

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS



VNIVERSITAT  
DE VALÈNCIA

TRABAJO DE

FIN DE MÁSTER

ESTUDIO GENERACIONAL DE LA  
SEGMENTACIÓN SEMÁNTICA PROFUNDA EN  
EL CONTEXTO MÉDICO

AUTOR:  
ALBERTO SOLANO CAÑADAS

TUTORES:  
KEVIN NICHOLAS DIETRICH  
EMILIO SORIA OLIVAS

# Índice

Sección	Página
<b>1. Introducción</b>	<b>3</b>
1.1. Antecedentes . . . . .	4
1.2. Segmentación y deep learning . . . . .	7
1.2.1. U-Net . . . . .	9
1.2.2. Vision Transformers . . . . .	10
1.2.3. ConvMixer . . . . .	13
1.3. DRIVE dataset . . . . .	15
<b>2. Resultados</b>	<b>16</b>
2.1. Aproximaciones “clásicas” . . . . .	16
2.2. Algoritmos de clustering . . . . .	17
2.3. Otros algoritmos de machine learning . . . . .	20
2.4. Deep learning . . . . .	24
2.4.1. U-Net . . . . .	25
2.4.2. Transformers . . . . .	31
2.4.3. ConvMixer . . . . .	39
<b>3. Conclusiones</b>	<b>44</b>

## Resumen

A lo largo de este trabajo se ha realizado un recorrido generacional por las diferentes técnicas de segmentación semántica, particularizando su uso en el contexto médico y haciendo especial hincapié en los algoritmos de deep learning más recientes que componen el *state-of-the-art*. Empleando un dataset con imágenes retinales asociado al problema de conseguir separar los vasos sanguíneos del globo ocular, se ha logrado: en primer lugar resolver el problema, exprimiendo al máximo la capacidad de la U-Net como arquitectura estándar; en segundo lugar y como principal hito del trabajo, batir a esta última modificando una arquitectura muy novedosa llamada ConvMixer. El modelo ideado es capaz de superar a la U-Net con menos de 300.000 parámetros entrenables, es decir, un factor 100 en la reducción del número de parámetros.

## 1. Introducción

El paradigma de la segmentación en imágenes en la actualidad está completamente monopolizado por las redes neuronales profundas [1]. Se emplean un conjunto de técnicas y arquitecturas de redes muy específicas, las cuales consiguen unos rendimientos superiores frente a otros de los métodos que quedan excluidos del conjunto de algoritmos que componen el denominado *deep learning*. Siguiendo con esta catalogación, estas técnicas consideradas el *state of the art* actualmente, se encasillan de manera aún más general dentro un “saco” bastante amplio como es el *machine learning*, y, concretamente, se puede decir que forman parte un gran bloque conformado por los problemas de clasificación en imágenes.

La segmentación es, en última instancia, una clasificación píxel a píxel, una extensión de la clasificación de imágenes en la que explícitamente se incluye el concepto de localización, ya que el modelo apunta al correspondiente objeto presente en la imagen y define su contorno. En el caso particular de la segmentación en deep learning, incrementando el nivel de abstracción se puede ver el problema a grandes rasgos como un problema de clasificación de carácter supervisado, es decir, para el entrenamiento de los modelos se han de emplear datos que se correspondan con una tupla imagen-etiqueta, siendo esta etiqueta en el caso de la segmentación una máscara de la propia imagen. Esta máscara no es más que una imagen la cual viene clasificada píxel a píxel de acuerdo a la segmentación propuesta para dicha imagen. En oposición a los problemas de aprendizaje no supervisado donde los algoritmos no se alimentan de etiquetas pre-assignadas para el entrenamiento, en aprendizaje supervisado siempre se asocia una máscara o “ground truth” a la imagen. Dependiendo del tipo de tarea de segmentación a llevar a cabo, esta máscara tendrá una forma específica u otra. Por ejemplo es posible tener un único elemento en la imagen el cual segmentar, tener varios, o incluso segmentar varios elementos de diferentes tipos, siendo este último ejemplo una clasificación no binaria sino multiclas.

Las técnicas de segmentación mediante algoritmos de deep learning se extienden de manera exponencial a través de multitud de problemas y escenarios diferentes desde la propia concepción del deep learning para la resolución de tareas dentro del campo de

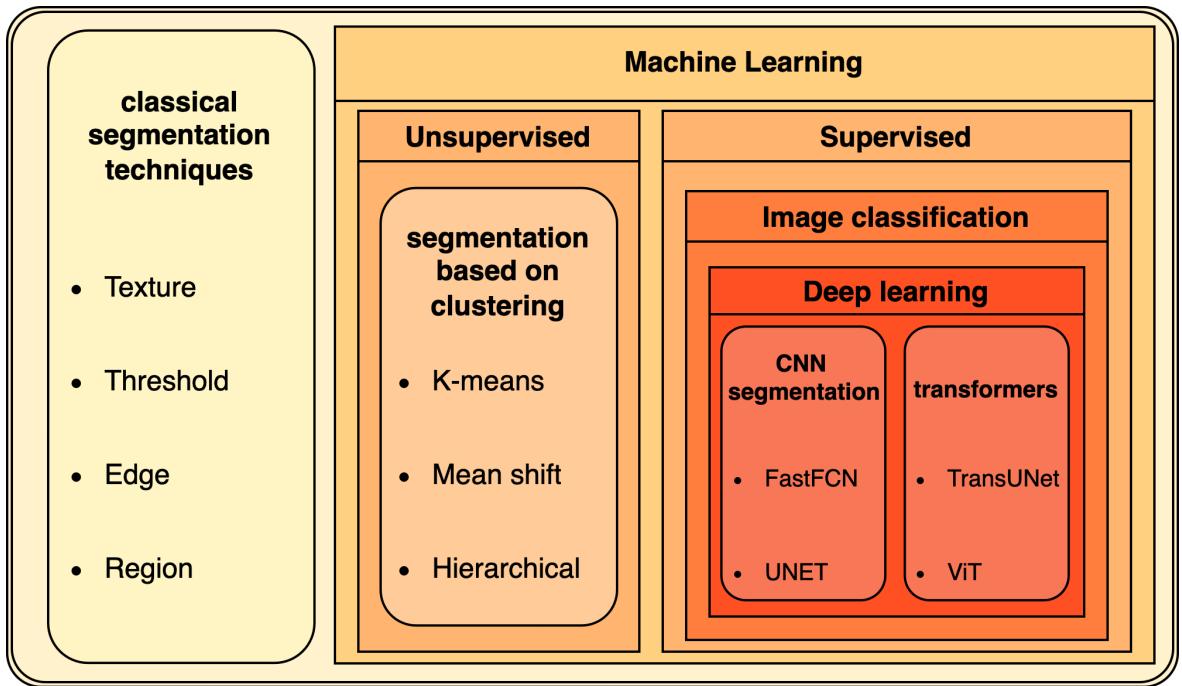
la visión por ordenador. Sin embargo, este desarrollo de la segmentación de imágenes sobre una base común de machine learning no siempre había existido. La aplicación de todas estas técnicas es relativamente reciente, pese a que el desarrollo nuevos métodos y soluciones de cara a la segmentación en imágenes se remonta a mucho antes de que existiera la primera red neuronal convolucional. Con el fin de situar el objeto principal de este trabajo, la revisión de algunos de los principales algoritmos de segmentación de imágenes mediante deep learning, conviene dotar de contexto a las técnicas de segmentación en imágenes como el ente en constante evolución a lo largo del tiempo que es, y por tanto hacer una pequeña revisión de los primeros métodos [2] que preceden a la revolución vivida en el campo de la visión por ordenador con la llegada de las primeras redes neuronales convolucionales plenamente funcionales[3].

La Figura [1] presenta algunos de los algoritmos de segmentación más populares [4], incluyendo algunas arquitecturas de deep learning más utilizadas en la actualidad en campos como la segmentación en imágenes médicas[5] o de satélite[6]. Sin embargo, este esquema no debe seguirse al pie de la letra y simplemente ha de servir como una herramienta ilustrativa ya que ciertos términos como el propio machine learning son relativamente ambiguos, lo cual lleva a que técnicas como la detección de bordes puedan abordarse desde un punto de vista más clásico, por ejemplo aplicando un filtro de Sobel a la imagen [7], o empleando técnicas más modernas de deep learning [8]. Además, algunos métodos por ejemplo de clustering, cumplen con la definición de aproximaciones basadas en la búsqueda de similitudes entre píxeles, cosa que los sitúa al mismo nivel que las técnicas *Region-based* y *Texture-based* de segmentación en imágenes. En cualquier caso, el esquema permite hacerse una idea del gran número de alternativas disponibles a la hora de segmentar una imagen.

## 1.1. Antecedentes

La segmentación surge como una de las principales líneas de investigación dentro del campo del procesado de imágenes digitales, junto a la extracción de características y el reconocimiento de patrones en imágenes [20]. Entorno a la década de los 60 se comenzaba a hablar del término “computer vision” de manera completamente entusiasta por parte de los primeros investigadores y universidades que se adentraban en el tema. Este interés venía creciendo desde el año 1956 cuando el término “artificial intelligence” fue por primera vez acuñado en un seminario que tuvo lugar en la universidad de Dartmouth [21].

Estos dos términos, computer vision y digital image processing, por aquel entonces seguían estando claramente diferenciados. Mientras que en el primer campo los esfuerzos de los científicos se concentraban en tratar de recrear las estructuras neurológicas humanas de forma digital, en el segundo, la finalidad era la de tomar una imagen y mediante algún tipo de algoritmo “mejorarla”. Esta mejora en la imagen puede consistir en una transformación a través de una corrección en el brillo o una reducción del ruido de la misma, la extracción de cierta información particular o la extracción parcial de una cierta región de la imagen para su ( posible) preparación para una tarea posterior. La segmentación ocuparía su lugar en este último punto.



**Figura 1:** “Framework” general de algunos de los algoritmos de segmentación más populares. Cada uno de los elementos pertenece a una o varias categorías dentro del conjunto más amplio de la segmentación en imágenes que recogería a todos ellos. Siguiendo un orden cronológico y de complejidad ascendente, de izquierda a derecha en primer lugar se encuentran los algoritmos de segmentación “clásicos”, los cuales siguen una aproximación por lo general más sencilla basada en la búsqueda de discontinuidades (Edge [9], Threshold [10]) o similitudes (Texture [11], Region [12]) dentro de la imagen. A continuación, los algoritmos de clustering jerárquico [13], K-means [14] o Mean shift [15], son reconocidos de manera generalizada por la comunidad como métodos de machine learning. Buscan crear grupos de píxeles similares a través de procesos computacionales recursivos hasta la convergencia. Por último, las arquitecturas basadas en redes neuronales convolucionales (FastFCN [16], U-net [5]) o incluso más recientemente el descubrimiento de los mecanismos de “self-attention” [17] en los denominados “transformers” (TransUNet [18], ViT [19]) son los últimos elementos que se incorporan al paradigma de la segmentación en imágenes en la actualidad.

En este contexto, a lo largo de esta década numerosos trabajos en el campo del procesado de imágenes digitales comienzan a publicarse, concretamente, a finales de la década, los primeros estudios en el campo particular de la segmentación comienzan a aparecer [22] [23]. Estos trabajos exploran los principales métodos que aparecen en la Figura 1, tanto las aproximaciones clásicas como las basadas en clustering. La aproximación más sencilla a la hora de segmentar una imagen, el método de *Thresholding*, consiste en precisamente establecer un umbral por ejemplo en la intensidad de los píxeles de manera que la imagen quedaría dividida en dos regiones. Esta aproximación tiene por supuesto muchas debilidades y sólo podría ser eficaz en imágenes muy simples, en las que el elemento a diferenciar o segmentar tuviera claramente unos valores en la intensidad muy diferentes al background del cual se quisiera separar. Debido a que el paradigma del procesado de imágenes digitales es tan amplio, en función del problema a tratar y su campo de aplicación, aparecen nuevos algoritmos más adecuados para los

diferentes problemas que van surgiendo, a la vez que van perfeccionándose los ya existentes para tareas más concretas. Campos como la imagen médica, la teledetección o las aplicaciones militares son algunos ejemplos de entornos favorables para el desarrollo de todos estos algoritmos.

