdida desde otra perspectiva, trabajando con tres funciones de pérdida distintas: adversaria, de consistencia cíclica y de mapeo de identidad. Cada una regula el comportamiento de las redes generadoras en un aspecto en particular, y únicamente con un correcto balance entre las tres se puede llegar a una correspondencia efectivamente útil para la adaptación de dominio.

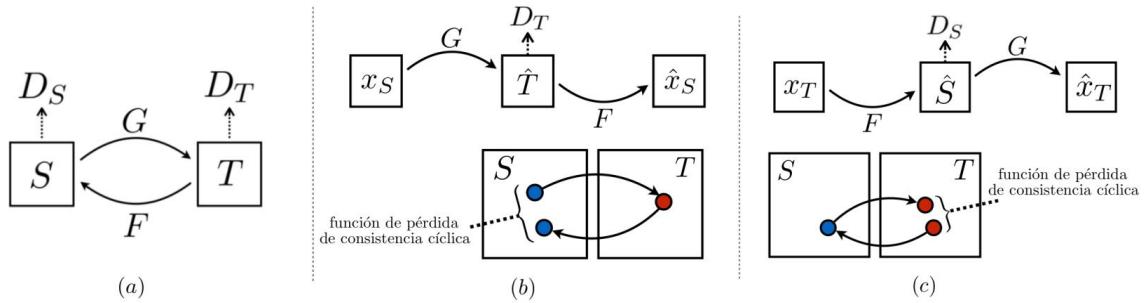


Fig. 3.3: (a) Esquema de la interacción entre los 2 modelos generadores $G : S \rightarrow T$ y $F : T \rightarrow S$, y sus adversarios discriminadores D_T y D_S respectivamente. (b) y (c) muestran la pérdida de consistencia cíclica calculada en ambas direcciones.

Formalmente, el objetivo del entrenamiento de la Cycle-GAN es aprender funciones de mapeo entre dos dominios S y T dadas las instancias de entrenamiento $\{x_{S_i}\}_{i=1}^N$ donde $x_{S_i} \in S$, y $\{x_{T_i}\}_{i=1}^N$ donde $x_{T_i} \in T$. Se denota a la distribución de los datos como $x_S \sim p_{data}(x_S)$ y $x_T \sim p_{data}(x_T)$.

El modelo de CycleGAN incluye dos funciones de transformación/generación $G : S \rightarrow T$ y $F : T \rightarrow S$, como se ilustra en la figura 3.3(a).

En contraposición a estos “generadores”, se tienen dos modelos “discriminadores” D_S y D_T . El objetivo de D_S es distinguir entre imágenes $\{x_S\}$ y $\{F(x_T)\}$, y, análogamente, D_T discrimina entre imágenes $\{x_T\}$ y $\{G(x_S)\}$.

La evaluación de las redes generadoras a realizar durante el entrenamiento consta de tres componentes:

1. **La función de pérdida adversaria**, para la cual D_S y D_T evalúan que se correspondan las imágenes transformadas con las reales del dominio opuesto [50].
2. **La función de pérdida de consistencia cíclica**, donde se penalizan las discrepancias entre cada x_{T_i} y $G(F(x_{T_i}))$, y análogamente entre cada x_{S_i} y $F(G(x_{S_i}))$. Representado en la figura 3.3 (b) y (c).
3. **La función de pérdida de mapeo de identidad**, a través de la cual se regularizan las funciones generadoras para que se asemejen lo más posible a la función identidad. De esta manera, las transformaciones realizadas son mínimas. [9]

En las siguientes secciones se explora más a fondo cada uno de estos tres términos y se examina su sinergia tanto en teoría como en casos específicos.

3.5.1. Función de pérdida adversaria

Se calcula la función de pérdida adversaria para ambas funciones generadoras. Para $G : S \rightarrow T$ y su discriminador D_T , el objetivo se expresa de la forma:

$$\begin{aligned}\mathcal{L}_{GAN}(G, D_T, S, T) = & \mathbb{E}_{x_T \sim p_{data}(x_T)} [\log(D_T(x_T))] \\ & + \mathbb{E}_{x_S \sim p_{data}(x_S)} [\log(1 - D_T(G(x_S)))]\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{GAN}(F, D_S, T, S) = & \mathbb{E}_{x_S \sim p_{data}(x_S)} [\log(D_S(x_S))] \\ & + \mathbb{E}_{x_T \sim p_{data}(x_T)} [\log(1 - D_S(F(x_T)))]\end{aligned}$$

donde G genera imágenes $G(x_S)$ visualmente similares a las imágenes del dominio T , mientras que D_T aprende a distinguir entre las instancias traducidas $G(x_{S_i})$ e instancias reales x_{T_i} .

G aprende a minimizar este objetivo contra su adversario D_T , que intenta maximizarlo. También calcula de manera análoga la función de pérdida adversaria para la función $F : T \rightarrow S$ y su correspondiente discriminadora D_S . [9]

3.5.2. Función de pérdida de consistencia cíclica

El entrenamiento regido por la función de pérdida adversaria puede, en teoría, construir las funciones de mapeo G y F que producen transformaciones idénticamente distribuidas a los dominios objetivo T y S respectivamente. Sin embargo, una estrategia que pueden adoptar las redes generadoras es transformar las entradas en imágenes arbitrarias del dominio objetivo, minimizando la pérdida adversaria pero fallando como traductores de dominio. Es por esto que la pérdida adversaria sola no puede garantizar que las traducciones produzcan la salida deseada, y con esta motivación se agrega el requisito de que las funciones de transformación deben tener “consistencia cíclica”.

Para comprender la idea intuitiva detrás de consistencia cíclica se puede considerar la siguiente analogía: al traducir una frase del español al inglés, y luego de vuelta al español, se debería obtener una frase muy similar a la original. En este ejemplo, la frase sería la instancia, y cada idioma un dominio distinto.

Formalmente, si se dispone de un traductor $G : S \rightarrow T$, y otro $F : T \rightarrow S$, entonces F y G deben ser inversos el uno del otro, y ambas funciones deben ser biyectivas. Se aplica esta suposición entrenando ambos mapeos a la vez, y agregando una función de pérdida de consistencia cíclica [51] que incentiva a las redes a cumplir $F(G(x_{S_i})) \approx x_{S_i}$ y $G(F(x_{T_i})) \approx x_{T_i}$ [9].

Este comportamiento de la GAN se rige mediante la siguiente función de pérdida de consistencia cíclica:

$$\begin{aligned}\mathcal{L}_{cyc}(G, F) = & \mathbb{E}_{x_S \sim p_{data}(x_S)} [\|F(G(x_S)) - x_S\|_1] \\ & + \mathbb{E}_{x_T \sim p_{data}(x_T)} [\|G(F(x_T)) - x_T\|_1]\end{aligned}$$

De esta manera, la función de pérdida total es:

$$\begin{aligned}\mathcal{L}(G, F, D_S, D_T) &= \mathcal{L}_{GAN}(G, D_T, S, T) \\ &+ \mathcal{L}_{GAN}(F, D_S, T, S) \\ &+ \lambda \mathcal{L}_{cyc}(G, F)\end{aligned}$$

Donde los primeros dos términos corresponden a la pérdida adversaria en ambas direcciones de la traducción y el tercero a la consistencia cíclica.

Para variar la importancia de una pérdida con respecto a la otra se agrega una constante λ que pondera la relación. En los experimentos preliminares del presente trabajo, una elección correcta de λ demostró ser imprescindible para un buen entrenamiento de las traducciones, y resulta interesante el balance que le brinda a la ecuación.

Si se le asigna a λ un valor muy alto, la red prioriza una buena reconstrucción de las imágenes, buscando que ocurran $F(G(x_S)) = x_S$ y $G(F(x_T)) = x_T$, desestimando el feedback de los discriminadores. El fenómeno de traducción entre los dominios $S =$ Caballos y $T =$ Cebras se representa en la figura 3.4 para el caso de un λ demasiado alto: se verifica que el modelo realiza transformaciones mínimas para poder reconstruir la imagen original de la mejor manera. Se observa que la transformación consiste en invertir los colores, lo cual acerca más al caballo a parecerse a una cebra, aunque no genera ningún patrón de rayas característico del dominio de cebras.

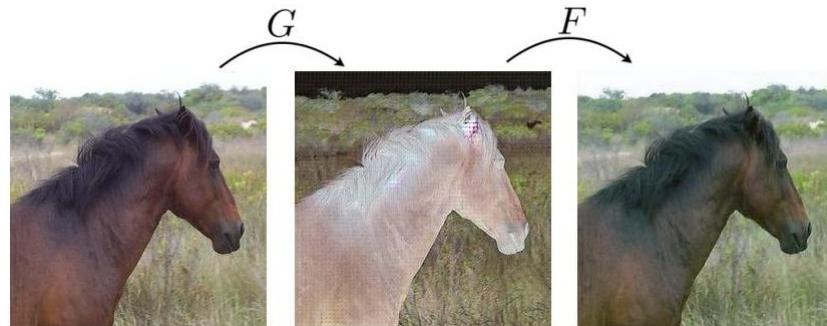


Fig. 3.4: Traducción 2D entre caballos y cebras con λ demasiado alto. G transforma de manera mínima y en consecuencia F reconstruye casi perfectamente con total facilidad.

En contraposición, al elegir un λ muy bajo ocurre lo contrario: la red generadora busca a toda costa engañar a los discriminadores, ignorando el feedback de la pérdida de consistencia cíclica. En los experimentos se observó de manera práctica que un λ excesivamente bajo genera transformaciones muy exageradas que logran engañar a los discriminadores, pero de las cuales es muy difícil reconstruir la imagen original.

Como se puede apreciar en el ejemplo de la Figura 3.5, las generadoras producen patrones de cebras aunque terminen deformando la imagen y en consecuencia las reconstrucciones resultan borrosas o incorrectas debido a que la imagen que deben reconstruir no tiene realmente aspecto de cebra. O sea, al agregar patrones de rayas a la imagen, G no genera realmente una instancia que parezca del dominio objetivo, sino que únicamente logra engañar al discriminador D_T , lo cual resulta ser lo más importante durante el entrenamiento al haber un λ tan bajo.

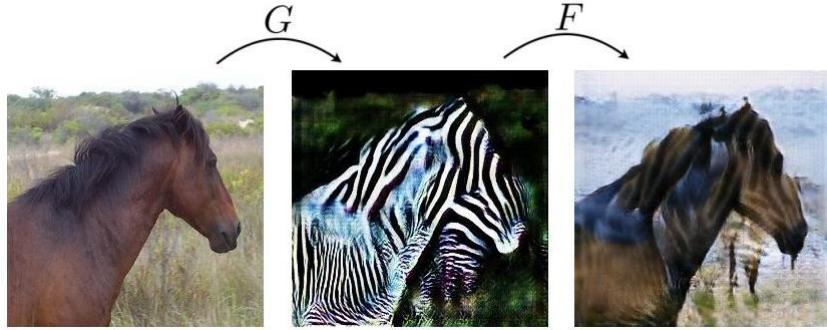


Fig. 3.5: Traducción 2D entre caballos y cebras con λ demasiado bajo. G se excede y F no puede reconstruir $G(x_S)$ al dominio de caballos partiendo de una “cebra” tan deformada.