Los años posteriores a 1970, mientras que en el campo de la segmentación se seguían logrando avances, el mundo de la inteligencia artificial por contra vivía uno de sus peores momentos. Poco a poco los investigadores se dieron cuenta de que las propuestas en esta línea rozaban lo fantástico, los problemas formulados por los propios científicos especializados en esta materia eran demasiado complejos a nivel técnico como para ser resueltos con los recursos disponibles. Poco a poco, con promesas que cayeron en saco roto y la falta de resultados, la financiación cayó, mientras la crítica seguía aumentando, no en vano a la época que comprende los años posteriores hasta aproximadamente la entrada los años 90 se le denominó el “Invierno de la IA” o “AI winter” [24]. Un famoso ejemplo fue cuando en 1966 uno de los estudiantes de Marvin Minsky, un famoso investigador de IA en el MIT lab, se le fue asignada la tarea de conectar una cámara a un ordenador de forma que el ordenador describiera lo que la cámara estaba viendo, tarea la cual obviamente no pudo ser resulta por el estudiante. La visión por ordenador resultó ser “one of the most difficult and frustrating challenges in AI over the next four decades” [25].

Una de las contribuciones más tempranas en el campo de la visión por ordenador fue el trabajo de LeCun en 1989 [26] en el reconocimiento de caracteres en códigos postales de EEUU. Desde aquel entonces comenzó un pequeño resurgimiento, acompañado e impulsado por el desarrollo de nuevo hardware y dispositivos más potentes. Sin embargo, el campo de la visión por ordenador no daría un impulso relevante hasta 2012. Contrariamente, en el campo de la segmentación, previamente a la llegada del nuevo siglo los avances acumulados a lo largo de los años eran más que perceptibles. Muchos de los algoritmos comentados se habían perfeccionado para lograr rendimientos óptimos en los diferentes problemas, surgieron nuevas variantes de estos algoritmos e incluso se hablaba de una prometedora futura línea de investigación, que consistiría en dejar de mejorar de forma aislada estos algoritmos segmentación ad hoc, para crear técnicas híbridas [27].

En torno a la primera década de los años 2000, los avances en el campo del machine learning se sucedieron con mucha rapidez. En general, aparecieron muchos algoritmos que a día de hoy son bastante populares, como por ejemplo las máquinas de vector soporte [28] (SVMs) en 1995, o las conocidas redes recurrentes LSTM [29] unos años más tarde. Las arquitecturas de las redes neuronales se hacían cada vez más complejas y su rendimiento comenzó a hacerse comparable al de las SVMs. A estas redes se les empezó a denominar “redes neuronales artificiales” (ANNs) y su popularidad generó avances por ejemplo en el estudio de nuevas funciones de activación y mecanismos de optimización al topar entre otros muchos nuevos problemas con el desvanecimiento del gradiente [30]. Por la parte del hardware, su desarrollo llevó a dispositivos mucho más baratos y, sobre todo, rápidos. En particular el uso de las GPUs, aceleraba de manera exponencial los cálculos en tareas de procesado de imágenes.

Este salto en la cantidad de recursos disponibles para los investigadores derivó en el descubrimiento de una nueva necesidad; los datos, el combustible fundamental de los

modelos de machine learning y deep learning que consigue dotar a los mismos de la capacidad de superar en rendimiento a otras alternativas. La década estuvo marcada por la incipiente búsqueda de nuevos datos que almacenar y explotar, los primeros en comenzar a subirse a la ola del recién aparecido *Big Data* fueron grandes compañías como Google [31] o las principales redes sociales como Twitter o Facebook. Nuevas bases de datos públicas fueron apareciendo y, para finales de la década, la cantidad de datos generados se media en zetta bytes. Una de las bases de datos más relevante en la evolución de la visión por ordenador fue ImageNet [32], un repositorio de libre acceso con más de 14 millones de imágenes etiquetadas creada en 2009 por Fei-Fei Li.

El campo de la segmentación bebió de todos estos avances en los algoritmos de machine learning, incorporando muchas de las técnicas de aprendizaje tanto supervisado como no supervisado en las distintas aplicaciones. Con la llegada de ImageNet, la segmentación se encontraba en plena búsqueda de un algoritmo de segmentación universal [33], precisamente porque ninguno de los algoritmos podía ser aplicado a todo tipo de imágenes, por lo que la segmentación se veía afectada por muchos factores como la homogeneidad de las imágenes, la textura, el contenido o la estructura espacial de las mismas. La llegada en 2012 de AlexNet [3] cambiaría para siempre el paradigma de la visión por ordenador. Al conseguir sobreponerse de manera abrumadora a todas las técnicas jamás vistas en la clasificación de imágenes, investigadores de todo el mundo comenzaron a utilizar las redes neuronales convolucionales para las aplicaciones con imágenes, dando inicio a una nueva etapa en el que el deep learning se consolidaba como el núcleo central de la visión por ordenador. Por primera vez, la segmentación y la visión por ordenador parecían converger hacia el mismo punto, siendo el deep learning el elemento en común y el candidato a algoritmo universal para las tareas de segmentación mencionado con anterioridad.

## 1.2. Segmentación y deep learning

Tras la demostración del potencial de las redes neuronales convolucionales para la clasificación de imágenes, una cascada de nuevos trabajos inspirados en estas arquitecturas se sucedieron en los años siguientes. Inevitablemente, en el campo de la segmentación en imágenes comenzaron a aparecer los primeros trabajos basados en estas arquitecturas.

Dentro de la segmentación en deep learning se suele diferenciar entre *Semantic Segmentation* e *Instance Segmentation*. Mientras que en la primera el objetivo es clasificar píxel a píxel en la imagen en base a unas etiquetas o categorías, la segunda pretende ir un paso más allá distinguiendo entre varios entes dentro de la misma categoría. Un ejemplo para ilustrar estas diferencias sería pretender detectar los peatones y los vehículos que aparecen en una imagen tomada en una calle a una cierta hora por una cámara (*Semantic Segmentation*), frente al querer distinguir el número de peatones y vehículos que aparecen en la imagen (*Instance Segmentation* [34]), en cuyo caso se hace necesaria la distinción entre los elementos que componen una misma categoría, en este ejemplo persona o vehículo.

La segmentación semántica es el problema más general y precede a la segmentación de instancias, que es un caso particular más complejo. Las primeras arquitecturas que

aparecen para resolver este problema son las llamadas “Fully Convolutional Networks” o FCN [35]. Estas redes aprovecharon algunas de las arquitecturas existentes conocidas como la VGG16, la GoogLeNet o la misma AlexNet, cuya efectividad ya se había comprobado. Sin embargo, reemplazaban las últimas capas de neuronas “*fully connected*” por nuevas capas convolucionales, por lo que las imágenes únicamente se realizaban operaciones de convolución, optimizándose los pesos de los filtros que dan lugar a dichas operaciones, de ahí su nombre. Estas arquitecturas son consideradas un hito en la historia de la segmentación ya que produjeron los primeros resultados dentro del campo de la segmentación empleando técnicas de deep learning. Al utilizar convoluciones en la capa de salida en lugar de neuronas simples, el modelo devuelve un conjunto de “heatmaps”. Mediante un proceso al que los autores denominan deconvolución, se combinan estos mapas de salida, por ejemplo a través de una interpolación, y se obtiene la imagen segmentada final.

El uso de este tipo de arquitecturas para tareas de segmentación comenzó a extenderse en la literatura, haciendo cada vez más populares y consolidándose como el state-of-the-art en la segmentación. Sin embargo, estas redes presentaban algunas desventajas como por ejemplo el no ser lo suficientemente rápidas para su uso en aplicaciones en tiempo real o el tener ciertamente difusa la localización en las últimas capas, lo que dificulta la precisión de la segmentación. Algunas propuestas de mejora en estas redes pasan por la adición de *Markov Random Fields* (MRFs) o *Conditional Random Fields* (CRFs) en las arquitecturas de deep learning con fin de incorporar e integrar un contexto semántico [36].

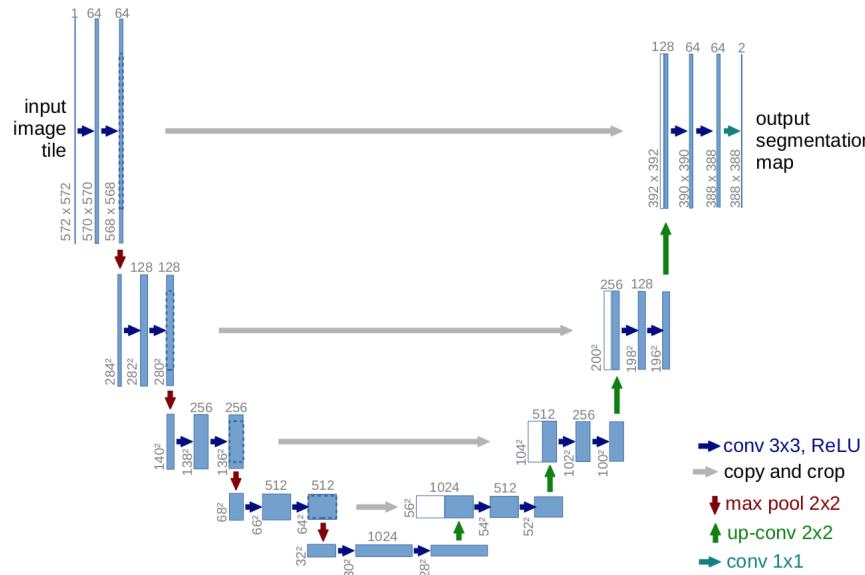
No se tardó en reemplazar [37] las operaciones de “deconvolución” de las redes FCN por una estructura complementaria completa de capas, que de alguna manera hacen la operación contraria a las primeras convoluciones características de las redes neuronales convolucionales para la clasificación de imágenes. Mientras que estas primeras convoluciones son las encargadas del *feature extraction*, las otras recomponen sucesivamente la imagen final a partir de las características extraídas. De esta manera, la arquitectura al completo forma una estructura simétrica con ambas partes claramente diferenciadas, las convoluciones, y, siendo rigurosos con el nombre técnico, las convoluciones “transpuestas”. A esta nueva familia de arquitecturas de deep learning para la segmentación de imágenes se le conoce con el nombre de modelos encoder-decoder.

Algunas de las redes más populares que emplean estas arquitecturas como la SegNet [38] o la HRNet [39] se diseñan para la tarea general de segmentación de imágenes. Sin embargo, surgieron modelos cuyo propósito estaba orientado directamente a la segmentación de imágenes médicas/biomédicas. La U-Net, mencionada anteriormente (Figura 1), es la más popular de todas ellas, y aunque en su origen se concibió para las aplicaciones médicas, a día de hoy esta arquitectura se emplea para multitud de tareas de segmentación de imágenes, sin ser exclusiva del campo médico. A continuación se pasa a describir en detalle dicha arquitectura, ya que forma parte de los modelos empleados en este trabajo, cuyo desempeño se evalúa en secciones posteriores (Véase sección 2.4.1).

### 1.2.1. U-Net

La U-net pertenece a la familia de redes de segmentación del estilo encoder-decoder. Como la obtención de imágenes médicas suele ser un proceso costoso y delicado, esta red se diseña explícitamente para lidiar con el problema de tener muestra de entrenamiento pequeña, apoyándose a su vez en la aplicación de técnicas de data augmentation en el entrenamiento de este modelo a la hora de conseguir los mejores resultados posibles. La modificación en la arquitectura añadiendo un segundo bloque de “*upsampling*” con respecto a su predecesora la FCN [35], hace que se consiga una segmentación más precisa, para ello se dispone de una serie de elementos clave a destacar en el diseño de la misma.

La primera parte también llamada “*downsampling*” sigue una misma estructura repetida un total de 4 veces. Este bloque se compone de la aplicación de dos sucesivas convoluciones a través de filtros de tamaño 3x3 (sin *padding*), cada una de ellas seguida por la aplicación de una función de activación RELU, cuyo resultado sirve como entrada a una operación de *max pooling* con *stride* = 2 para reducir a la mitad las dimensiones de las imágenes resultantes. Con cada sucesiva aplicación de este conjunto de operaciones (el bloque anterior) se duplica el número de filtros [64 → 128 → 256 → 512 → 1024] hasta finalmente quedar un total de 1024 pequeñas imágenes o “*tiles*”.