Se puede notar un buen balance en la Figura 3.6 producto de un λ adecuado, y puede observarse que no ocurren transformaciones ni muy ínfimas (como con λ alto) ni muy exageradas (como con λ bajo), sino que son moderadas y producen una imagen que sí parece visualmente del dominio de las cebras.

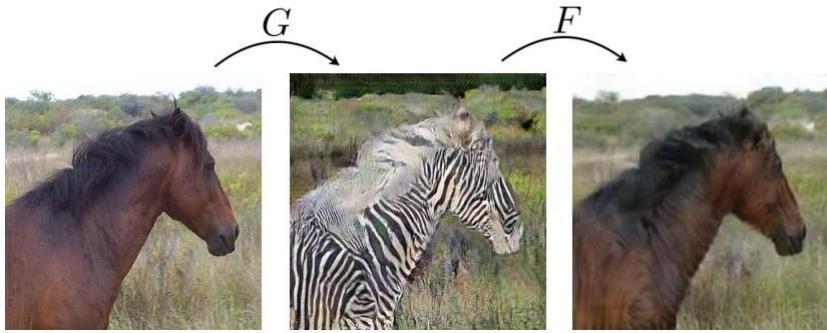


Fig. 3.6: Traducción 2D entre caballos y cebras con λ adecuado. G produce una transformación balanceada, y F logra reconstruirla de forma aceptable.

3.5.3. Función de pérdida de mapeo de identidad

Los experimentos preliminares de caballos y cebras mostraron que las dos funciones de pérdida ya definidas son suficientes para lograr una buena traducción entre dichos dominios. Sin embargo, al embarcarse en el área de las MRI de cerebros, se notó inmediatamente la necesidad de una tercera función de pérdida, marcada como opcional en ciertos trabajos actuales [9].

Al aplicar la fórmula utilizada para caballos y cebras sobre MRIs cerebrales, se obtuvo el comportamiento reflejado en la Figura 3.7. Dicho comportamiento corresponde a una transformación que engaña a los discriminadores (satisfaciendo la pérdida adversaria) y reconstruyendo la imagen original de manera adecuada (satisfaciendo la constancia cíclica) pero con transformaciones exageradas. Estas exageraciones arruinan el uso de Cycle-GAN como herramienta para adaptación de dominio, ya que modifican la forma original del cerebro que se quiere segmentar.

Para evitar estas transformaciones no deseadas se adopta la técnica de Taigman et al. [52], que regulariza a los generadores acercándolos al mapeo identidad cuando se les pasa como entrada instancias reales del dominio objetivo. En este trabajo se eligió una

frecuencia de 4:1, es decir, por cada 4 elementos del dominio origen que se alimente a la red generadora, se alimenta con uno del dominio objetivo, y de esta manera la generación se acerca más a la identidad. Esto aplica a ambos generadores de manera simétrica.

Formalmente, la función de pérdida de mapeo de identidad se define de la siguiente manera:

$$\begin{aligned}\mathcal{L}_{id}(G, F) = & \mathbb{E}_{x_S \sim p_{data}(x_S)} [\|F(x_S) - x_S\|_1] \\ & + \mathbb{E}_{x_T \sim p_{data}(x_T)} [\|G(x_T) - x_T\|_1]\end{aligned}$$

Un ejemplo de la necesidad y consecuencia de aplicar regularización de mapeo de identidad se puede apreciar en la Figura 3.7. Ante la misma entrada 3.7(a), el modelo sin la regularización propuesta aplica transformaciones que quizás engañen al discriminador y respeten la consistencia cíclica, pero también deforman la imagen cerebral 3.7(b). Por otro lado, aplicar la regularización implica una generación más parecida a una identidad, aplicando transformaciones mínimas para conservar la forma original del cerebro y donde los cambios permiten “engaños” al discriminador y respetar la consistencia cíclica 3.7(c).

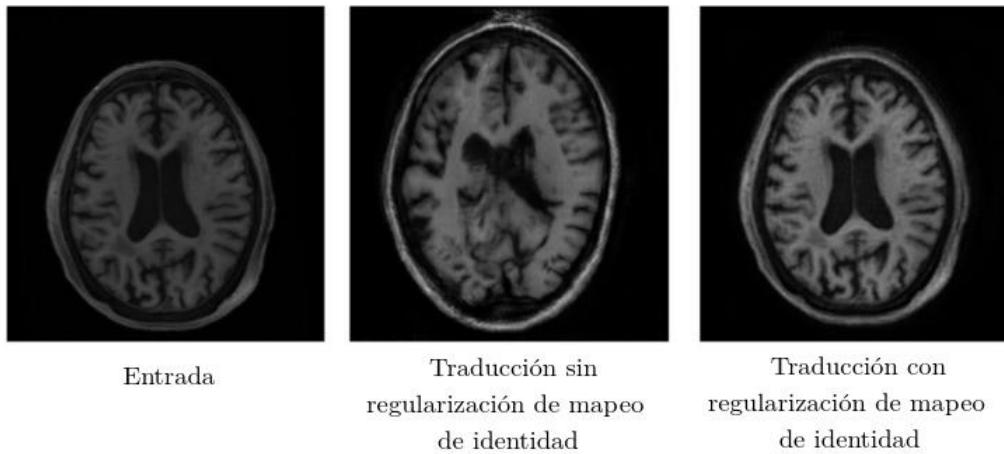


Fig. 3.7: Comparación de la misma MRI 2D transformada en dos modelos con y sin regularización de mapeo de identidad.

3.6. Arquitectura de CycleGAN

3.6.1. Arquitectura de la red generadora

Para las redes generadoras, CycleGAN utiliza la arquitectura propuesta por Johnson et al. [53], debido a sus notables resultados para transferencia de estilos. La arquitectura consiste en 2 capas convolucionales, 9 bloques residuales y 2 capas de convolución transpuesta; considerando que estas últimas pueden ser implementadas de dos maneras (lo cual se explica en la Sección 4.3.3).

La función de normalización utilizada en la red es “Instance Normalization” [54] en reemplazo de la estándar “Batch Normalization”, debido a sus resultados positivos en la estilización de imágenes [55]. Como función de activación se utiliza ReLU [56] a lo largo de la red generadora.

Resulta importante destacar que la arquitectura generadora es puramente convolucional, lo cual implica que su entrada no tiene tamaño fijo. Esto permite realizar el entrenamiento con parches y la evaluación con imágenes completas. La Sección 2.4.2 profundiza sobre redes puramente convolucionales y el entrenamiento con parches.

3.6.2. Arquitectura de la red discriminadora

La red discriminadora utiliza la arquitectura PatchGAN propuesta en [57], la cual consiste en 4 capas convolucionales de tamaño de kernel 64, 128, 256 y 512, seguidas de una convolución para producir una salida unidimensional. Se aplica “InstanceNormalization” luego de todas las capas menos la primera, la cual no tiene normalización. Como función de activación se utiliza leaky ReLU [58] con 0.2 de pendiente.

4. METODOLOGÍA DE TRABAJO PARA LA EXPERIMENTACIÓN

En este trabajo se propone el desarrollo de un modelo de adaptación de dominio basado en el uso de CycleGAN. En esta sección se cubre la elección de los elementos que componen el diseño experimental, incluyendo hardware, librerías, red segmentadora, estrategias de muestreo, implementación de CycleGAN y datasets de los distintos dominios a adaptar.

4.1. Librerías y hardware utilizado

Todos los modelos, experimentos y herramientas utilizados en este trabajo fueron desarrollados en Python 3, utilizando Keras sobre TensorFlow 2.0 [59], mediante la imagen pública de Docker provista por el equipo de TensorFlow.

Como herramienta de visualización de las MRI 3D se utilizó ITK-SNAP[60], la cual permitió apreciar visualmente las entradas y salidas 3D de los modelos, con las etiquetas sobre la imagen y con la posibilidad de comparar las imágenes de manera sincronizada.

Se aprovechó el uso de GPU para acelerar los procesos de entrenamiento y evaluación, ya que de otro modo los experimentos habrían tardado semanas en lugar de días. La computadora con la cual fueron implementados y ejecutados todos los experimentos consta de las siguientes características:

- **Motherboard:** ASUS Z170-P.
- **CPU:** Intel Core i5-6500 CPU 3.20GHz.
- **RAM:** 24GB DDR4
- **Sistema operativo:** Ubuntu 18.04
- **GPU:** RTX 2060 con 40 tensor cores y 6GB RAM GDDR6. Drivers CUDA 10.

El rendimiento utilizando el hardware mencionado fue el siguiente:

- El tiempo requerido para el entrenamiento del modelo segmentador con un dataset de 17 imágenes de resonancia fue de 2 horas aproximadamente.
- El tiempo requerido para el entrenamiento de las redes adversarias en 2D fue cercano a las 24 horas.
- El tiempo requerido para el entrenamiento de las redes adversarias en 3D fue cercano a las 48 horas.

4.2. Características del dataset

Se trabajó con el dataset provisto por el WMH Segmentation Challenge, presentado en MICCAI 2017[8]. Se eligieron estos datos ya que son ideales para el objetivo propuesto: se dividen en dos dominios (definidos por el centro médico de captura), con 20 imágenes cada uno y con distinta configuración de parámetros de adquisición. Se nota que se preservan entre estos dos dominios dos aspectos importantes:

- Toda MRI consiste en una imagen de modalidad T1 y otra de modalidad FLAIR, las cuales se corresponden punto a punto. De esta manera, la red puede considerarla como una misma imagen con dos canales, combinando la información que proveen ambas para una mejor representación del cerebro.
- Toda MRI tiene el mismo tamaño de voxel en mm^3 . Esto es fundamental para limitar la dificultad del problema de adaptación de dominio.

Los dos dominios incluidos en el dataset se diferencian por el centro médico y el equipo utilizado para capturar las imágenes. Las características específicas de los dos dominios son:

- **University Medical Center, Utrecht – Máquina 3T Philips Achieva.**
 - Secuencia T1 (192 cortes, tamaño de voxel: $1,00 \times 1,00 \times 1,00 mm^3$, TR/TE: 7.9/4.5 ms).
 - Secuencia FLAIR (48 cortes transversales, tamaño del voxel: $0,96 \times 0,95 \times 3,00 mm^3$, TR/TE/TI: 11000/125/2800 ms)
- **National University Health System, Singapur – Máquina 3T Siemens TrioTim.**
 - Secuencia T1 (tamaño de voxel: $1,00 \times 1,00 \times 1,00 mm^3$, TR/TE/TI: 2300/1.9/900 ms).
 - Secuencia FLAIR (48 cortes transversales, tamaño de voxel: $1,0 \times 1,0 \times 3,00 mm^3$, TR/TE/TI: 9000/82/2500 ms)

4.3. Implementaciones utilizadas de CycleGAN

La implementación de CycleGAN del presente trabajo requirió ciertas modificaciones para ser adaptada a la tarea en cuestión. En primer lugar, se decidió remover la capa de Activación ReLU final de los generadores para no limitar el rango de salida de la misma y de esa forma poder generar imágenes en cualquier rango de colores posible. Además, se implementó la convolución traspuesta de una manera alternativa a la capa original de Keras debido a un defecto en la reconstrucción, lo cual se explica en detalle en la Sección 4.3.3.

Por último, se implementaron dos versiones alternativas de CycleGAN: en 2D (original) y en 3D, las cuales se explican en las secciones 4.3.1 y 4.3.2 respectivamente. Cada una requirió distintos cambios en la arquitectura, proceso de entrenamiento, evaluación y muestreo de datos.

4.3.1. Implementación 2D de CycleGAN

Con la implementación original de CycleGAN 2D no es posible transformar MRIs tridimensionales. El enfoque elegido para abordar esta problemática consistió en dividir la imagen 3D en cortes axiales 2D de la MRI (ver Figura 2.5). Luego, transformar cada corte individualmente utilizando CycleGAN 2D, y finalmente unir los cortes transformados. Este proceso se esquematiza en la Figura 4.1.

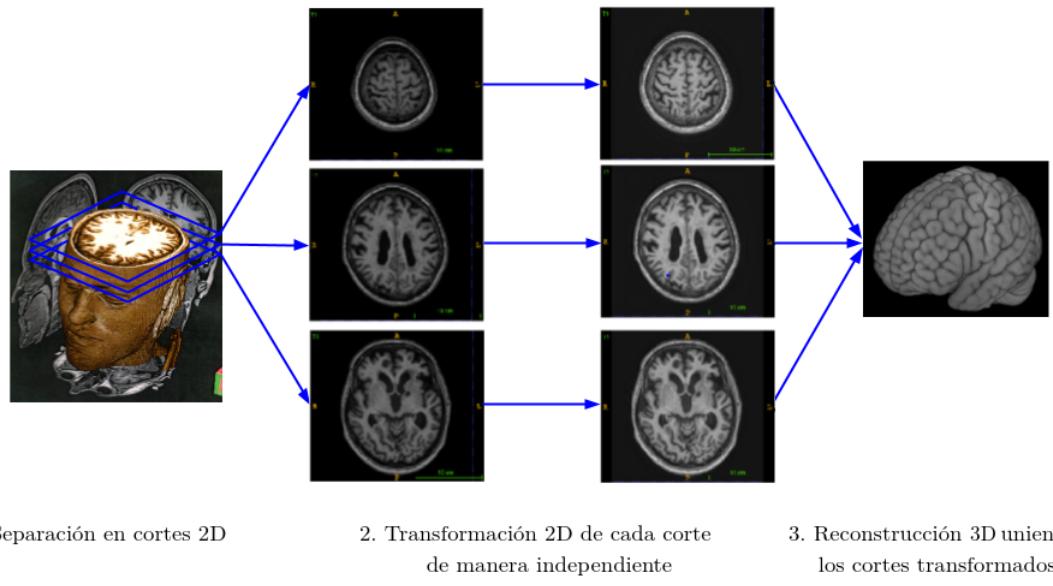


Fig. 4.1: Esquema de transformación 3D mediante CycleGAN 2D. En el dataset las MRI consisten en 48 cortes axiales, mientras que en este esquema se muestran solo 3 cortes por simplicidad.

Dado que los cortes axiales del cerebro tienen distintos tamaños, estructura general y patrones internos, el entrenamiento de las redes generadoras con cortes extraídos de manera indiscriminada provoca que la red no atine a realizar transformaciones que correspondan a la imagen de entrada.

La solución adoptada en el presente trabajo consiste en una variación de **curriculum learning**, propuesto por Yoshua Bengio et al. [61]. Dicho trabajo propone comenzar el entrenamiento de una red neuronal con ejemplos simples para luego ir avanzando hacia instancias cada vez más complicadas durante el transcurso de las épocas de aprendizaje (ver ejemplos en la Figura 4.2).

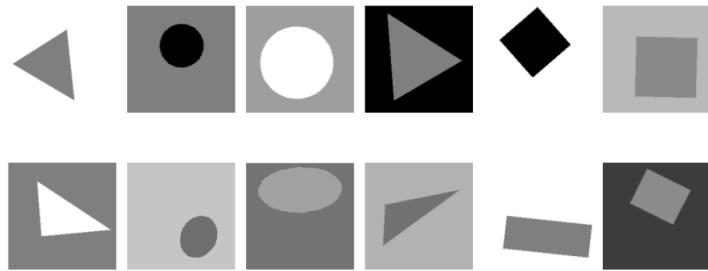


Fig. 4.2: Ejemplo de currículum learning extraído del trabajo de Bengio et al. [61]. Arriba, instancias simples de menor variabilidad utilizadas al principio del entrenamiento. Abajo, instancias más complejas y variables, utilizadas para entrenar al final del proceso.

Siguiendo esta idea, se entrenó Cycle-GAN 2D con instancias extraídas de los cortes axiales centrales de las MRIs, avanzando hacia los extremos durante el progreso del entre-

namiento. Esto produjo mejores resultados y, sumado al uso de regularización de mapeo de identidad (ver Sección 3.5.3), se pudo llegar a la traducción deseada en 2D.

4.3.2. Implementación 3D de CycleGAN

Para adaptar la arquitectura 2D CycleGAN original a 3D fue necesario reemplazar cada componente 2D por su equivalente tridimensional. Estas componentes son la capas de Convolución, UpSampling, Padding y Convolución Transpuesta.

La implementación 3D tiene la ventaja de no requerir ningún tipo de procesamiento extra en la etapa de predicción para trabajar sobre toda la imagen volumétrica. La desventaja reside en que no resulta posible utilizar las MRIs tridimensionales en su totalidad en la etapa de entrenamiento, debido a que esto requiere una gran capacidad de memoria de GPU, de la cual no se dispuso (ver hardware utilizado en la Sección 4.1). Por tanto, se hizo necesario utilizar parches de menor dimensión como instancias de entrenamiento. Esta práctica común también fue utilizada al entrenar las redes segmentadoras del presente trabajo, y se describe en la Sección 4.4.

Se eligió como tamaño de parches ($48 \times 48 \times 48 \times 2$), siendo dicho tamaño el máximo posible para la memoria de la GPU utilizada. El muestreo de parches trajo consigo una nueva problemática acerca de la política de extracción de dichos parches 3D de las imágenes. En experimentos preliminares se evaluaron distintas políticas de extracción, las cuales consistieron en requerir que determinado porcentaje central de la MRI esté dentro de la máscara del cerebro producida por ROBEX [62]. Finalmente, la política con mejor rendimiento resultó ser exigir únicamente que el voxel central pertenezca a dicha máscara del cerebro (al igual que el muestreo de las redes segmentadoras descrito en la Sección 4.4).

4.3.3. Implementación de la capa de convolución transpuesta

Los primeros resultados de la generación de MRIs no fueron satisfactorios debido a un “artifact” (defecto en imagen) presente en todas las imágenes sintéticas, el cual consiste en un efecto “cuadriculado” que cubre gran parte de la imagen. Un ejemplo puede observarse en la Figura 4.3. Este defecto es común en la generación de imágenes mediante redes neuronales profundas, y se lo denomina “checkerboard artifact” (artifact de tablero). A continuación, se explica en profundidad la causa de dicho artifact y la solución propuesta en este trabajo.

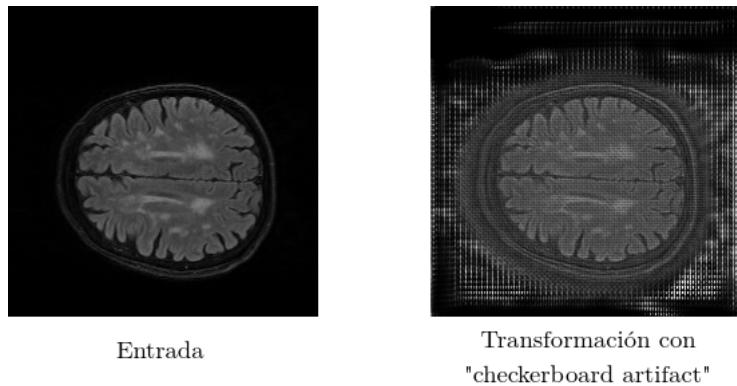


Fig. 4.3: Ejemplo del “checkerboard artifact” en una MRI cerebral generada con CycleGAN.

A partir de una imagen de entrada, la red generadora produce descripciones de alto nivel y baja resolución a través del “downsampling”. Esto sirve de boceto para luego ser expandido en una imagen de alta resolución llenando detalles mediante convoluciones transpuestas, lo cual se denomina “upsampling” (ver Sección 2.5.1).

Sin embargo, la convolución transpuesta puede producir cierto solapamiento en la salida, lo cual provoca que la salida de dos véxeles interseque, y dicha intersección se ve “pintada” más de una vez. En particular, la convolución transpuesta produce solapamiento cuando el tamaño de kernel no es divisible por el “stride” (los saltos que hace el filtro) [63]. Esto se ve reflejado en el caso unidimensional en la Figura 4.4.

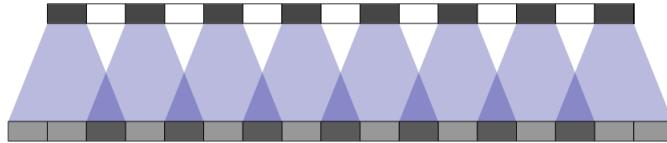


Fig. 4.4: Visualización de la generación del “checkerboard artifact” en una dimensión. Corresponde al caso de la generadora de CycleGAN, que tiene tamaño de kernel 3 y stride 2.

Existen diversas soluciones posibles para esta problemática. La más sencilla es modificar las capas para que el stride sea divisible por el kernel size.

Sin embargo, con el objetivo de preservar la arquitectura original de CycleGAN lo más posible, en el presente trabajo se optó por reemplazar las funciones estándar de convolución transpuesta por una interpolación seguida de una convolución. Más específicamente, se realizó una interpolación de vecino más cercano de (2×2) en 2D, $(2 \times 2 \times 2)$ en 3D, seguida de una convolución de tamaño de kernel 3 y stride 1. En otras palabras, se duplicaron los véxeles y se dividió por 2 el stride. De esta manera se evitó el checkerboard artifact, a cambio de una pequeña pérdida de precisión en la imagen generada.

4.4. Red segmentadora y su muestreo de datos

Las MRI volumétricas de dos canales (T1 y FLAIR) son codificadas en matrices de dimensiones $256 \times 256 \times 48 \times 2$. Estas dimensiones exigen mucha memoria de GPU para poder entrenar con las imágenes enteras, por lo que se implementó un entrenamiento por parches. Dicha técnica consiste en entrenar la red con parches de menor dimensión y luego

evaluar sobre las imágenes completas. Por este motivo se eligió como red segmentadora a U-Net en su versión 3D, la cual es puramente convolucional y por lo tanto no exige un tamaño de entrada fijo (lo cual se explica en la Sección 2.4.2). Se eligió como tamaño de parche de entrenamiento ($32 \times 32 \times 32 \times 2$).