**Figura 2:** Esquema de la arquitectura general de la U-Net extraído del *paper* original [5]. Pese a que esta ilustración aplica a las imágenes empleadas en dicho trabajo, que poseen dimensiones de 572x572 píxeles, esta arquitectura es extensible a imágenes de distintos tamaños de forma prácticamente inmediata. Únicamente cambiarían consecutivamente las dimensiones de salida tras las sucesivas convoluciones en función de las que se tengan a la entrada. La característica forma de “U” de la arquitectura es lo que da origen a su nombre.

A continuación, en la parte de *upsampling* o también llamada *expansive part*, se realizan sucesivas convoluciones transpuestas. Es común en la literatura encontrar el término “deconvolución”, probablemente heredado de la FCN, sin embargo este nombre es poco

acertado ya que la deconvolución es una operación matemática que revierte una convolución, sin embargo en una convolución transpuesta, se reconstruyen las dimensiones originales pero no devuelve el mismo input previo a realizar la convolución. Básicamente esta operación consiste en añadir un cierto *padding* a la imagen original y realizar una convolución usual a través de un filtro. Como los parámetros de este filtro quedan libres de ser aprendidos por la red, se consigue una imagen a la salida de mayores dimensiones la cual, en cierto modo, preserva los atributos de la imagen de entrada.

El primer *upsampling* toma como entrada 1024 imágenes de dimensiones (pongamos genéricamente) MxN y tras la convolución transpuesta se obtienen 512 imágenes de tamaño (2M)x(2N), habiendo reducido a la mitad el número de canales y duplicado las dimensiones. Adicionalmente, otro elemento relevante de la arquitectura son las llamadas “*skip connections*”, que aparecen representadas en la Figura 2 como flechas grises. Gracias a estas conexiones se consigue combinar la información de capas previas para obtener una mayor precisión. Simplemente se copian las capas previas a las imágenes de salida de las convoluciones transpuestas, en el caso anterior del primer *upsampling*, combinaría las 512 imágenes de tamaños (2M)x(2N) con sus imágenes análogas de la parte de *downsampling*. A través de estos elementos se ensambla la arquitectura completa de la U-Net.

### 1.2.2. Vision Transformers

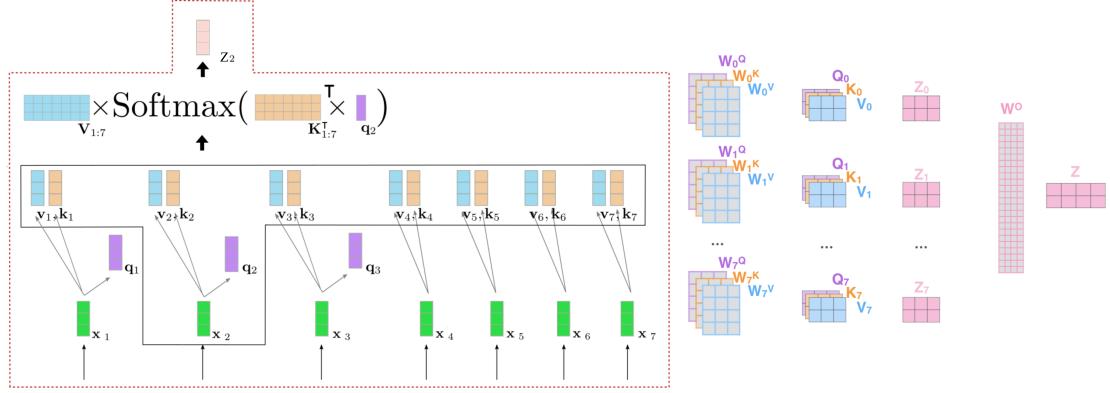
La llegada de los *transformers* al campo del procesado del lenguaje natural se produce en 2017 de la mano de Vaswani et al. [40]. Esta arquitectura introduce lo que se conoce como mecanismos de atención como novedad con respecto al resto de modelos que componían el state-of-the-art en aquel entonces.

La idea tras los mecanismos de atención es el ser capaz, en el caso del NLP, de retener información del contexto global de una frase a través de establecer relaciones entre las palabras que conforman la frase. Al igual que con los mecanismos cognitivos del ser humano, aprender qué parte de los datos es más importante que otra depende del contexto, por ejemplo en la frase “El peatón no cruzó la calzada al ver el semáforo porque estaba en rojo”, la conexión entre el término “estaba” y “semáforo” es evidente, sin embargo hay que incorporar este mecanismo de atención para evidenciar esta relación a nivel de código.

Los autores, incorporaron en la arquitectura de su red una serie de elementos, entre los que destaca el uso recurrente de capas de “*self-attention*”. Estas capas aprovechan varios mecanismos para capturar las relaciones entre las palabras.

Básicamente, el mecanismo de *self-attention* consiste en, partiendo de los vectores individuales de cada una de las palabras una vez salen del embedding, se combinan a través de una serie de operaciones un conjunto de matrices llamadas “Query”, “Key” y “Value” que denotaremos como Q, K y V respectivamente. Estas matrices realmente son abstracciones útiles para definir el mecanismo de atención, que de manera simplificada para un único vector que representaría la segunda palabra de una frase aplicaría de la siguiente forma: Haciendo inicializado los valores de estas matrices, se toma el producto escalar de los vectores  $q_2$  y  $k_i$ , siendo  $q_2$  la proyección de  $x_2$  a través del embedding;

el subíndice  $i$  representa a cada una de las palabras de la frase, incluyendo la propia segunda palabra. Posteriormente, se normalizan los valores en base a un valor que tiene que ver con la dimensión de  $K$  y se aplica una función *softmax* sobre todos los valores resultantes, de manera que las dependencias más débiles prácticamente se eliminan al tender a cero. Finalmente, se suman los vectores  $v_i$  pesados por el resultado de la *softmax* y se obtiene el vector final  $z_2$  que se “envía” a una red neuronal densa que recoge la salida de esta capa de atención y la convierte en la entrada de una nueva capa de atención.



**Figura 3:** Esquema detallado del mecanismo de atención en el transformer. A la izquierda, se representa el paso de una oración de 7 palabras por un único módulo de atención del transformer. Por simplicidad solo se muestran 3 de los 7 vectores  $q_i$ . En la parte derecha se muestra el ensamblado de la salida de múltiples módulos de atención.

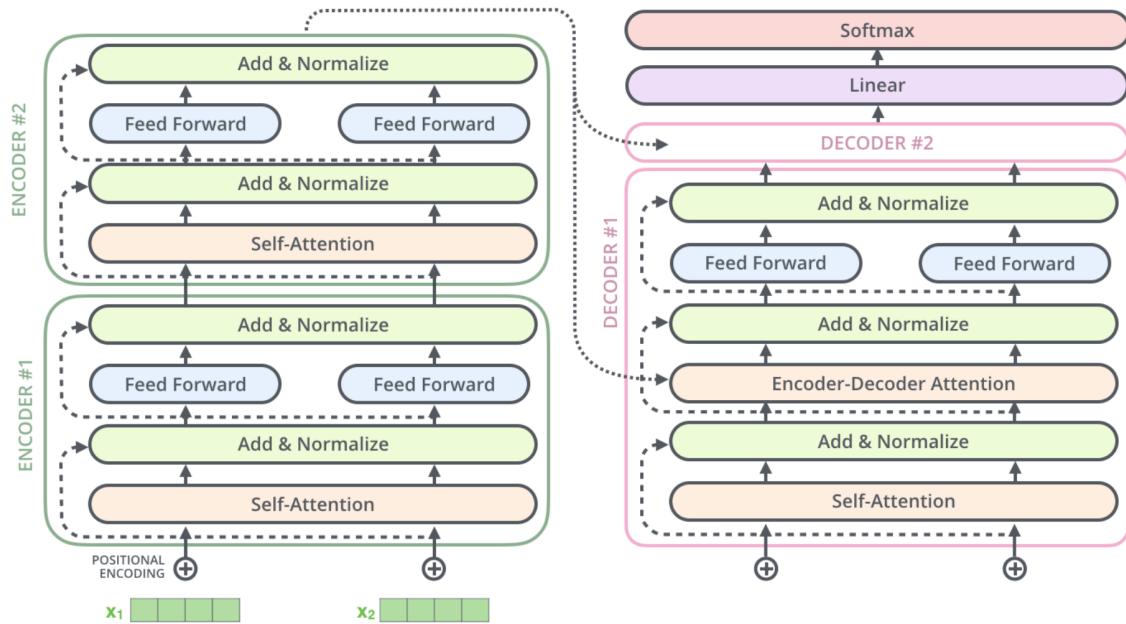
Estas operaciones se podrían repetir para todas las palabras de la frase y de esta manera se construiría la matriz  $Z$  con los resultados para la frase entera. Sin embargo, justamente estas operaciones pueden representarse y realizarse de manera matricial, lo cual es mucho más conveniente en cuanto a eficiencia a la hora del entrenamiento e inferencia. El esquema del ejemplo descrito anteriormente se presenta en el bloque izquierdo la Figura 3.

Este proceso se puede replicar varias veces para conseguir “*multi-head attention*”, que es el mecanismo explicado en el *paper*. Simplemente, los datos de salida del embedding se dividen entre los diferentes “*heads*” en los que se realiza este juego de multiplicación de matrices independientemente. Con el producto escalar de la concatenación de las diferentes matrices  $Z$  de los distintos *heads*, con otra matriz abstracta cuyos pesos son entrenados, se obtiene el output final  $Z$  que espera la red neuronal de entrada, tal y como se describe en la parte derecha de la Figura 3. Al dividir estas operaciones en varios *heads*, se consigue expandir la capacidad del modelo para fijarse en diferentes posiciones, además de poder parallelizar los cálculos sobre los *heads*, lo cual es otro punto clave de esta arquitectura.

Un par de detalles completan el módulo de encoding que se realiza en estos *transformers*, el primero de ellos es la adición de un vector adicional a los embeddings con la noción de la posición de las palabras, o la relación de distancia entre ellas, mientras que el segundo es la introducción de capas adicionales de normalización.

Los decoders siguen una estructura muy similar pero con pequeñas diferencias, como por ejemplo que toman su propia matriz  $Q$ , pero las matrices  $K$  y  $V$  del *stack* de

encoders en unas capas exclusivas de esta parte llamadas “enconder-decoder *attention layers*”. Además, en el cálculo de los productos escalares matriciales de los decoders, se enmascaran los valores “futuros” para que sólo se preste atención a posiciones anteriores, la Figura 4 esquematiza esta arquitectura.



**Figura 4:** Arquitectura a modo ilustrativo de la estructura del encoder y decoder para el caso particular de un transformer con 2 de estos bloques. En el artículo original de Vaswani et al. se utilizan 6 de ellos con esta misma estructura. Extraído de J Alammar [41].

Gracias a las arquitecturas de los *transformers* y sus derivados, se ha alcanzado un consenso absoluto [42] por parte de la comunidad en que estas arquitecturas componen el state-of-the-art del NLP. Este reconocimiento ha propiciado un impulso en el uso de esta clase de redes para tareas de NLP. Como era de esperar, los investigadores no han tardado en tratar de cruzar la frontera del NLP hacia nuevos campos. La introducción de mecanismos de atención en tareas de computer vision como clasificación de imágenes o segmentación parece ganar popularidad con el paso del tiempo [43], algunos autores llegan a plantear la combinación arquitecturas tipo CNNs con capas de *transformers*, llegando incluso a reemplazar por completo las convoluciones [44]. Sin embargo los resultados obtenidos no llegan a ser mejores que las arquitecturas basadas en ResNet, al menos hasta la llegada en 2020 de los “*Vision Transformers*” (ViT) [45] de la mano del equipo de Google-research.