Para tener la seguridad de que los parches tengan buena cantidad de información para la red, se tomaron únicamente aquellos con voxel central dentro del cerebro (y no en el cráneo o fuera de la cabeza). Para hacer este chequeo se utilizó la herramienta ROBEX (RObust Brain EXtraction) [62].

Además, se tuvo que compensar el hecho de que hay una enorme cantidad de voxels sanos en comparación con los de lesión. Se optó por entrenar la red con una cantidad equitativa de parches centrados en un voxel sano y parches centrados en un voxel con lesión WMH. Los pasos a seguir entonces fueron:

1. Extraer todo parche posible de las imágenes de MRI con voxel central ubicado dentro del cerebro. Para este chequeo se utiliza la máscara obtenida con ROBEX.
2. Separar estos parches en dos conjuntos: uno de aquellos centrados en un voxel sano y otro de aquellos centrados en un voxel con lesión WMH.
3. Remover elementos de manera aleatoria del conjunto de parches sanos para que tenga el mismo tamaño que el de parches con lesión.
4. Mezclar ambos conjuntos aleatoriamente, produciendo así un conjunto de datos de entrenamiento balanceado.

4.5. Validación cruzada y división de datos

Validación cruzada (k -fold cross-validation) es un procedimiento de muestreo utilizado para evaluar cualquier modelo de aprendizaje automático en un dominio acotado. Su objetivo es estimar el rendimiento del modelo sobre datos nunca antes vistos por el mismo. Generalmente resulta en una estimación menos sesgada y optimista que otros métodos de muestreo, como una simple división de datos de entrenamiento y evaluación. El único parámetro que requiere es un valor constante k , que define la cantidad de grupos en los que se dividen los datos.

El procedimiento en general consiste en los siguientes pasos:

1. Desordenar el dataset D de manera aleatoria.
2. Dividir el dataset D en k grupos.
3. Por cada grupo g :
 - a) Tomar g como datos de hold-out.
 - b) Tomar el resto de los grupos $(D - g)$ como datos de entrenamiento.
 - c) Guardar el resultado del modelo entrenado con $(D - g)$ evaluado sobre g .
4. Promediar los k resultados.

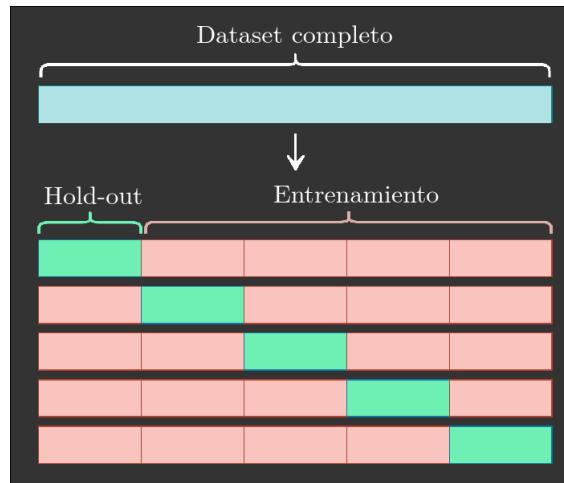


Fig. 4.5: Esquema de ejemplo de k-fold con $k=5$.

Es importante notar que cada instancia de los datos es asignada a un grupo únicamente y se preserva dicha asignación a lo largo del procedimiento. Esto significa que cada instancia tiene la oportunidad de ser usada como parte del hold-out 1 vez, y es utilizada para entrenamiento $k-1$ veces.

En este trabajo se eligió $k = 7$ debido a que un k menor implica una cantidad muy escasa de imágenes de entrenamiento para la red segmentadora. Considerando que los datasets tienen 20 imágenes, fijando $k = 7$ se tienen 17 imágenes de entrenamiento y 3 de hold-out en cada iteración.

De manera independiente al k-fold, dentro de cada conjunto de entrenamiento los datos se subdividen en entrenamiento (80 % de los datos) y validación (el 20 % restante).

4.6. Normalización

Se aplicó a las MRIs la técnica de normalización por z-scores, lo cual es una práctica común en el procesamiento de este tipo de imágenes. A cada voxel se le resta la media de la intensidad de los voxels de su MRI y se la divide por el desvío estándar de la misma.

$$z_i = \frac{x_i - \mu}{\sigma} \quad (4.1)$$

4.7. Data augmentation

Las redes neuronales profundas requieren una gran cantidad de datos para su entrenamiento. Por ello, se intentó realizar “data augmentation” (aumentación de datos) al entrenar la red segmentadora. Se probaron múltiples técnicas, tales como duplicar imágenes de manera espejada y agregar ruido gaussiano de leve intensidad. Sin embargo, no se notaron mejoras significativas durante su uso, probablemente debido a que la cantidad de parches 3D utilizados para entrenar es suficientemente grande y representativa de la distribución de los datos. Por ello, no se realizó aumentación de datos para la red segmentadora en los experimentos del presente trabajo.

4.8. Métricas

4.8.1. Coeficiente de Dice

La métrica utilizada para evaluar la calidad de las segmentaciones es el coeficiente de Sørensen–Dice. También es conocido como “coeficiente de similaridad”, ya que mide la similaridad entre dos conjuntos X e Y. Se define de la forma:

$$Dice = \frac{2|X \cap Y|}{|X| + |Y|} \quad (4.2)$$

Su aplicación como métrica para segmentación de imágenes corresponde a pensar X como el Ground Truth, e Y como las etiquetas predichas. De esta manera, se calcula la similaridad entre el output del modelo y el resultado esperado. La intuición matemática es que, si ambos conjuntos son idénticos, el coeficiente es igual a 1.0, mientras que si no tienen ningún elemento en común el coeficiente resulta en 0.0.

Como los modelos segmentadores asignan probabilidad de pertenencia de cada clase a los vértices, existen dos maneras de calcular el coeficiente de Dice:

- Asignando una clase a cada vértice en base a su probabilidad, para luego calcular Dice (se denomina **Hard Dice**). Resulta ser una medida más estricta, por lo que se la utilizó para evaluar los modelos segmentadores ya entrenados.
- Calculando Dice en base a las probabilidades asignadas por el modelo (se denomina **Soft Dice**). Se utilizó como métrica de entrenamiento de los modelos segmentadores.

Se nota como $Dice(f_A(x_i))$ al coeficiente de Dice resultante de evaluar un modelo entrenado sobre el dominio A sobre una instancia x_i . Por consiguiente, se nota como $Dice(f_A(B))$ al conjunto de coeficientes de Dice correspondientes a la evaluación de un modelo entrenado sobre el dominio A sobre los elementos de otro dominio $B = \{x_{B_1} \dots x_{B_n}\}$. Formalmente:

$$Dice(f_A(B)) = \{Dice(f_A(x_{B_i})) : 1 \leq i \leq n\} \quad (4.3)$$

4.8.2. Divergencia de Jensen–Shannon

Se utilizó una variación de la divergencia de Jensen-Shannon para cuantificar la diferencia de dominio dentro de un mismo dataset y entre pares de datasets.

La divergencia de Jensen-Shannon original es una métrica utilizada para medir la similitud entre dos distribuciones de probabilidad P y Q. Se calcula de la siguiente manera:

$$JS(P, Q) = \frac{H(P, M) + H(M, Q)}{2} \quad (4.4)$$

Donde H es la función de entropía, y $M = \frac{P+Q}{2}$.

La variación propuesta para el presente trabajo se denota Δ_{JS} , y parte de la idea de considerar los histogramas de las imágenes como distribuciones de probabilidad.

De esta manera, se calcula Δ_{JS} dentro de un mismo dataset $D = \{x_1, \dots, x_n\}$ como el promedio de las divergencias de Jensen-Shannon entre pares de imágenes distintas:

$$\Delta_{JS}(D) = \frac{1}{n(n-1)} \sum_{i=1, j=1, i \neq j}^n JS(hist(x_i), hist(x_j)) \quad (4.5)$$

donde $hist(x)$ es el histograma de las intensidades de los vóxeles de x . Por otro lado, la divergencia Δ_{JS} entre dos dominios distintos $S = \{x_{S_1}, \dots, x_{S_n}\}$ y $T = \{x_{T_1}, \dots, x_{T_m}\}$ se calcula como el promedio de la divergencia entre cada par de imágenes x_{S_i} y x_{T_j} :

$$\Delta_{JS}(S, T) = \frac{1}{n.m} \sum_{i=1}^n \sum_{j=1}^m JS(hist(x_{S_i}), hist(x_{T_j})) \quad (4.6)$$

Además, se nota al conjunto de las divergencias de un dominio $D = \{x_1, \dots, x_n\}$ como

$$\Delta_{JS}(D \times D) = \{JS(hist(x_i), hist(x_j)) : 1 \leq i < j \leq n\} \quad (4.7)$$

Por último, el conjunto de las divergencias entre cada par de elementos de los dominios $S = \{x_{S_1}, \dots, x_{S_n}\}$ y $T = \{x_{T_1}, \dots, x_{T_m}\}$ se nota $\Delta_{JS}(S \times T)$ y se calcula de la siguiente manera:

$$\Delta_{JS}(S \times T) = \{JS(hist(x_i), hist(x_j)) : 1 \leq i \leq n, 1 \leq j \leq m\} \quad (4.8)$$

5. RESULTADOS DE LA EXPERIMENTACIÓN

Teniendo en cuenta el objetivo exploratorio, se experimentó con CycleGAN para conocer su efectividad como herramienta de adaptación de dominio, utilizando los datasets extraídos de MICCAI 2017 (Ver Sección 4.2), que contienen 20 imágenes adquiridas en el centro médico UMC en Utrecht y otras 20 adquiridas en NUHS en Singapur. En esta sección se denominan simplemente como “Utrecht” (dominio U) y “Singapur” (dominio S). Como preprocesamiento en los experimentos, ambos dominios fueron normalizados por medio de Z-Scores, método estándar utilizado para reducir las diferencias entre imágenes (ver Sección 4.6).

Para analizar la efectividad de las funciones de mapeo entre dominios $G : U \rightarrow S$ y $F : S \rightarrow U$ se utilizaron dos estrategias: una directa y una indirecta:

- En la estrategia directa, se cuantificó la diferencia en las distribuciones de intensidad de las imágenes por medio de la divergencia de Jensen-Shannon Δ_{JS} (ver Sección 4.8.2), computada sobre los histogramas de intensidades de las imágenes. Este experimento es presentado en la Sección 5.1.
- En la estrategia indirecta, se optó por cuantificar la performance de un modelo de segmentación de lesiones entrenado en un dominio origen, al ser utilizado en el dominio destino transformado por la función de mapeo. Este experimento es presentado en la Sección 5.2, donde la performance del modelo es medida por medio del coeficiente de *Dice* (definido en la Sección 4.8.1).

Las modalidades T1 y FLAIR se trataron como dos canales de la misma MRI, por lo que las transformaciones se realizaron sobre ambos canales a la vez para cada imagen. Sin embargo, en cada experimento se analizó el impacto de la transformación de manera separada para cada modalidad. Esto se debe a la posibilidad de que las imágenes sean muy similares en una modalidad y muy distintas en la otra.

Para exponer la comparación entre conjuntos de resultados se utiliza el diagrama “Boxplot” [64], debido a que permite apreciar los cuartiles y outliers de cada conjunto, además de la media (notada con un rombo rojo). Mediante Boxplots es posible examinar la diferencia entre los conjuntos de resultados de manera más profunda que comparando sus promedios y de manera más simple que comparando todos los valores en una tabla.

En dichos Boxplots, se nota a los modelos generadores de las implementaciones de CycleGAN 2D y CycleGAN 3D (ver Sección 4.3) como:

- F_{2D} y G_{2D} a los generadores de CycleGAN 2D.
- F_{3D} y G_{3D} a los generadores de CycleGAN 3D.

5.1. Análisis por divergencia de Jensen-Shannon

Se determinó una cota inferior y una superior para la eficacia de F y se pretendió como objetivo conocer su ubicación entre dichas cotas. Para ello, se compararon los siguientes 4 conjuntos en la Sección 5.1.1:

- $\Delta_{JS}(S \times U)$. Son las divergencias inter-dominio entre origen y destino. Se la utilizó como cota superior (peor caso) de la eficacia de adaptación de dominio.
- $\Delta_{JS}(F_{2D}(S) \times U)$. Son las divergencias inter-dominio utilizando la herramienta de adaptación de dominio 2D. Mide la eficacia de adaptación de dominio mediante CycleGAN 2D.
- $\Delta_{JS}(F_{3D}(S) \times U)$. Son las divergencias inter-dominio utilizando la herramienta de adaptación de dominio 3D. Mide la eficacia de adaptación de dominio mediante CycleGAN 3D.
- $\Delta_{JS}(U \times U)$. Es la divergencia intra-dominio del dominio destino. Se la utilizó como cota inferior (mejor caso) de la eficacia de adaptación de dominio.

Para medir la eficacia de G , se compararon 4 conjuntos de manera análoga en la Sección 5.1.2:

- $\Delta_{JS}(S \times U)$. Es la divergencia inter-dominio entre origen y destino. Se la utilizó como cota superior (peor caso) de la eficacia de adaptación de dominio.
- $\Delta_{JS}(S \times G_{2D}(U))$. Mide la eficacia de adaptación de dominio mediante CycleGAN 2D.
- $\Delta_{JS}(S \times G_{3D}(U))$. Mide la eficacia de adaptación de dominio mediante CycleGAN 3D.
- $\Delta_{JS}(S \times S)$. Es la divergencia intra-dominio del destino. Se la utilizó como cota inferior (mejor caso) de la eficacia de adaptación de dominio.

5.1.1. Eficacia de F transformando el dominio Singapur

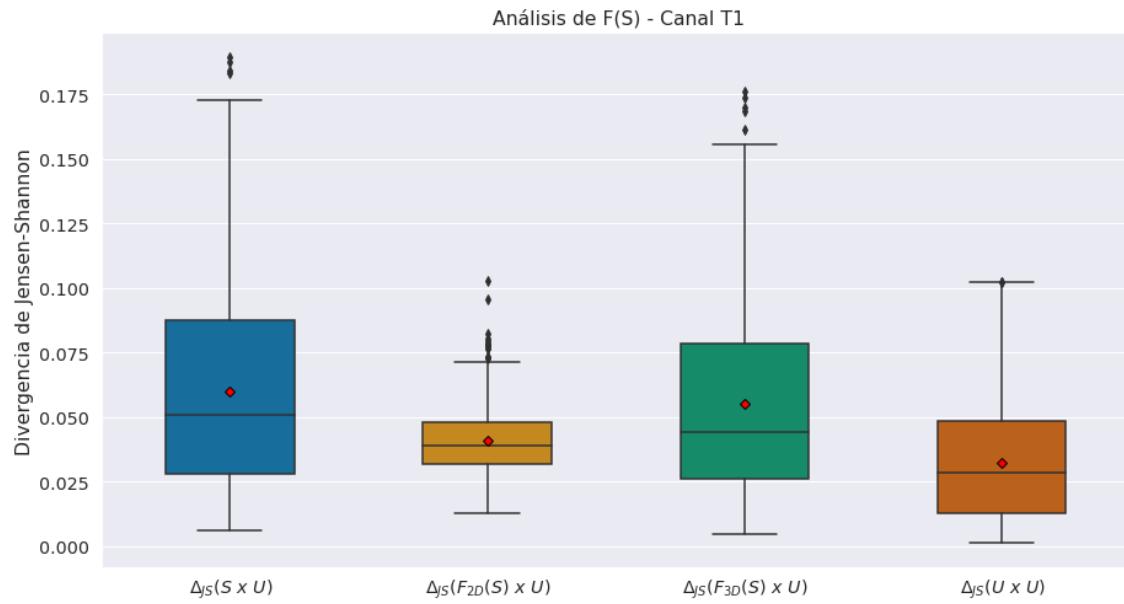


Fig. 5.1: Evaluación de la eficacia de $F : S \rightarrow U$, según divergencia de Jensen-Shannon Δ_{JS} en el canal T1.

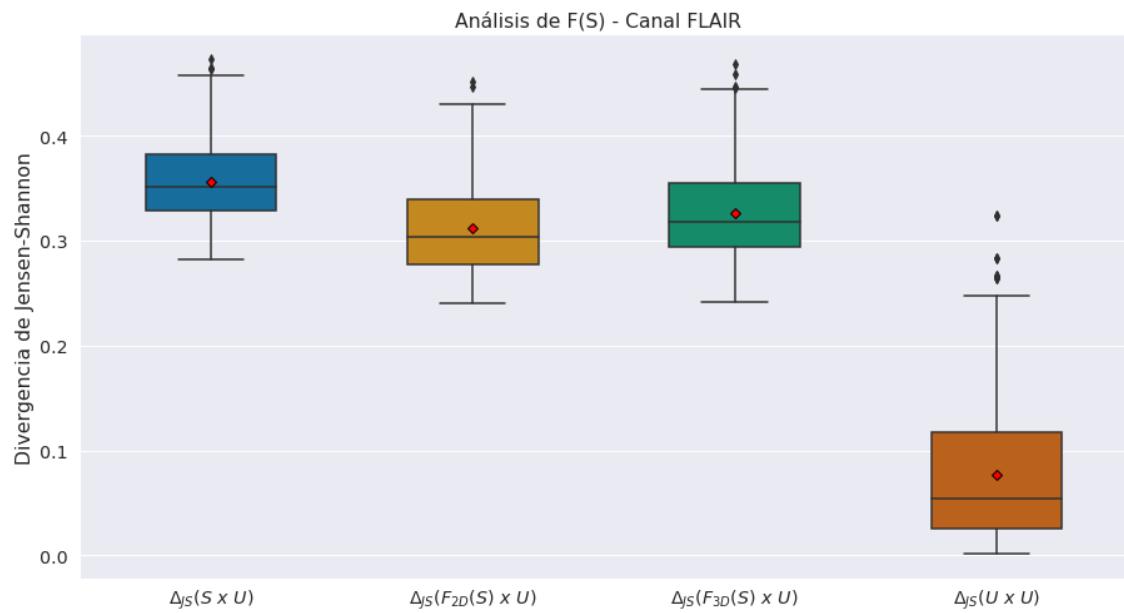


Fig. 5.2: Evaluación de la eficacia de $F : S \rightarrow U$, según divergencia de Jensen-Shannon Δ_{JS} en el canal FLAIR.

En el primer experimento exhibido en las Figuras 5.1 y 5.2 se verifica la diferencia de escalas entre la divergencia en las modalidades FLAIR y T1. Esto indica que la modalidad T1 es de mayor similaridad entre los dominios que la modalidad FLAIR.

En el caso de FLAIR, el baseline inferior da a entender que aún hay lugar para mejorar las transformaciones: sigue habiendo bastante divergencia entre los dominios F (Singapur) y Utrecht, en comparación con la divergencia dentro de Utrecht mismo.

Sin embargo, se aprecia cierta reducción de divergencia que ofrece CycleGAN al transformar las imágenes de Singapur, tanto para la implementación 2D como para 3D. Esta reducción se verificó a través del test de Wilcoxon, el cual determinó una diferencia significativa entre $\Delta_{JS}(F(S) \times U)$ y $\Delta_{JS}(U \times U)$, tanto para F_{2D} como para F_{3D} . El p-valor utilizado fue de 0.05.

5.1.2. Eficacia de G transformando el dominio Utrecht

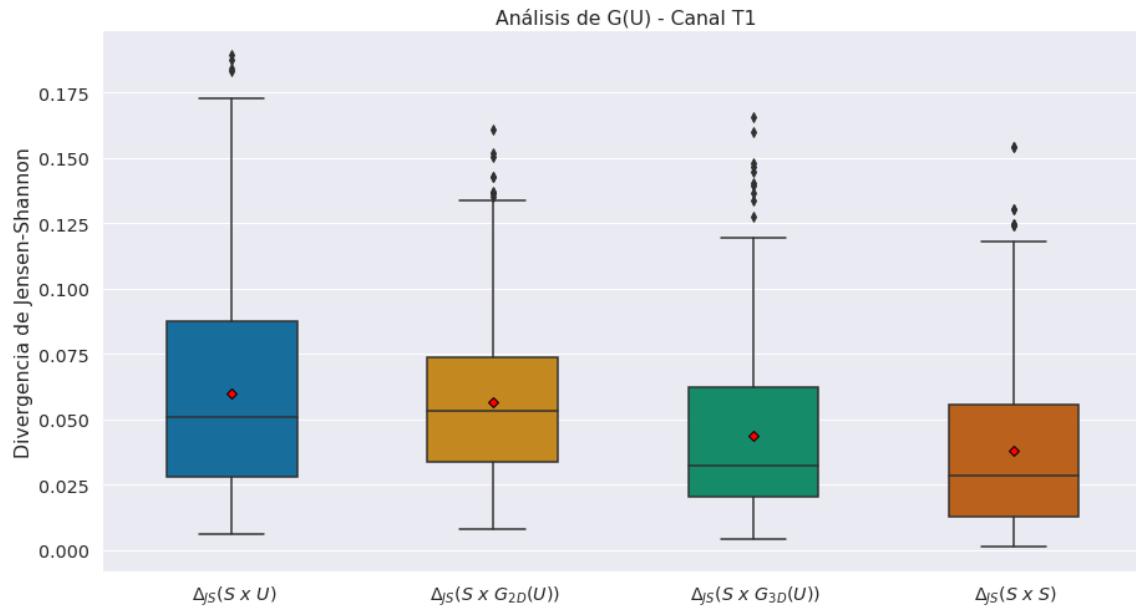


Fig. 5.3: Evaluación de la eficacia de $G : U \rightarrow S$, según divergencia de Jensen-Shannon Δ_{JS} en el canal T1.