El objetivo de los autores era el de aplicar la estructura del transformer original [40] con el mínimo número posible de modificaciones para que pudiera extrapolarse a imágenes. Con este planteamiento, la idea de los autores es la de dividir la imagen en pequeños “*patches*” sin solapamiento que, posteriormente, se transforman en un vector 1D a través de una proyección lineal entrenable (“*flattened*”). Este proceso al cual le denominan *patch embeddings*, se combina con *embeddings* posicionales entrenables en 1D para retener la información de la posición de los píxeles. El resultado sirve de entrada para el primer encoder del *transformer*.

Es importante mencionar este trabajo ya que abre las puertas al uso de *transformers* en computer vision y más concretamente en el ámbito de la segmentación. Tal y como mencionan en las conclusiones de su trabajo: “*While these initial results are encouraging, many challenges remain. One is to apply ViT to other computer vision tasks, such as detection and segmentation.*”

Precisamente a raíz de este artículo, nuevas arquitecturas especializadas en segmentación de imágenes surgieron tomando como inspiración la idea principal de trabajar con estos recortes de la imagen original, algunas de ellas han sido seleccionadas para su uso a lo largo de este trabajo.

### 1.2.3. ConvMixer

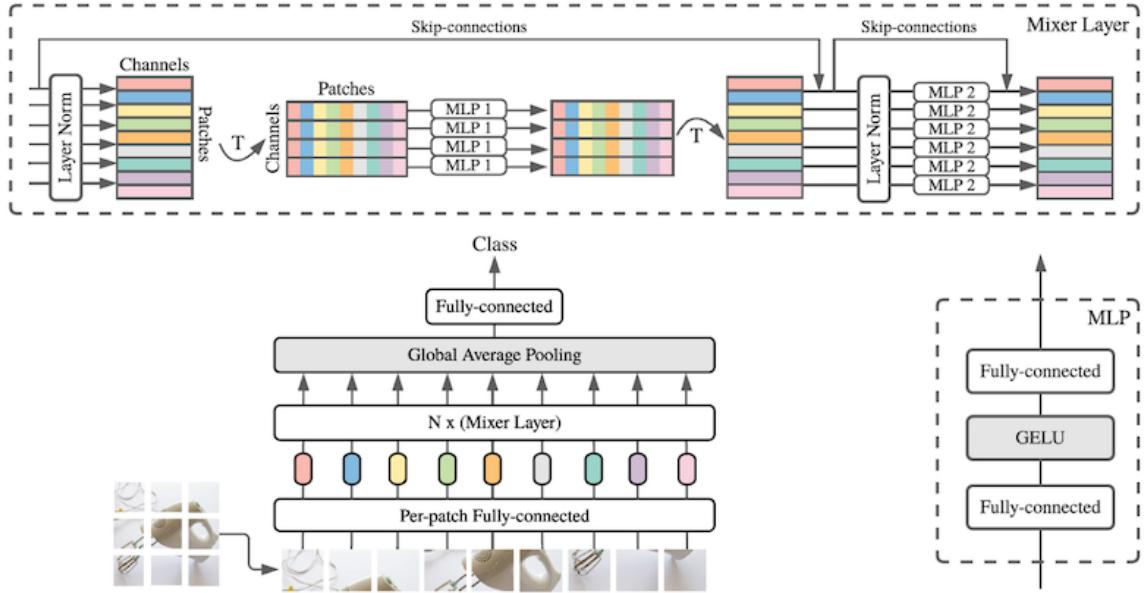
A lo largo de estos últimos años, cuando parecía que las redes convolucionales y/o los *transformers* se habían convertido en el *go-to model* en el mundo de la visión por ordenador a la hora trabajar con imágenes, investigadores del equipo de Google recientemente rescataron los *Multi Layer Perceptrons* (MLPs) para la tarea de clasificación de imágenes. A través de una arquitectura bautizada como MLP-Mixer [46], que emplea exclusivamente capas densas de neuronas, consiguen resultados comparables al *state-of-the-art* dado por los establecidos ViTs o CNNs.

La mezcla de las características extraídas es algo común en las tareas de visión, por ejemplo implícitamente en las CNNs de manera local al aplicar un kernel, pero también entre las diferentes salidas de las operaciones anteriores al emplear operaciones de *pooling* para combinar los mismos. Esto se hace a diferente escala en función del tamaño del kernel. En el MLP-Mixer, al no haber convoluciones, la combinación de estas características se realiza de manera separada uniendo la combinación de las características a través de los diferentes canales (*per-location*) y a lo largo de las diferentes coordenadas espaciales (*cross-location*). Además todo ello se hace en paralelo empleando la división en bloques de la imagen original.

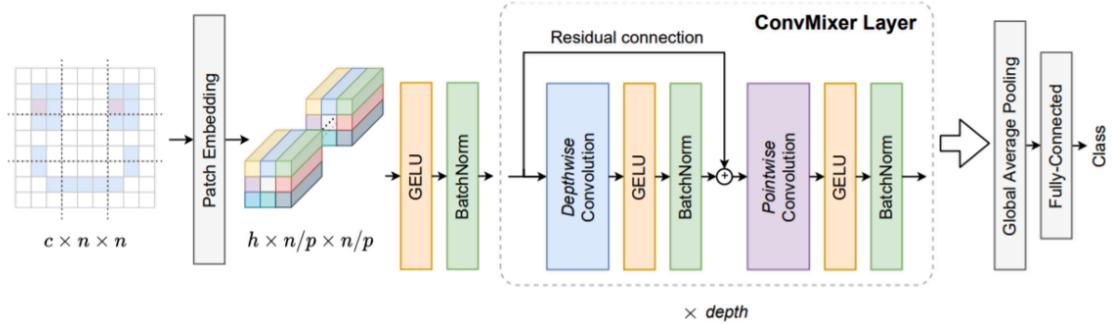
Sin entrar en mucho más detalle sobre esta arquitectura, haberla descrito es importante ya que sirve de inspiración para una nueva arquitectura similar, la cual incorpora convoluciones como elemento adicional característico. Esta nueva arquitectura llamada ConvMixer [47], muestra resultados comparables, e incluso mejores que las arquitecturas tipo ResNet y los ViTs en varios problemas de *computer vision* entre los que destaca ImageNet. Todo ello con una implementación especialmente sencilla, por lo que resulta especialmente atractiva para ser probada en este trabajo.

La ConvMixer hace uso de los ”*patch embeddings*” característicos de los *transformers*, pero sirviéndose exclusivamente de convoluciones para extraer el contenido de la imagen sin emplear ningún mecanismo de atención como los vistos en los ViTs.

La arquitectura de la ConvMixer, representada en la Figura 6, imita algunas de las características del MLP-Mixer tal y como se ha adelantado. Toma como entrada una imagen, la cual se divide en *patches* a través de una convolución usual del tamaño del *patch* (y mismo *stride*) hasta un número de canales *h*. Tras normalizar los *h* canales y pasar por una función de activación GELU, el resultado se hace pasar paralelamente



**Figura 5:** Estructura general del MLP-Mixer, extraído de [46].



**Figura 6:** Arquitectura original de la ConvMixer [47].

por varias copias de un mismo módulo (definido el número de copias en el parámetro *depth*) denominado ConvMixer.

Este bloque contiene en su núcleo una primera convolución 2D (*depthwise*, ver convoluciones separables [48]) con un tamaño de kernel *k*, estableciendo el número de filtros equivalentes a *h* y manteniendo las dimensiones de entrada mediante *padding*, seguido de una convolución *pointwise* 1x1, que también mantiene las dimensiones originales. Tras cada una de las convoluciones se realiza el mismo paso por la GELU y la normalización respectivamente.

Finalmente se realiza un *pooling* con el promedio sobre las salidas de los diferentes módulos ConvMixer y se conecta la imagen final con una red neuronal con un tamaño *h* para la capa oculta y con un número de neuronas de salida igual al número de clases del problema de clasificación en cuestión.

### 1.3. DRIVE dataset

El dataset DRIVE (Digital Retinal Images for Vessel Extraction)[\[49\]](#) consiste en un conjunto de datos de imágenes tomadas durante un programa de detección de retinopatía diabética en los Países Bajos. En total se tomaron 400 fotografías de la retina de diferentes adultos mediante una cámara de alta resolución. Del total de estas imágenes se seleccionó aleatoriamente una submuestra más reducida de 40 fotografías para su posterior etiquetado por los expertos, conformando así el dataset.

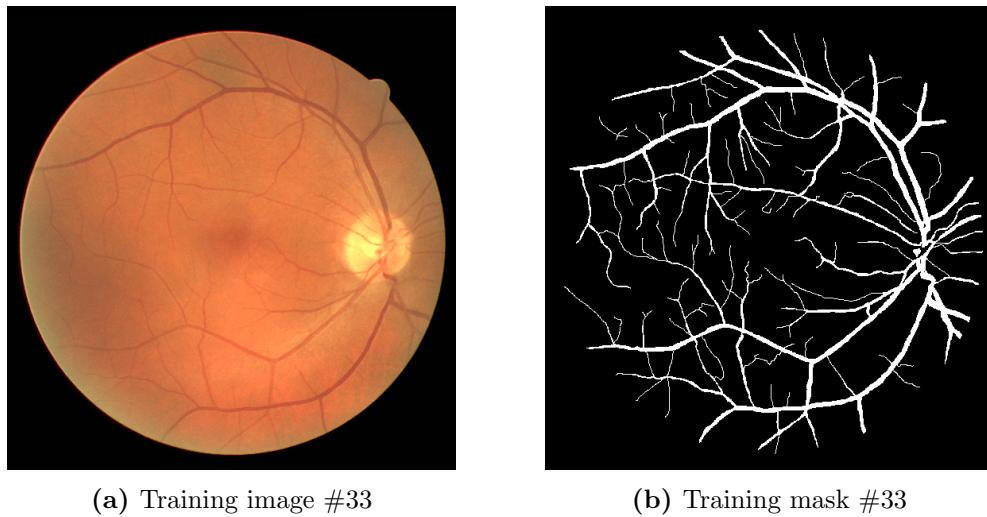
La retinopatía diabética está relacionada con ciertos atributos morfológicos de los vasos sanguíneos de la retina, como la longitud, el grosor, las ramificaciones de las mismas o los diferentes ángulos que forman estos vasos [\[50\]](#). En este contexto, para el diagnóstico, tratamiento o seguimiento de esta enfermedad, el análisis de estos vasos sanguíneos es fundamental. La única forma que tienen los médicos de extraer esta información es precisamente la inspección de la retina del paciente. Sin embargo, a efectos del ojo humano, se requiere un trabajo adicional para separar las componentes de la imagen que son relevantes de las que no lo son. El ojo del experto ha de realizar un procesado preliminar para aislar los vasos sanguíneos de la retina antes de poder analizar su morfología.

Automatizar esta tarea le permitiría al experto ahorrar un tiempo y esfuerzo podría concentrar en el complejo análisis en sí de la morfología de los vasos sanguíneos. Por tanto, la finalidad de este conjunto de datos es la de aislar los vasos sanguíneos del conjunto de la imagen retinal. Siguiendo este propósito, el etiquetado de estas imágenes mencionado anteriormente consiste en clasificar píxel a píxel la presencia (o no) de vaso sanguíneo en la retina.

Agrupando todos estos elementos, el conjunto de datos se compone de imágenes retinales de 584x565 píxeles con sus correspondientes máscaras, estas últimas aíslan los vasos sanguíneos presentes en la imagen. Estas 40 imágenes se han subdividido a su vez en un grupo de entrenamiento, en el que se proporciona tanto la imagen como su correspondiente máscara, y un grupo de test, del que sólo se dispone de la imagen pero no de la máscara.

Esta partición en train-test de los datos por parte de los propietarios de estas imágenes, proporcionando de manera pública las máscaras de las imágenes de entrenamiento únicamente, ha permitido la creación de un desafío global[\[50\]](#) con el que poner a prueba los diferentes modelos de los participantes, pudiendo evaluar el desempeño de los modelos sobre estas imágenes de test.