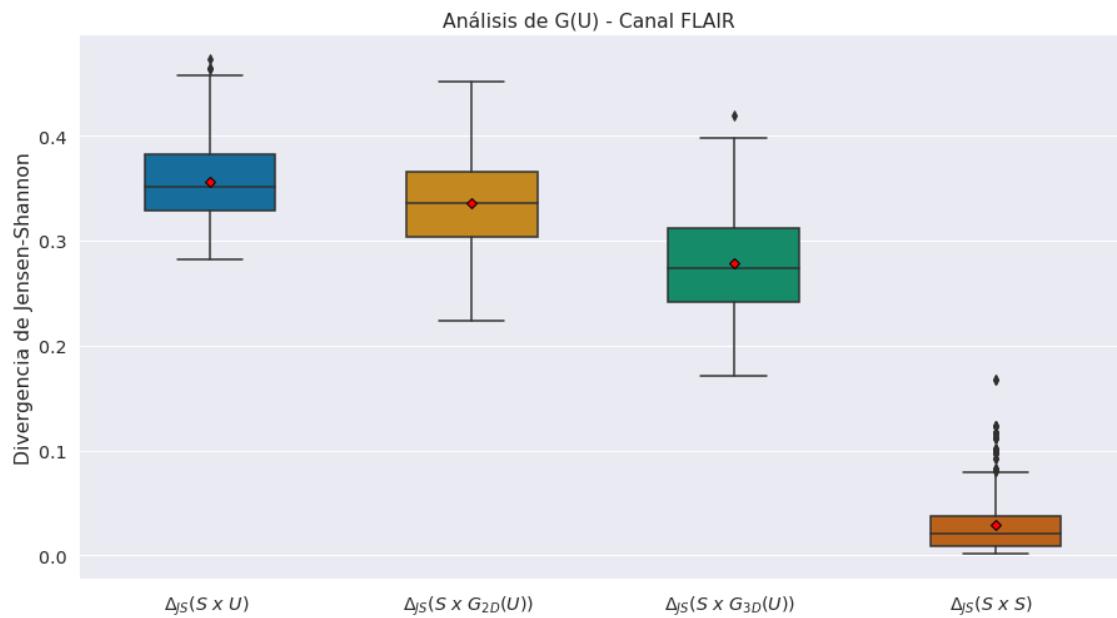


Fig. 5.4: Evaluación de la eficacia de $G : U \rightarrow S$, según divergencia de Jensen-Shannon Δ_{JS} en el canal FLAIR.

Al analizar $G : U \rightarrow S$ mediante divergencia de Jensen-Shannon se puede apreciar en las Figuras 5.3 y 5.4 una clara reducción de divergencia al utilizar CycleGAN 2D y 3D.

Este resultado visual se ve respaldado el test de Wilcoxon. A través del dicha prueba, se determinó una diferencia significativa entre los conjuntos $\Delta_{JS}(S \times G(U))$ y $\Delta_{JS}(S \times S)$, tanto para G_{2D} como para G_{3D} . El p-valor elegido fue de 0.05

5.2. Análisis por coeficiente de Dice

La mejor manera de evaluar la herramienta propuesta de adaptación de dominio es comprobar si efectivamente cumple con su objetivo.

Para ello, nuevamente se determinó una cota inferior y una superior a la eficacia de F , y se pretendió conocer su ubicación entre dichas cotas. Esta vez la eficacia fue medida con Coeficiente de Dice al segmentar el dominio objetivo. Con esta idea, se compararon los siguientes 4 conjuntos en la Sección 5.2.1:

- $Dice(f_S(U))$. Conjunto de coeficientes de *Dice* al evaluar cruzando dominios sin utilizar adaptación alguna. Sirve de cota inferior (peor caso) de la eficacia de adaptación de dominio.
- $Dice(f_S(G_{2D}(U)))$. Conjunto de coeficientes de *Dice* al evaluar cruzando dominios adaptando con CycleGAN 2D. Mide la eficacia de adaptación de dominio mediante CycleGAN 2D.
- $Dice(f_S(G_{3D}(U)))$. Conjunto de coeficientes de *Dice* al evaluar cruzando dominios adaptando con CycleGAN 3D. Mide la eficacia de adaptación de dominio mediante CycleGAN 3D.

- $Dice(f_U(U))$. Conjunto de coeficientes de $Dice$ al haber entrenado en el mismo dominio de evaluación. Sirve de cota superior (mejor caso) de la eficacia de adaptación de dominio.

Para determinar la eficacia de G mediante coeficiente de $Dice$, se compararon 4 conjuntos de manera análoga en la Sección 5.2.2:

- $Dice(f_U(S))$. Conjunto de coeficientes de $Dice$ al evaluar cruzando dominios sin utilizar adaptación de dominio. Sirve de cota inferior (peor caso) de la eficacia de adaptación de dominio.
- $Dice(f_U(F_{2D}(S)))$. Conjunto de coeficientes de $Dice$ al evaluar cruzando dominios adaptando con CycleGAN 2D. Mide la eficacia de adaptación de dominio mediante CycleGAN 2D.
- $Dice(f_U(F_{3D}(S)))$. Conjunto de coeficientes de $Dice$ al evaluar cruzando dominios adaptando con CycleGAN 2D. Mide la eficacia de adaptación de dominio mediante CycleGAN 3D.
- $Dice(f_S(S))$. Conjunto de coeficientes de $Dice$ al haber entrenado en el mismo dominio de evaluación. Sirve de cota superior (mejor caso) de la eficacia de adaptación de dominio.

Para los casos $f_S(S)$ y $f_U(U)$ se utilizó 7-fold cross validation para evitar la evaluar en imágenes utilizadas durante el entrenamiento. Esta técnica se explica en la Sección 4.5.

5.2.1. Eficacia de F transformando el dominio Singapur

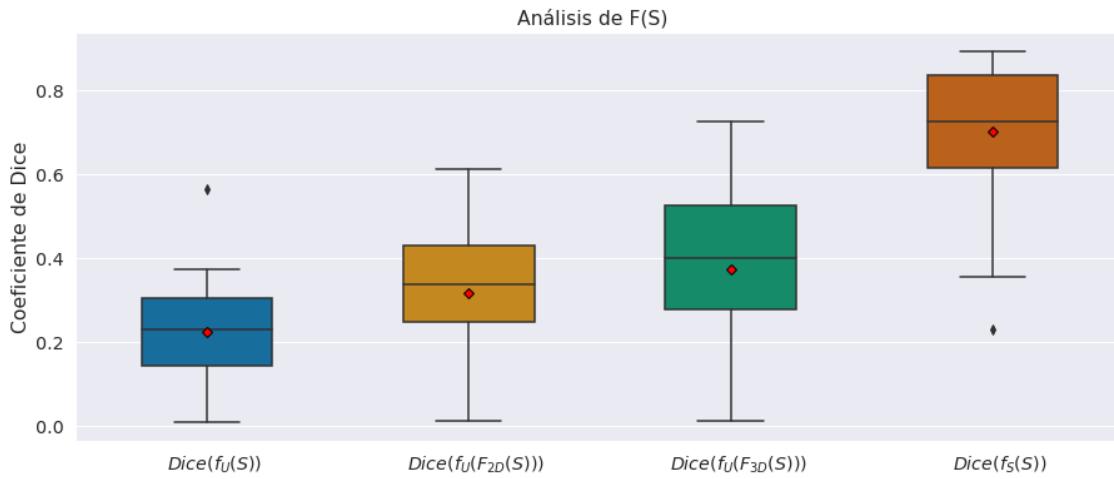


Fig. 5.5: Evaluación de la eficacia de $F : S \rightarrow U$, según coeficiente de Dice.

Al fijar Singapur como dominio objetivo, se puede notar en la Figura 5.5 que el modelo entrenado sobre Utrecht presenta una performance órdenes de magnitud más baja que la

de un modelo entrenado con datos del dominio Singapur. Con una media superior a 0.2, el primer Boxplot da a entender que f_U logra transferir cierto conocimiento aprendido con U al predecir lesiones sobre S , pero aún así el rendimiento es muy bajo.

Por otro lado, el Boxplot de $f_S(S)$ ofrece un baseline superior que consiste en una media de 0.7 y una varianza considerable, siendo este último el estado del arte en segmentación de lesiones de WMH [65].

En el medio se observa que la performance de ambos modelos adaptadores de dominio se sitúan entre los dos baselines. Esto se corresponde con los análisis de divergencia de dominios de la Sección 5.1, demostrando que la herramienta propuesta cumple con su objetivo parcialmente y que tiene lugar para mejorar.

Se corrobora este análisis en la Figura 5.6 mediante la comparación visual de tres ejemplos pertenecientes a los cuatro conjuntos comparados en el Boxplot. Para las tres MRIs de Singapur, la segmentación $f_U(S)$ parece bastante desacertada en comparación con el Ground Truth. Luego, CycleGAN 2D y 3D demuestran su mejora en las columnas $f_U(F_{2D}(S))$ y $f_U(F_{3D}(S))$. Finalmente, $f_S(S)$ demuestra ser más cercano aún al GT, dando lugar a CycleGAN a mejorar su eficiencia.

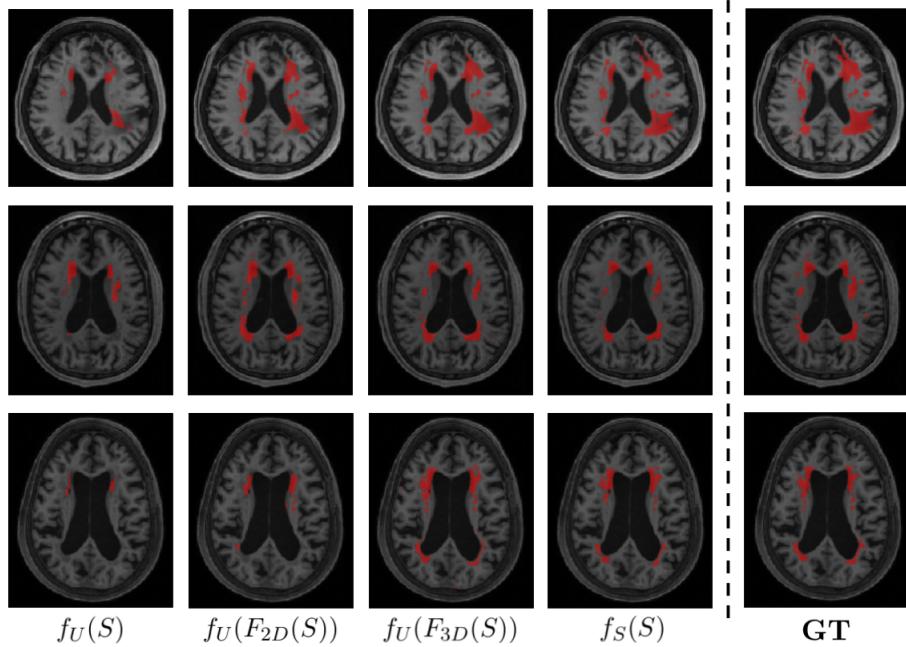


Fig. 5.6: Tres ejemplos concretos de la mejora mediante CycleGAN segmentando WMH sobre el dominio Singapur. Las filas corresponden a tres MRIs de Singapur. Las primeras cuatro columnas identifican el contexto de la segmentación realizada. A la derecha, la segmentación correcta de cada imagen, denominado “Ground Truth”.

5.2.2. Eficacia de G transformando el dominio Utrecht

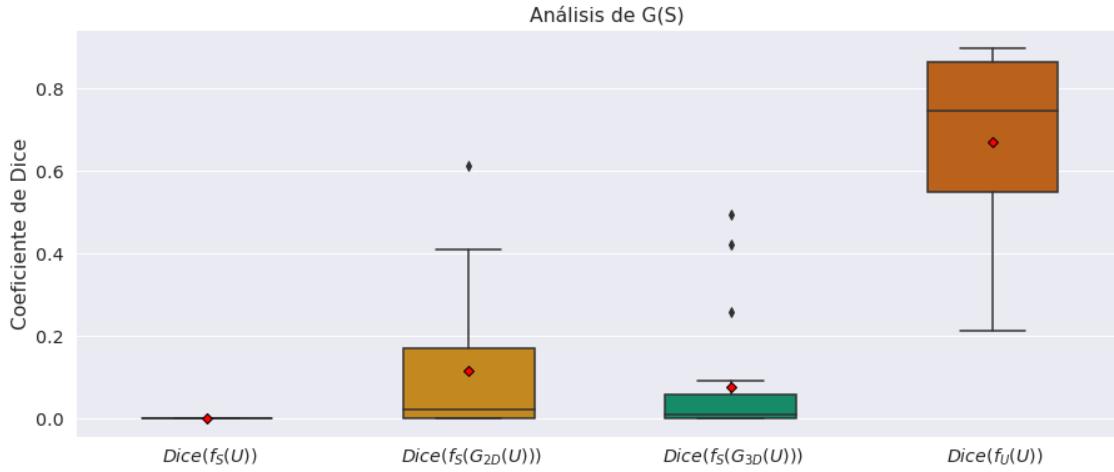


Fig. 5.7: Evaluación de la eficacia de $G : U \rightarrow S$, según coeficiente de Dice.

Al intercambiar dominios origen y destino se obtiene la Figura 5.7. En ella, se puede notar que los cuatro contextos de segmentación obtienen peores resultados con Utrecht como dominio objetivo. Esto evidencia una mayor dificultad en la segmentación de Utrecht, incluso habiendo entrenado con el mismo dominio de evaluación.

A partir de la primer columna, se observa que el modelo entrenado con Singapur no es capaz de aplicar lo aprendido al segmentar Utrecht en absoluto. La Figura anterior 5.5 demostró que sí existe dicha capacidad en el sentido inverso: un modelo entrenado con Utrecht es relativamente capaz de aplicar lo aprendido al segmentar Singapur. Esta asimetría tiene sentido al considerar que las características aprendidas sobre un dominio pueden ser útiles al segmentar sobre el otro, pero no necesariamente ocurre en el sentido opuesto. Es posible que el segmentador entrenado con Utrecht aprenda características y patrones más generales, mientras que el entrenado con Singapur se limite a observar aspectos más específicos de su dominio de entrenamiento.

Con Utrecht como dominio objetivo, la Figura 5.7 sitúa nuevamente a los modelos generadores 2D y 3D entre los baselines propuestos. Ambos logran mejorar la media en 0.1, y se destacan ciertos casos particulares que mejoran aún más (0.3, 0.4, 0.5 y 0.6). Estos últimos casos son notables al tener en cuenta que toda imagen obtiene un resultado de 0 al no aplicar ninguna transformación.

En la Figura 5.8 se corrobora este análisis por medio de la comparación de tres elementos de ejemplo pertenecientes a los cuatro conjuntos comparados en el Boxplot. En este caso, vemos reflejado el *Dice score* de 0 en las segmentaciones nulas de la columna $f_S(U)$. Por su parte, CycleGAN demuestra cierta mejora en las columnas $f_S(G_{2D}(U))$ y $f_S(G_{3D}(U))$. Sin embargo, en este caso CycleGAN es ampliamente superado por el modelo entrenado con el dominio objetivo en la columna $f_U(U)$, la cual se acerca mucho más al GT. Dicho comportamiento corresponde con la Figura 5.7, donde se evidencia que la tarea de segmentar Utrecht es de mayor dificultad que segmentar Singapur, la cual obtenía mejores resultados en los cuatro contextos de segmentación.

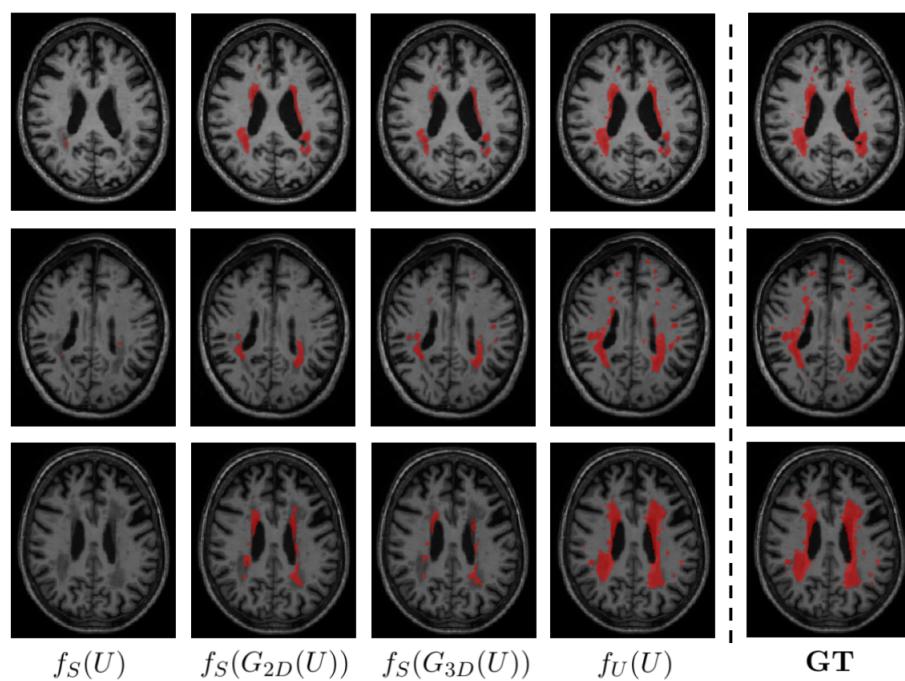


Fig. 5.8: Tres ejemplos concretos de la mejora mediante CycleGAN segmentando WMH sobre el dominio Utrecht. Las filas corresponden a tres MRIs de Utrecht. Las primeras cuatro columnas identifican el contexto de la segmentación realizada. A la derecha, la segmentación correcta de cada imagen, denominado “Ground Truth”.

6. CONCLUSIONES Y TRABAJO FUTURO

El objetivo de este trabajo es proponer una solución viable para el problema de adaptación de dominio al segmentar imágenes de resonancia magnética (MRI) con foco en la hiperintensidad de materia blanca (WMH) cerebral.

Se consideraron dos dominios S y U , cuya diferencia radica en el centro médico de captura, el equipo utilizado y su configuración. Luego de verificar las diferencias en términos de la distribución de intensidades en ambos dominios y corroborar que entrenar un modelo en uno de estos dominios permite predecir correctamente sobre el mismo pero no sobre el otro, se procedió a implementar una estrategia de adaptación de dominio para resolver esta problemática.

A partir de este planteo, la solución propuesta fue emplear CycleGAN: un modelo de redes adversarias que “transfiere el estilo” de una imagen a otra, en base a un entrenamiento que consta de varias funciones de pérdida: adversaria, de consistencia cíclica y de identidad.

Se explicó conceptualmente el funcionamiento de CycleGAN, dando ejemplos prácticos representativos del comportamiento de cada una de las partes. De esta manera, se expusieron y justificaron las decisiones de implementación claves para el correcto funcionamiento del algoritmo. También se propuso un esquema de entrenamiento basado en curriculum learning para lograr transformar imágenes tridimensionales mediante la transformación 2D de sus cortes.

Luego se presentó una variación sobre la implementación original de CycleGAN: modificar las redes generadoras para que transformen imágenes tridimensionales. Para ello, se aprovechó la característica puramente convolucional de las redes generadoras para realizar un entrenamiento con parches de menor dimensión que las imágenes completas.

Se evaluaron tanto CycleGAN 2D como 3D, demostrando la utilidad de ambas para la tarea de adaptación de dominio. Se cuantificó la eficacia de sus transformaciones con dos métricas: primero se midió la calidad de las transformaciones mediante una variante de la divergencia de Jensen-Shannon, luego se midió su efectividad como herramienta de adaptación de dominio mediante el coeficiente de Dice al evaluar la segmentación de imágenes transformadas.

Ninguna de las implementaciones (2D y 3D) demostró una clara superioridad sobre la otra. Esto se debe, probablemente, a que cada una aprende a partir un contexto distinto alrededor del píxel central: la versión 2D aprende a partir del corte axial en el que se ubica, mientras que la versión 3D considera un contexto volumétrico, pero de menor tamaño.

Los resultados empíricos alcanzaron a validar las herramientas propuestas en el marco de experimentación establecido, dando lugar a un posible trabajo futuro de analizar más cantidad de datasets y de mayor heterogeneidad.

La conclusión de este documento se acerca a la del trabajo original de CycleGAN: “Aunque nuestro método obtenga múltiples resultados convincentes, están lejos de ser uniformemente positivos” [9]. Sin embargo, este trabajo es apenas un paso hacia el aprovechamiento del potencial que tiene CycleGAN como adaptador de dominio. La comprensión del comportamiento de CycleGAN, de sus partes generadoras, discriminadoras y de las redes segmentadoras será clave para alcanzar su máximo potencial, y un análisis en profundidad requerirá de un enfoque desde la interpretabilidad de dichos modelos, lo cual

representa un problema de investigación abierto en la actualidad.

En cuanto al futuro cercano, una tarea a abordar será la de usar las transformaciones durante el entrenamiento en lugar de la evaluación. Por otro lado, también sería de gran interés probar esta herramienta sobre otras imágenes médicas (ya sea MRI de otras partes del cuerpo u otros estudios médicos como ecografías, tomografías, etc.).

Finalmente, otra prueba que excedió el objetivo del trabajo es la de entrenar las redes utilizando punto flotante de 16 bits en lugar de 32. Utilizar punto flotante de menor precisión significaría mayor capacidad de almacenamiento en la GPU, lo cual implica poder almacenar modelos más grandes y complejos, posiblemente mejorando los resultados y/o la velocidad del entrenamiento [66].