Este es uno de los motivos por el cual se ha elegido evaluar sobre este conjunto de datos el desempeño de los diferentes modelos empleados a lo largo de este trabajo. El disponer de una plataforma a modo de concurso en la que poder recibir retroalimentación sobre las predicciones del modelo permite, por un lado tener unas métricas reales y no sesgadas de manera optimista, por otro, tener referencias de trabajos de otros investigadores y científicos de datos en general, los cuales han elaborado sus propios modelos. Con todo, este conjunto de datos se ha convertido en una de las referencias para la comparativa de las diferentes técnicas en el campo de la segmentación.



**Figura 7:** Muestra de una de las imágenes de entrenamiento junto a su correspondiente máscara etiquetada por el experto. Ambas disponibles de forma pública para el desarrollo de modelos de segmentación.

## 2. Resultados

A continuación se presentan las diferentes aproximaciones que se han elaborado para la resolución general del problema de la segmentación en imágenes, haciendo hincapié en los resultados obtenidos sobre el dataset DRIVE [1.3] y, siguiendo en la línea de la introducción llevada a cabo en este trabajo, un orden de menor a mayor grado de complejidad. En primer lugar se presentan los resultados de algoritmos que quedan fuera del ámbito del “deep learning”, sirviéndonos como “benchmark” / “baseline” para el desarrollo de modelos más avanzados de deep learning, que pretenden lograr unos mejores resultados que estas aproximaciones de carácter más clásico.

Posteriormente, dentro de las alternativas posibles que emplean el deep learning para la segmentación de imágenes, se busca por un lado explorar y explotar las técnicas de preprocesado y “data augmentation” y, por otro, evaluar distintos modelos y arquitecturas con la finalidad última de conseguir la mejor métrica posible.

### 2.1. Aproximaciones “clásicas”

En primer lugar se evalúan algunos de los algoritmos mencionados en la introducción de este trabajo [1]. Como se ha comentado previamente, con muchas de estas aproximaciones no se pretende lograr unos grandes resultados en cuánto a métricas se refiere, sin embargo conviene poner en valor algunos de los métodos empleados en base a los resultados obtenidos, ya que, precisamente atendiendo al principio de parsimonia, en muchas ocasiones y problemas de la vida real, la simplicidad es uno de los atributos más importantes a valorar en los algoritmos, especialmente si existe una disputa entre varias alternativas y ambas son capaces de lograr resolver el mismo problema.

La gran mayoría de los métodos son de carácter no supervisado, por lo que no es

necesario dividir la muestra total de entrenamiento en conjuntos train-test para dar métricas relacionadas con el desempeño de los modelos. La idea general para estar aproximaciones es testear los algoritmos sobre una imagen en particular y comparar el resultado con su máscara. Si bien, para ser consistentes sería apropiado recorrer todas las imágenes del conjunto de entrenamiento y proporcionar un resultado promedio más robusto. Esto no será del todo necesario, ya que, a priori, únicamente con observar el resultado de la segmentación en una o varias de ellas, basta para extraer las conclusiones pertinentes sobre el desempeño de estos modelos, teniendo en cuenta la idea preconcebida de que fácilmente serán superados por las alternativas más complejas que se explorarán más adelante.

El primer algoritmo clásico puesto a prueba es el *Thresholding*. Para encontrar el threshold óptimo se ha realizado un ejercicio de inspección a través de los distintos canales (R, G y B) de la imagen. La Figura 8 ilustra los resultados obtenidos con esta técnica.

Para establecer el umbral del algoritmo, se ha buscado a través de los canales de la primera imagen (Imagen 1) el punto en el que se distinguen mejor los vasos sanguíneos al establecer un corte en uno de los canales. En este caso, se encuentra un intervalo en el canal “B” de la imagen en el que se consiguen diferenciar parte de los vasos sanguíneos. Se puede apreciar como incluso algunos pequeños vasos sanguíneos “secundarios” resaltan sobre el fondo ocular, pero a su vez también se puede notar cómo el thresholding es altamente sensible a la iluminación en la imagen y al ruido, por lo que no es posible distinguir los vasos sanguíneos de las zonas muy iluminadas, o contrariamente, pobremente iluminadas. Además quedan marcados como vasos sanguíneos, como es de esperar, muchos puntos aislados que entran dentro de la horquilla definida por el umbral.

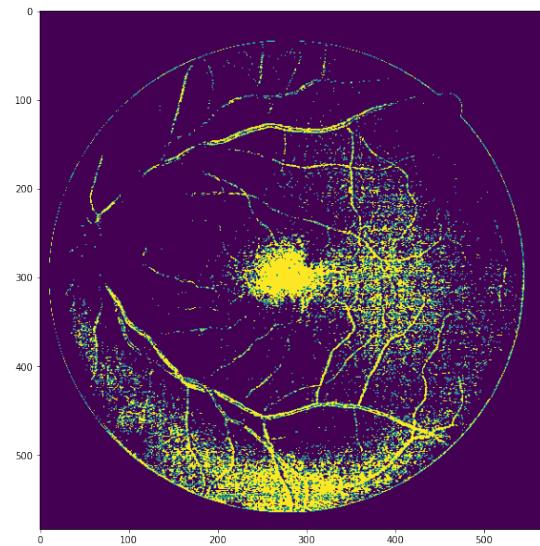
El principal problema que tiene emplear este tipo de algoritmo es, como se ha descrito, la alta sensibilidad a las alteraciones en la imagen en cuanto a cambios en el brillo, contraste, etc... Este fenómeno se pretende ilustrar en la misma Figura 8, cuando aplicamos este umbral a una segunda imagen (Imagen 2), el algoritmo ni de lejos es capaz de obtener resultados similares a la segmentación conseguida con la primera imagen. Sin profundizar sobre una muestra mayor de imágenes, se puede concluir que este algoritmo es altamente dependiente de las alteraciones en la intensidad de los distintos canales de la imagen de entrada, por tanto, no sirve para generalizar a un conjunto de imágenes con diferencias en la intensidad de los canales. Esto propicia que haya que diseñar uno a uno el threshold óptimo para cada imagen; proceso que es más costoso que la inspección manual de las imágenes por parte del experto, que es precisamente la tarea que se pretende acelerar, por lo que esta alternativa no es útil para esta aplicación en concreto. Ante esto, se opta por dar el siguiente paso hacia la exploración de los algoritmos de clustering.

## 2.2. Algoritmos de clustering

En primer lugar, dentro de los algoritmos de clustering se ha testeado es el K-means, introducido con anterioridad 1, principalmente porque dentro de los algoritmos de clustering es la alternativa más sencilla y rápida, por lo que suele ser común emplear



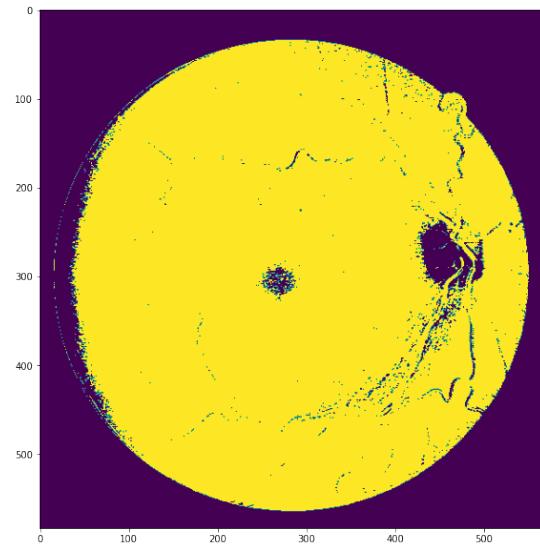
(a) Imagen 1 del dataset de entrenamiento



(b) Threshold aplicado a la Imagen 1



(c) Imagen 2 del dataset de entrenamiento

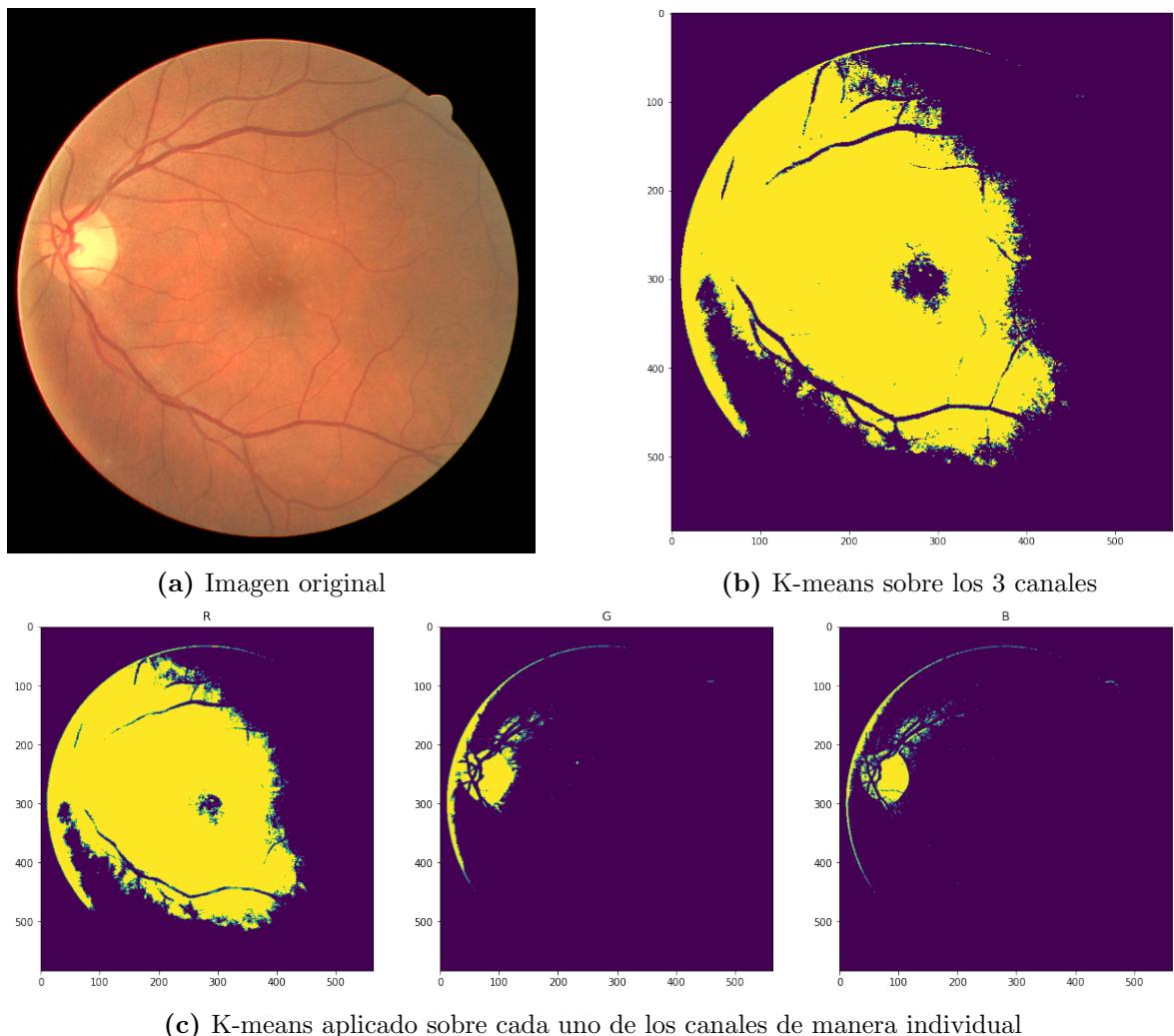


(d) Threshold aplicado a la Imagen 2

**Figura 8:** Algoritmo de umbral aplicado a dos imágenes aleatorias diferentes del conjunto de entrenamiento. El algoritmo distingue los vasos sanguíneo a través de un corte sobre el canal “B” de la imagen.

este algoritmo como una primera aproximación al problema, ya que muchas veces es suficiente como para resolverlo. El K-means es muy flexible y el único parámetro que manualmente siempre se ha de especificar es el número de clusters resultantes, que en el problema que nos ocupa serían 2 (vasos sanguíneos y resto). Sin embargo, precisamente esta flexibilidad hace que se pueda plantear el problema desde varias perspectivas; haciendo el clustering sobre todos los canales como un problema de 3 dimensiones, o, en su defecto, probando canal a canal, lo cual puede tener sentido si existe una diferencia relevante entre la intensidad del vaso sanguíneo y el resto de la imagen en un canal en particular.

Como se puede apreciar en la Figura 9, el algoritmo no es del todo capaz de hacer una distinción entre los vasos sanguíneos y el resto del ojo, esto ocurre tanto para



**Figura 9:** Resultados del clustering mediante el algoritmo K-means sobre una de las imágenes del dataset de entrenamiento, en particular, la Imagen 1 sobre la que se también se prueba el thresholding en la Figura 8 anterior.

el clustering sobre los tres canales como el individual para cada canal, de hecho, se puede apreciar como el canal “R” es el que contiene la mayor capacidad discriminante, mientras que los otros dos canales parecen no aportar información. Es importante destacar que el clustering se ha realizado explícitamente sobre la zona donde se sitúa el globo ocular, dejando al margen el contorno al que manualmente se le asigna la pertenencia al cluster de ”no vaso”.

Los resultados obtenidos en esta imagen confirman que el K-means no va a lograr una segmentación acorde a unos estándares mínimos para ser de utilidad al experto. Por tanto, se ha planteado el uso de otro algoritmo de la misma familia para comprobar si es posible obtener un mejor resultado con las técnicas de clustering.

DBSCAN [51] es un algoritmo basado en densidades muy utilizado en el ámbito del clustering. El algoritmo localiza regiones de alta densidad que están separadas una de la otra por regiones de baja densidad. Densidad, en este contexto, se define como el número de puntos dentro de un radio especificado. Precisamente, este radio es uno de

los parámetros a introducir manualmente, junto al número mínimo de puntos que ha de tener dentro del radio un punto para ser considerado “core point”. Los puntos core son aquellos que potencialmente definen el centro de un cluster, mientras que los puntos fronterizos no tienen el mínimo de puntos especificado dentro de este radio. Recorriendo todos los puntos y delimitando los clusters uniendo los puntos frontera a sus respectivos puntos core se forman los distintos clusters. El algoritmo permite encontrar cualquier forma arbitraria de cluster sin verse afectado por el ruido, lo cual es sumamente útil para muchas aplicaciones. Sin embargo concretamente para la segmentación, puede que este algoritmo no sea el más indicado, ya que no se puede especificar el número de clusters de forma manual y, en este caso, se necesita que el resultado arroje dos grupos únicamente. Aún así, estableciendo una combinación de hiperparámetros adecuada, puede que el algoritmo consiga limitarse a estos dos grupos.

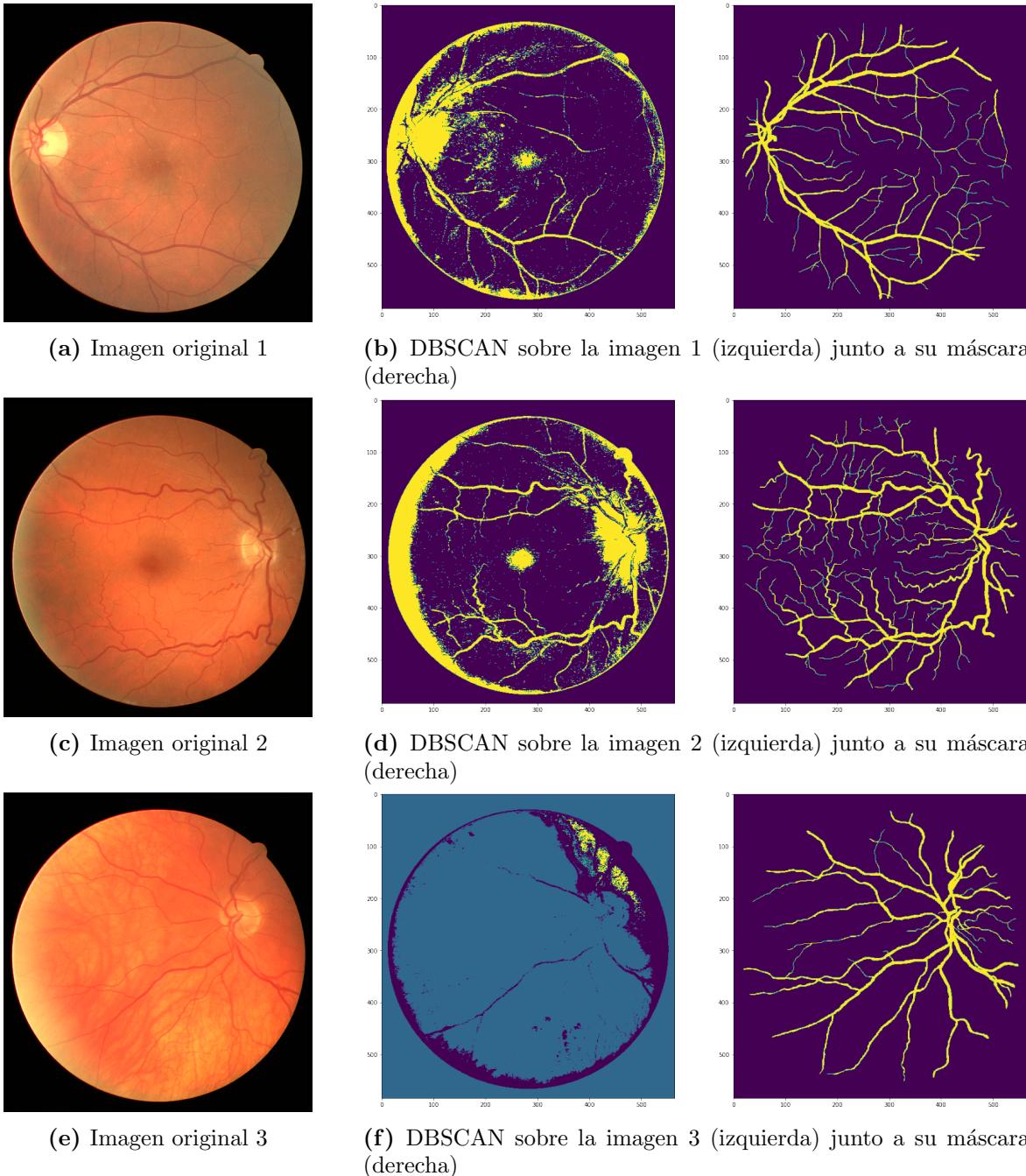
La Figura 10 muestra los resultados obtenidos con este algoritmo. Concretamente, se ha jugado con los dos hiperparámetros hasta obtener un resultado de dos agrupamientos sobre la Imagen 1 intentando que fuera el mejor posible. Posteriormente, se han mantenido fijos estos dos hiperparámetros optimizados para la Imagen 1 y se ha reentrenado el algoritmo con el resto de las imágenes. El primer punto que salta a la vista es el hecho que, en la tercera imagen 101, el algoritmo no consigue formar dos clusters y en su defecto forma 4, lo cual es motivo suficiente como para no considerar este algoritmo como candidato a resolver el problema de segmentación con estos datos. Sin embargo, es interesante ver como esta combinación de hiperparámetros funciona relativamente bien para las 2 primeras imágenes, especialmente para la segunda, ya los hiperparámetros solo se han optimizado para la primera. Si bien el algoritmo no es capaz de resolver el problema, es cierto que funciona considerablemente mejor que el K-means (ver Figura 9).

No es necesario seguir explorando alternativas de clustering, ya que la finalidad de estos algoritmos no es en última instancia la segmentación, por lo que no están diseñados para este propósito. Es interesante para el problema el poner a prueba estos modelos por si pudieran funcionar relativamente bien, en cuyo caso sería un buen punto de partida sobre el cual buscar una mejora a través de nuevos algoritmos. Sin embargo, al no obtener los resultados deseados con el clustering, se plantea dar el paso hacia nuevas aproximaciones que pudieran mejorar los mismos.

### 2.3. Otros algoritmos de machine learning

Por último, dentro de esta primera batería de algoritmos no pertenecientes a la categoría deep learning se ha querido evaluar el desempeño de algunos algoritmos de machine learning empleados principalmente en problemas de clasificación. En concreto, se plantea el uso de un clasificador Random Forest para clasificar píxel a píxel la pertenencia (o no) a un vaso sanguíneo.

Random Forest es, en términos generales, es un algoritmo de “bagging”, el modelo se compone de una combinación de varios algoritmos. Sin embargo, no basta con cualquier algoritmo, sino que se emplean “*weak learners*” como estimadores base para construir la predicción global 52. Estos *weak learners* son algoritmos que funcionan ligeramente



**Figura 10:** Resultados del clustering mediante el algoritmo DBSCAN sobre 3 imágenes del dataset de entrenamiento. Se ha fijado un radio de 3 ( $\text{eps}=3$ ) y un número mínimo de 500 puntos ( $\text{min\_samples}=500$ ) para todas las imágenes. Se presenta en cada fila de manera ordenada: la imagen original, el resultado de la segmentación y la máscara o *label* de esta imagen original.

mejor que la predicción puramente aleatoria. En el caso del Random Forest, los árboles de decisión son los que hacen de *weak learners*, siendo entrenados sobre una submuestra del conjunto de datos de entrenamiento (*bootstrap*).

El uso de este algoritmo como aproximación al problema de segmentación se debe a varios motivos. En primer lugar, en referencia al Random Forest, suele conseguir buenos

resultados de manera general [1], lo cual se debe a que, tal y como demostró Schapire [54], la combinación de estos *weak learners* produce en su conjunto un “*strong learner*” con un rendimiento superior.

Además, existen ciertas ventajas estadísticas como, por ejemplo, la resistencia a valores anómalos. La varianza se reduce al promediar las predicciones sobre todos los *weak learners*, que de manera aislada sí pueden tener un alto sesgo y varianza. Como estos *weak learners* se entranan con una muestra reducida del conjunto de entrenamiento, cada uno de ellos trabaja con un subconjunto reducido del total de las columnas de entrenamiento. Esto hace que se pueda tratar con conjuntos de datos con un gran número de variables, manteniendo la rapidez que caracteriza a estos algoritmos más sencillos. Gracias a esto último, se permite a su vez otorgar una cierta medida de la importancia de las distintas variables.

La Figura [11] muestra algunos resultados obtenidos en la clasificación con este algoritmo. Para poder proporcionar unas métricas que acompañen los resultados mostrados en la figura, se ha seguido una metodología de *Cross Validation* sobre el conjunto de entrenamiento, esto es, dividir las imágenes en conjuntos aleatorios y escoger y reservar una submuestra como ”test”, a la par que se alimenta el modelo con el resto de imágenes. Repitiendo este proceso con tantos ”folds” de entrenamiento y test como se indique, es posible proporcionar una métrica robusta del desempeño del modelo.