7. ARTÍCULOS CIENTÍFICOS GENERADOS

Una versión preliminar del trabajo desarrollado en esta Tesis de Licenciatura fue presentada en *Latin American Meeting In Artificial Intelligence, Khipu 2019* [67], realizada en la ciudad de Montevideo, Uruguay. La presentación fue en forma de poster científico, exhibido por el autor del presente trabajo, y obtuvo una mención como mejor poster del congreso.

A su vez, el trabajo fue publicado en SIPAIM 2020 (Symposium on Medical Information Processing and Analysis) [68].

Bibliografía

- [1] Lawrence Roberts. *Machine Perception of Three-Dimensional Solids*. 01 1963.
- [2] Nicolas Roulet, Diego Fernández Slezak, and Enzo Ferrante. Joint learning of brain lesion and anatomy segmentation from heterogeneous datasets. 03 2019.
- [3] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain [j]. *Psychol. Review*, 65:386 – 408, 12 1958.
- [4] Alexander Wissner-Gross. Datasets over algorithms. <https://www.edge.org/response-detail/26587>.
- [5] Annegreet Opbroek, M Ikram, Meike Vernooij, and Marleen de Bruijne. Transfer learning improves supervised image segmentation across imaging protocols. *IEEE transactions on medical imaging*, 34, 11 2014.
- [6] Mohsen Ghafoorian, Nico Karssemeijer, Tom Heskes, Inge WM van Uden, Clara I Sanchez, Geert Litjens, Frank-Erik de Leeuw, Bram van Ginneken, Elena Marchiori, and Bram Platel. Location sensitive deep convolutional neural networks for segmentation of white matter hyperintensities. *Scientific Reports*, 7(1):5110, 2017.
- [7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. volume 9351, pages 234–241, 10 2015.
- [8] MICCAI. Wmh segmentation challenge. <http://wmh.isi.uu.nl/>.
- [9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2242–2251, 2017.
- [10] Marinus T. Vlaardingerbroek and Jacques A. den Boer. Magnetic resonance imaging: Theory and practice. pages 94–06. Springer, 1994.
- [11] Lawrence N Tanenbaum, Apostolos John Tsioridis, Angela N Johnson, Thomas P Naidich, Mark C DeLano, Elias R Melhem, Patrick Quarterman, SX Parameswaran, A Shankaranarayanan, M Goyen, et al. Synthetic mri for clinical neuroimaging: results of the magnetic resonance image compilation (magic) prospective, multicenter, multireader trial. *American Journal of Neuroradiology*, 38(6):1103–1110, 2017.
- [12] John N. Rydberg, Colleen A. Hammond, Roger C. Grimm, Bradley James Erickson, Clifford R. Jack, John Huston, and Stephen J. Riederer. Initial clinical experience in mr imaging of the brain with a fast fluid-attenuated inversion-recovery pulse sequence. *Radiology*, 193 1:173–80, 1994.
- [13] Narmada M. Balasooriya and Ruwan D. Nawarathna. A sophisticated convolutional neural network model for brain tumor classification. *2017 IEEE International Conference on Industrial and Information Systems (ICIIS)*, pages 1–5, 2017.

- [14] J Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 8(6):679–698, June 1986.
- [15] Bischoff L. Adams R. Seeded Region Growing. *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol.16, No.6, pp. 641-647. 1994.
- [16] Marloes M. J. Letteboer, Ole Fogh Olsen, Erik Bjørnager Dam, Peter W. A. Willem, Max A. Viergever, and Wiro J. Niessen. Segmentation of tumors in magnetic resonance brain images using an interactive multiscale watershed algorithm. *Academic radiology*, 11 10:1125–38, 2004.
- [17] Simon Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2nd edition, 1998.
- [18] Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.
- [19] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014.
- [20] Agnes Lydia and Sagayaraj Francis. Adagrad - an optimizer for stochastic gradient descent. 05 2019.
- [21] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.
- [22] N. Akhtar and Ragavendran. U. neural comput & applic. 2019.
- [23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv 1409.1556*, 09 2014.
- [24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. 09 2014.
- [25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.
- [26] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE international conference on computer vision*, pages 1520–1528, 2015.
- [27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
- [28] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.
- [29] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018.

- [30] Ricardo Guerrero, Chen Qin, Ozan Oktay, C. Bowles, L. Chen, Richard Joules, Robin Wolz, M. Valdes-Hernandez, David Dickie, J. Wardlaw, and D. Rueckert. White matter hyperintensity and stroke lesion segmentation and differentiation using convolutional neural networks. *NeuroImage: Clinical*, 17, 06 2017.
- [31] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.
- [32] V. Newcombe J. P. Simpson A. D. Kane D. K. Menon D. Rueckert K. Kamnitsas, C. Ledig and B. Glocker. Efficient multi-scale 3d cnn with fully connected crf for accurate brain lesion segmentation. *Medical Image Analysis*, 36:61–78, 2017.
- [33] L. Yu L. Zhao J. Qin D. Wang V. CT Mok L. Shi Q. Dou, H. Chen and P. A. Heng. Automatic detection of cerebral microbleeds from mr images via 3d convolutional neural networks. *IEEE transactions on medical imaging*, 35(5):1182–1195, 2016.
- [34] S. J. Pan and Q. Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2010.
- [35] Hoo-chang Shin, Holger Roth, Mingchen Gao, Le Lu, Ziyue Xu, Isabella Nogues, Jianhua Yao, Daniel Mollura, and Ronald Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *IEEE Transactions on Medical Imaging*, 35, 02 2016.
- [36] Mohsen Ghafoorian, Alireza Mehrtash, Tina Kapur, Nico Karssemeijer, Elena Marchiori, Mehran Pesteie, Charles R. G. Guttmann, Frank-Erik de Leeuw, Clare Tempany, Bram van Ginneken, Andrey Fedorov, Purang Abolmaesumi, Bram Platel, and William Wells. Transfer learning for domain adaptation in mri: Application in brain lesion segmentation. pages 516–524, 09 2017.
- [37] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance. 12 2014.
- [38] Pim Moeskops, Jelmer M Wolterink, Bas HM van der Velden, Kenneth GA Gilhuijs, Tim Leiner, Max A Viergever, and Ivana Išgum. Deep learning for multi-task medical image segmentation in multiple modalities. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 478–486. Springer, 2016.
- [39] Mauricio Orbes-Arteaga, Thomas Varsavsky, Carole Sudre, Zach Eaton-Rosen, Lewis Haddow, Lauge Sørensen, Mads Nielsen, Akshay Pai, Sébastien Ourselin, Marc Modat, Parashkev Nachev, and Manuel Jorge Cardoso. *Multi-domain Adaptation in Brain MRI Through Paired Consistency and Adversarial Learning*, pages 54–62. 10 2019.
- [40] F. Huszár J. Caballero A. Cunningham A. Acosta A. Aitken A. Tejani J. Totz Z. Wang et al. C. Ledig, L. Theis. Photo-realistic single image super-resolution using a generative adversarial network. In CVPR, 2017.
- [41] P. Isola R. Zhang and A. A. Efros. Colorful image colorization. In ECCV, 2016.

- [42] Konstantinos Kamnitsas, Christian Baumgartner, Christian Ledig, Virginia Newcombe, Joanna Simpson, Andrew Kane, David Menon, Aditya Nori, Antonio Criminisi, Daniel Rueckert, et al. Unsupervised domain adaptation in brain lesion segmentation with adversarial networks. In *International conference on information processing in medical imaging*, pages 597–609. Springer, 2017.
- [43] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.
- [44] Judy Hoffman, Eric Tzeng, Taesung Park, Jun-Yan Zhu, Phillip Isola, Kate Saenko, Alexei A Efros, and Trevor Darrell. Cycada: Cycle-consistent adversarial domain adaptation. *arXiv preprint arXiv:1711.03213*, 2017.
- [45] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [46] Philipp Seeböck, David Romo-Bucheli, Sebastian Waldstein, Hrvoje Bogunović, José Ignacio Orlando, Bianca S Gerendas, Georg Langs, and Ursula Schmidt-Erfurth. Using cyclegans for effectively reducing image variability across oct devices and improving retinal fluid segmentation. *IEEE 16th International Symposium on Biomedical Imaging*, 2019.
- [47] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.
- [48] Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010.
- [49] Anderson A. Susskind, J. and G. E. Hinton. The toronto face dataset. technical report utml tr 2010-001, u. toronto. 2010.
- [50] M. Mirza B. Xu D. Warde-Farley S. Ozair A. Courville I. Goodfellow, J. Pouget-Abadie and Y. Bengio. Generative adversarial nets. In NIPS, 2014.
- [51] M. Aubry Q. Huang T. Zhou, P. Krahenbuhl and A. A. Efros. Learning dense correspondence via 3d-guided cycle consistency. In CVPR, 2016, 2010.
- [52] A. Polyak Y. Taigman and L. Wolf. Unsupervised cross-domain image generation. In ICLR, 2017.
- [53] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.
- [54] Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization: The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

- [55] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6924–6932, 2017.
- [56] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML’10, pages 807–814, USA, 2010. Omnipress.
- [57] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1125–1134, 2017.
- [58] Andrew L. Maas. Rectifier nonlinearities improve neural network acoustic models. 2013.
- [59] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.
- [60] Paul A. Yushkevich, Joseph Piven, Heather Cody Hazlett, Rachel Gimpel Smith, Sean Ho, James C. Gee, and Guido Gerig. User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability. *Neuroimage*, 31(3):1116–1128, 2006.
- [61] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.
- [62] Juan Eugenio Iglesias, Cheng-Yi Liu, Paul M Thompson, and Zhuowen Tu. Robust brain extraction across datasets and comparison with publicly available methods. *IEEE transactions on medical imaging*, 30(9):1617–1634, 2011.
- [63] Augustus Odena, Vincent Dumoulin, and Chris Olah. Deconvolution and checkerboard artifacts. *Distill*, 2016.
- [64] Robert McGill, John W. Tukey, and Wayne A. Larsen. Variations of box plots. *The American Statistician*, 32(1):12–16, 1978.
- [65] Hongwei Li, Gongfa Jiang, Jianguo Zhang, Ruixuan Wang, Zhaolei Wang, Wei-Shi Zheng, and Bjoern Menze. Fully convolutional network ensembles for white matter hyperintensities segmentation in mr images. *NeuroImage*, 183:650–665, 2018.
- [66] Dhiraj D. Kalamkar, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, Nataraj Jammalamadaka,

- Jianyu Huang, Hector Yuen, Jiyan Yang, Jongsoo Park, Alexander Heinecke, Evangelos Georganas, Sudarshan Srinivasan, Abhisek Kundu, Misha Smelyanskiy, Bharat Kaul, and Pradeep Dubey. A study of BFLOAT16 for deep learning training. *CoRR*, abs/1905.12322, 2019.
- [67] Khipu - latin american meeting in artificial intelligence. <http://khipu.ai>. 2019.
- [68] Julian Alberto Palladino, Diego Fernandez Slezak, and Enzo Ferrante. Unsupervised domain adaptation via cyclegan for white matter hyperintensity segmentation in multicenter mr images. <https://arxiv.org/abs/2009.04985>, 2020.