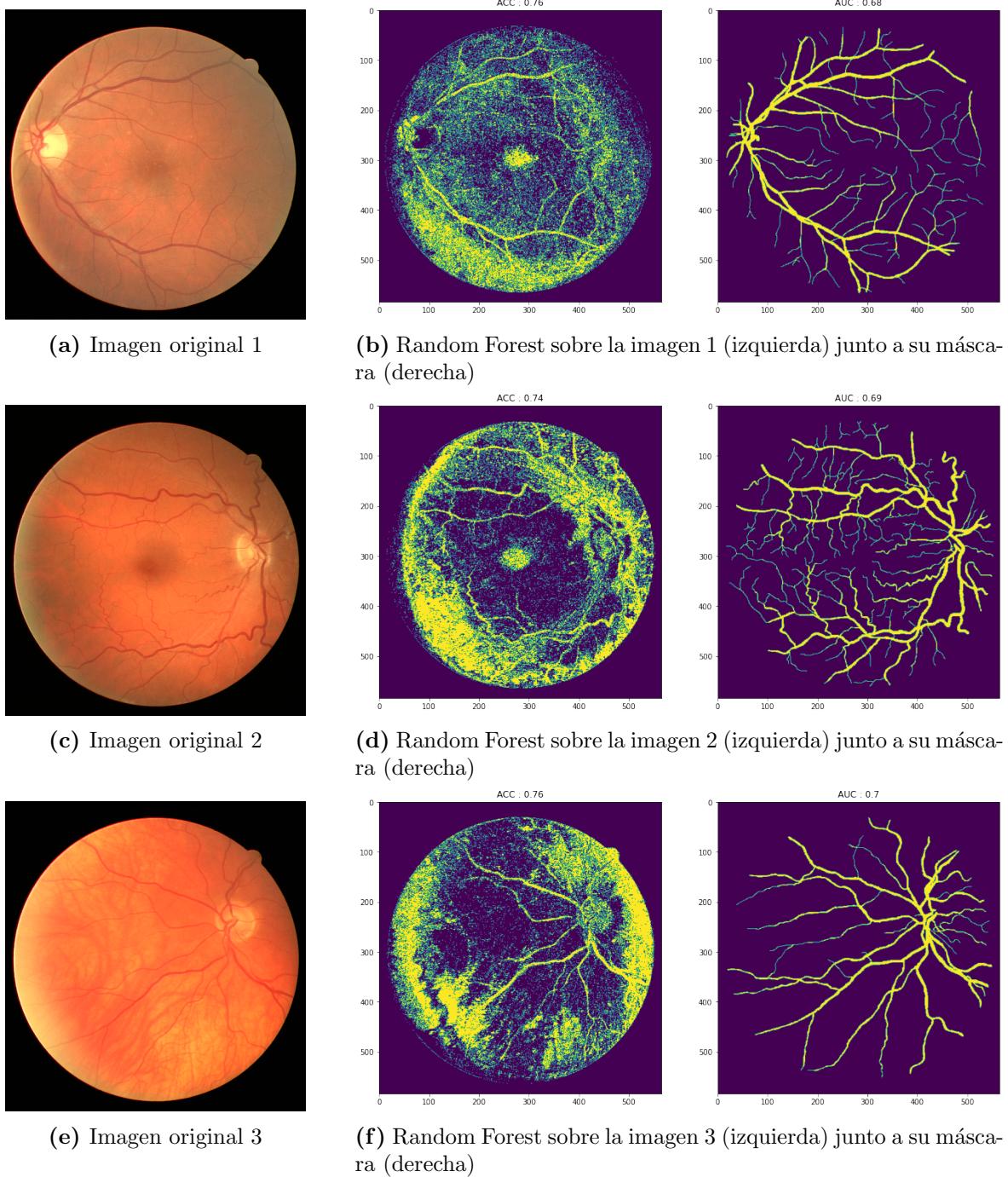
Random Forest	Fold 1	Fold 2	Fold 3	Fold 4	Global
Accuracy	0.642	0.603	0.769	0.793	0.702
AUC	0.660	0.670	0.571	0.664	0.641

**Tabla 1:** Resultados obtenidos aplicando 4-fold CV. Se muestran los resultados en cada *fold*, así como la media del *accuracy* y el AUC obtenidos.

Se ha partido de un subconjunto de 16 imágenes de entrenamiento sobre las que se realiza un ”4-fold”, por lo que el modelo entrena con 12 imágenes y proporciona una métrica relativa a las 4 restantes, repitiendo este mismo proceso 3 veces adicionales, hasta que todas las imágenes han formado parte del test set. Los resultados obtenidos se presentan en la Tabla [1]. Los resultados que aparecen en la Figura [11] en cambio, forman parte de un caso particular en el que se entrena de igual modo con 12 de las imágenes de entrenamiento, pero se ha excluido del mismo conjunto las imágenes de la Figura [10], de esta manera, se permite una comparación directa con los algoritmos previos, a la par que se mantienen válidas las métricas que se están proporcionando en la Tabla [1] para este caso en particular.

Para entrenar el algoritmo de Random Forest se ha establecido una serie de pesos en las clases, en concreto, a la categoría de ”vaso sanguíneo” se le otorga mayor peso que a su complementaria, por un factor igual a la proporción en la que esta clase está infrarrepresentada con respecto a la mayoritaria. De esta manera, se evita un sesgo en el algoritmo hacia la clase mayoritaria que, en este caso, sería la ausencia de vasos sanguíneos. Por tanto, se sacrifica el *accuracy* para ganar en otras métricas como la

<sup>1</sup>Esto no quiere decir que sea la mejor alternativa para cualquier problema. Recordando el teorema ”no free lunch” [53], no existe un algoritmo que sea universalmente mejor que los demás, pues siempre habrá casos donde funcione peor que otros.



**Figura 11:** Resultados del Random Forest aplicado a 3 imágenes distintas con las que el modelo no ha sido entrenado. En la parte superior se indica el *accuracy* (ACC) y el AUC obtenido para cada imagen.

precisión, lo cual es más que razonable en esta clase de problemas. A través de los pesos, se logra un mejor resultado<sup>2</sup> en el AUC.

Los resultados obtenidos, que se resumen tanto en la Figura 11 como en la Tabla 1, evidencian el amplio margen de mejora en la calidad de la máscara que produce

---

<sup>2</sup>Repetiendo este 4-fold con el mismo modelo pero sin asignar pesos a las clases se obtiene un AUC promedio de 0.56.

el Random Forest. Si bien algunos de los vasos “principales” son distinguibles del resto del globo ocular, existen muchos píxeles aislados que son también clasificados incorrectamente como vasos y que aportan una sensación de ruido a la máscara final. El algoritmo parece no funcionar del todo bien, esto puede deberse a que el número de *features* disponibles para el entrenamiento únicamente son 3, los canales de la imagen, esto no da mucho margen para entrenar los *weak learners* de forma que sean lo más heterogéneos entre sí. Sin embargo el problema principal es que este algoritmo no tiene en cuenta la información de la vecindad de los píxeles y trata de manera individual a cada uno, por eso parece cubrir todo el espacio de píxeles aislados introduciendo este molesto ruido que no permite distinguir los auténticos vasos sanguíneos.

Se hace necesario emplear algoritmos que tengan en cuenta que un píxel marcado como vaso sanguíneo no es solo un punto aislado independiente, sino que ha de tener algún tipo de prolongación. La forma de introducir este concepto de vecindad en el algoritmo es empleando redes neuronales convolucionales, que como se ha introducido en secciones anteriores [1], son la base de la visión por ordenador. Mediante estas operaciones de convolución, es posible captar precisamente la información de la localidad en la imagen, por lo que los algoritmos de segmentación que emplean filtros convolucionales están mucho más especializados en aplicaciones con imágenes. En este punto, pese a que puede seguir tratando de mejorar las métricas con algoritmos de bagging sometidos a un “fine tuning” o probando otras aproximaciones como por ejemplo los algoritmos de boosting, conseguir una mejora sustancial con respecto a este “baseline” establecido sigue la línea de explotar las características de las redes convolucionales. Por este motivo, el siguiente paso lógico lleva a introducirse en un nuevo paradigma, el deep learning.

## 2.4. Deep learning

Como se ha introducido anteriormente [1.2], el desarrollo de las redes neuronales convolucionales propició el desarrollo del deep learning en el campo de la segmentación. La popularidad de las distintas arquitecturas que han ido surgiendo con el paso del tiempo en este campo, ha desplazado a todos los algoritmos vistos hasta este punto hacia niveles inferiores en una escala de rendimiento, en la que en la parte superior se hallaría el “state-of-the-art”. En la actualidad, revisando el estado del arte [55] en las tareas de segmentación en imágenes y sus variaciones, en todos los casos los primeros puestos son ocupados por redes que entrarían dentro de la categoría deep learning.

Los resultados previos obtenidos han establecido un punto de partida, una visión genérica sobre la segmentación en imágenes que puede servir como aproximación general a cualquier dataset, pudiendo lograr en última instancia mejores o peores resultados. Los algoritmos mostrados hasta este punto son relativamente sencillos de implementar en cualquier problema de similares características. Con sus fortalezas y debilidades, permiten dar cuenta de la dificultad real del problema en cuestión y, en muchos casos, dependiendo del problema puede que baste con alguno de ellos para resolverlo, por lo que es aconsejable evaluar en un primer lugar estos algoritmos antes de adentrarse en el campo de deep learning, en el que los modelos son más costosos de diseñar, entrenar y optimizar. Para los datos de DRIVE en concreto, los resultados obtenidos con estos

algoritmos distan de conseguir el objetivo final con el que se ha diseñado este conunto de datos. Hasta ahora, en ningún caso se ha invertido mucho tiempo en “exprimir” el potencial de los modelos para lograr la mejor métrica posible, ya que el salto cuantitativo en las métricas al emplear modelos de deep learning iba a superar con creces cualquier modelo optimizado. Sin embargo, de ahora en adelante se otorga más relevancia a las métricas, tratando de lograr en primera línea el resolver el problema; en segunda, el hacerlo con la mejor métrica posible. En esta línea, los resultados obtenidos sobre imágenes reservadas para validación, y en especial, las métricas que ponen a disposición en el challenge de DRIVE [1.3] a través de los *submits*, son la referencia principal a la hora de comparar el rendimiento de los distintos modelos.

#### 2.4.1. U-Net

La U-Net pese a ser diseñada para el ámbito de la biomedicina, ha demostrado buenos resultados en otros campos con el paso del tiempo [56] [57] [58]. La principal ventaja que tiene este modelo es el no necesitar una gran cantidad de imágenes de entrenamiento para lograr buenos resultados. Aprovechando las técnicas de data augmentation, por ejemplo a través de deformaciones elásticas en la imagen, es posible construir un conjunto de datos de mayor volumen a partir de pocas imágenes, lo cual es útil en ámbitos como la biomedicina, en los que etiquetar las imágenes puede ser un proceso especialmente costoso.

Para entrenar la U-Net, se han combinado técnicas de data augmentation sobre las imágenes de entrenamiento junto a ciertas prácticas concretas en el propio proceso de entrenamiento que permiten conseguir unos mejores resultados. Mediante estas técnicas se han desarrollado dos modelos que comparten la misma arquitectura y muchas características similares, sin embargo, con el segundo modelo se ha tratado de exprimir al máximo las técnicas de data augmentation para exprimir al máximo esta arquitectura.

Para su caso de uso en el *dataset* particular sobre el cual se ha evaluado la *performance* de este modelo se han realizado algunas ligeras modificaciones en la arquitectura de la red. En particular, se ha modificado la red para aceptar cualquier tamaño de imagen de entrada y devolver a la salida una máscara de un solo canal, con las mismas dimensiones que la imagen de entrada. Además, se ha incorporado una interpolación para evitar problemas en el caso de introducir imágenes con dimensiones impares a la hora de concatenar los diferentes *tiles* de imágenes en las *skip connections*.

En cuanto al detalle de las técnicas empleadas en el proceso de entrenamiento, en primer lugar se ha optado por dividir el conjunto total de imágenes de entrenamiento (20 imágenes) en un conjunto propiamente de entrenamiento con 16 imágenes, y otro de validación con las 4 imágenes restantes. En segundo lugar, habiendo reservado un conjunto de imágenes únicamente para la validación, se ha establecido un criterio de guardado de los pesos del modelo al superar una métrica de referencia sobre este conjunto. De esta manera, se evita el sobreajuste a los datos de entrenamiento a la par que se tiene un historial de las métricas de validación para cada época de entrenamiento. La métrica de control sobre la cual se guardan estos pesos es el coeficiente de Sørensen-

Dice, o simplemente el DICE, que se puede expresar como:

$$DICE = \frac{2 |X \cap Y|}{|X| + |Y|} \quad (1)$$

Esta métrica suele ser habitual en los problemas de segmentación ya que representa la similitud entre dos muestras. En concreto, se evalúa la fracción entre el área de solapamiento entre la predicción y el *ground truth* sobre el área de unión de las mismas. A nivel cuantitativo es equivalente al conocido *F1 score*. DICE, además de ser robusto al desbalanceo de las clases (al contrario que el *accuracy*), es la métrica principal que se evalúa en el *leaderboard* del concurso, por lo que tiene sentido escoger esta métrica como criterio de guardado de pesos.

Otra técnica empleada en el entrenamiento ha sido la asignación de pesos a las clases en la función de pérdida del modelo (*Binary Cross Entropy with logits*, ver la implementación [59]). En concreto, se le otorga un mayor peso a la clase minoritaria para evitar el sesgo de predecir con mayor frecuencia la categoría mayoritaria.

Por último, en ambos modelos se ha empleado como optimizador el algoritmo Adam [60] con una técnica para establecer el *learning rate* llamada “Reducción del *learning rate on plateau*”, que consiste en mantener un learning rate fijo hasta que, tras un cierto número de iteraciones las métricas no mejoran. En este caso se establece un LR de partida y se reduce en un factor 0.5 tras 100 épocas sin lograr una mejora, todo ello con un umbral máximo a partir del cual se queda fijo y no continúa reduciéndose. Este *set-up* con la división en train-test de las imágenes para el entrenamiento, así como el criterio de guardado de pesos y la reducción del *learning rate* se ha mantenido a lo largo del entrenamiento de todos los modelos vistos en este estudio.

Sin contar de momento con las especificaciones propias del bucle de entrenamiento como el número de épocas o el tamaño del *batch* de imágenes, un punto clave que repercute en el desempeño del modelo es el diseño de las técnicas de data augmentation. De cara a evitar el sobreajuste y contar con un mayor número de muestras con las que entrenar el modelo, el uso de estas técnicas para crear datos “artificiales” es fundamental. Esto mismo se demuestra al comparar los dos modelos finales seleccionados para incluirse en este trabajo; el primero de ellos cuenta con algunas transformaciones más generales, mientras que para el entrenamiento del segundo de ellos se ha tratado de explotar al máximo estas técnicas, añadiendo nuevas transformaciones y optimizando las que ya se habían empleado.

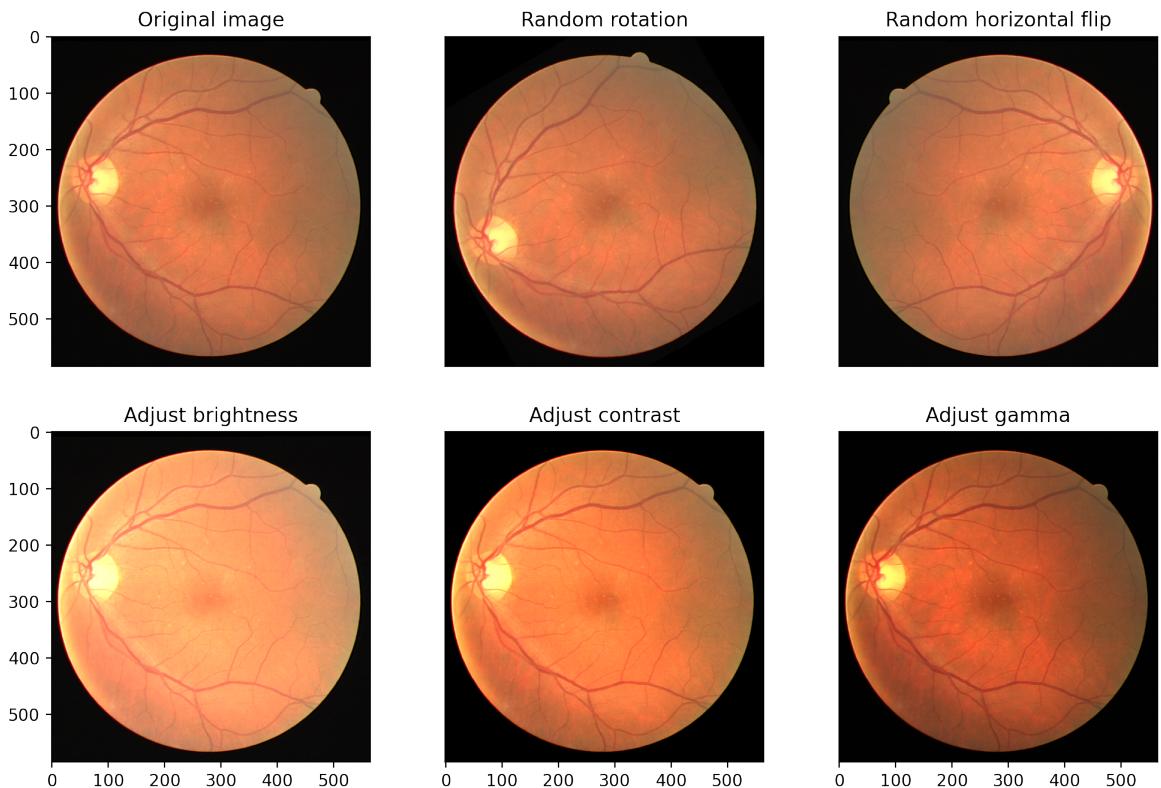
En esta primera iteración, dentro del conjunto de las transformaciones realizadas a las imágenes de entrenamiento, la Tabla 2 incluye la descripción individual de todas las posibilidades. En el entrenamiento, a menudo se combinan varias de ellas sobre una misma imagen.

<sup>3</sup>La transformación es una operación no lineal que se usa para codificar y decodificar luminancia. Un factor gamma mayor a 1 oscurece las sombras, mientras que una gamma < 1 las ilumina, lo cual representa la siguiente fórmula:

$$I_{out} = 255 * I_{in} * gain * \left( \frac{I_{in}}{255} \right)^\gamma$$

Transformación	Parámetros	Descripción
Random rotation	$\theta \in [-\pi, \pi]$	Realiza una rotación aleatoria un ángulo $\theta$
Random horizontal flip	$p = 50\%$	Realiza con probabilidad “p” una reflexión con respecto al eje vertical
Adjust brightness	$\alpha \in [0.8, 1.2]$	Ajusta el brillo de la imagen un factor $\alpha$ , siendo 1 el brillo original de la imagen
Adjust contrast	$\beta \in [0.8, 1.2]$	Ajusta el contraste de la imagen un factor $\beta$ , siendo 1 el contraste original de la imagen
Adjust gamma	$\gamma \in [0.9, 1.1]$ $gain = 1$	Realiza una corrección gamma <sup>3</sup> de la imagen, $\gamma = 1$ mantiene la imagen original

**Tabla 2:** Listado de transformaciones realizadas a las imágenes durante el entrenamiento del primer modelo basado en la U-Net.

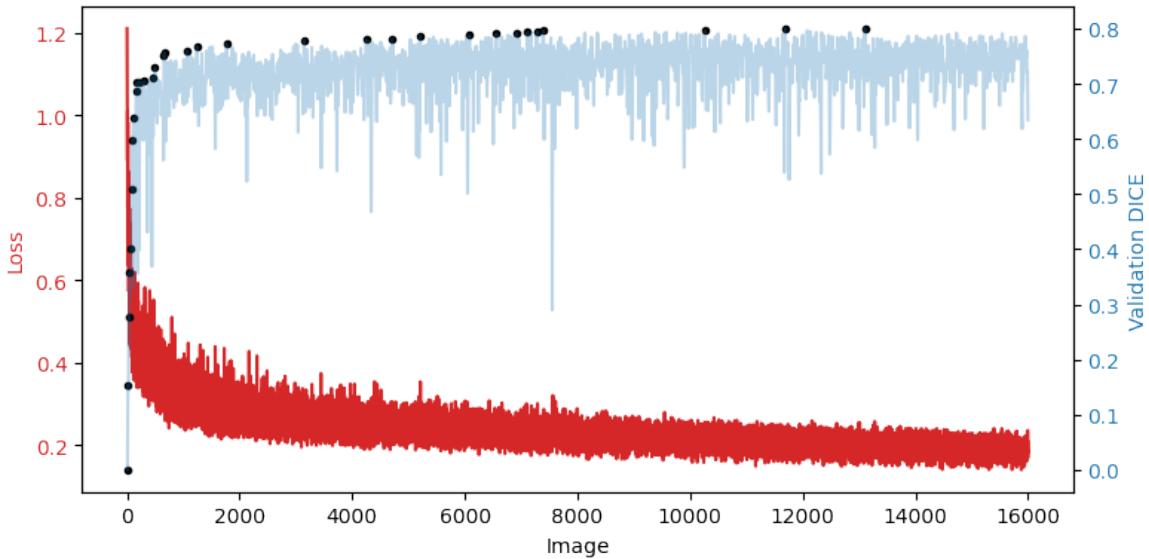


**Figura 12:** Representación de las transformaciones descritas en la Tabla 2 sobre una de las imágenes del conjunto de entrenamiento. Se ha empleado un factor de 1.4 para los ajustes de brillo, contraste y gamma en la imagen, de esta manera, las deformaciones sobre la imagen original son más extremas y puede apreciarse el efecto de las mismas.

Combinando la información de la Tabla 2 y la Figura 12 se puede visualizar y entender la forma de enriquecer el conjunto de datos de entrenamiento. Las transformaciones no se aplican de manera consecutiva, sino que se ha optado, en primer lugar, por aplicar sistemáticamente la rotación con un ángulo aleatorio, esto es, porque el problema de segmentación de los vasos deber ser invariante ante rotaciones; seguidamente la reflexión con respecto al eje vertical, la cual se realiza con un 50 % de posibilidades; mientras que,

en último lugar, se elige con la misma probabilidad una única transformación entre las que varían la intensidad de la imagen. La elección sistemática de una de estas últimas transformaciones viene a que, si el parámetro es cercano a 1, la transformación apenas modificará la imagen original, por lo que tampoco es necesario alimentar al modelo con las imágenes originales.

Por último, atendiendo al propio bucle de entrenamiento, se ha optado por entrenar a lo largo de 1000 épocas, empleando un tamaño de *batch* de 2 imágenes, principalmente debido a los límites de la propia GPU. Al reservar 4 de las imágenes para la validación, al final de cada época se computan varias métricas, entre ellas el DICE [1], que sirve para guardar los pesos según el criterio establecido.



**Figura 13:** Representación de la curva de entrenamiento del primer modelo. Los puntos de color negro indican el guardado de los pesos y coinciden con el mayor DICE logrado en validación hasta esa iteración.

La Figura [13] muestra la curva de entrenamiento de este primer modelo, en ella se puede apreciar que, hasta prácticamente el final del entrenamiento, se sigue mejorando el resultado en validación, por lo que se puede confirmar que el modelo efectivamente no solo está entrenando sino que no sobreajusta a los datos de entrenamiento.

El último *checkpoint* se encuentra en torno al final del entrenamiento alrededor de la época número 800, con un  $DICE \approx 0.8$ . Tener los pesos de la red en este punto, permite reconstruir las predicciones y las métricas sobre el conjunto de validación, así como predecir sobre las imágenes test que se evalúan en la competición.

La Figura [14] en conjunto con la tabla [3] permiten tener una visión completa del desempeño de este primer modelo de segmentación basado en la U-Net. En comparación con los métodos anteriores, este modelo supera con un amplio margen de diferencia a todos ellos tal y como se había anticipado. Con respecto a las métricas obtenidas, los resultados en validación con la muestra de 4 imágenes que se habían reservado con este propósito y los resultados en test, a cuyas máscaras no se tiene acceso, son muy similares, lo cual refuerza la premisa de que el modelo es capaz de generalizar. Por otro lado, aunque las métricas en validación sean una buena guía para seleccionar el

<b>Val accuracy</b>	<b>Val AUC</b>	<b>Val DICE</b>	<b>Test DICE</b>	<b>Leaderboard</b>
0.964	0.984	0.800	0.794	1332

**Tabla 3:** Resumen general de los resultados obtenidos con el primer modelo entrenado basado en la U-Net. La puntuación sombreada representa el DICE obtenido en las predicciones sobre las 20 imágenes test en la plataforma de la competición de DRIVE [50]. El *leaderboard* representa la posición en el concurso cuando se realizó la subida de las predicciones para este modelo.

mejor modelo, las métricas sobre los datos el concurso son “la prueba de fuego” para valorar los distintos modelos. Por tanto, encontrar similitudes entre los resultados en validación y en test es algo positivo a destacar.

En contraste con el resto de modelos, esta aproximación basada en la U-Net por primera vez establece un punto de partida real sobre el que tratar de mejorar para seguir escalando posiciones en el *leaderboard*. En referencia esto, de ahora en adelante con los sucesivos modelos e iteraciones sobre este conjunto de datos se ha buscado refinar y optimizar los modelos para tratar de lograr el mejor DICE posible.

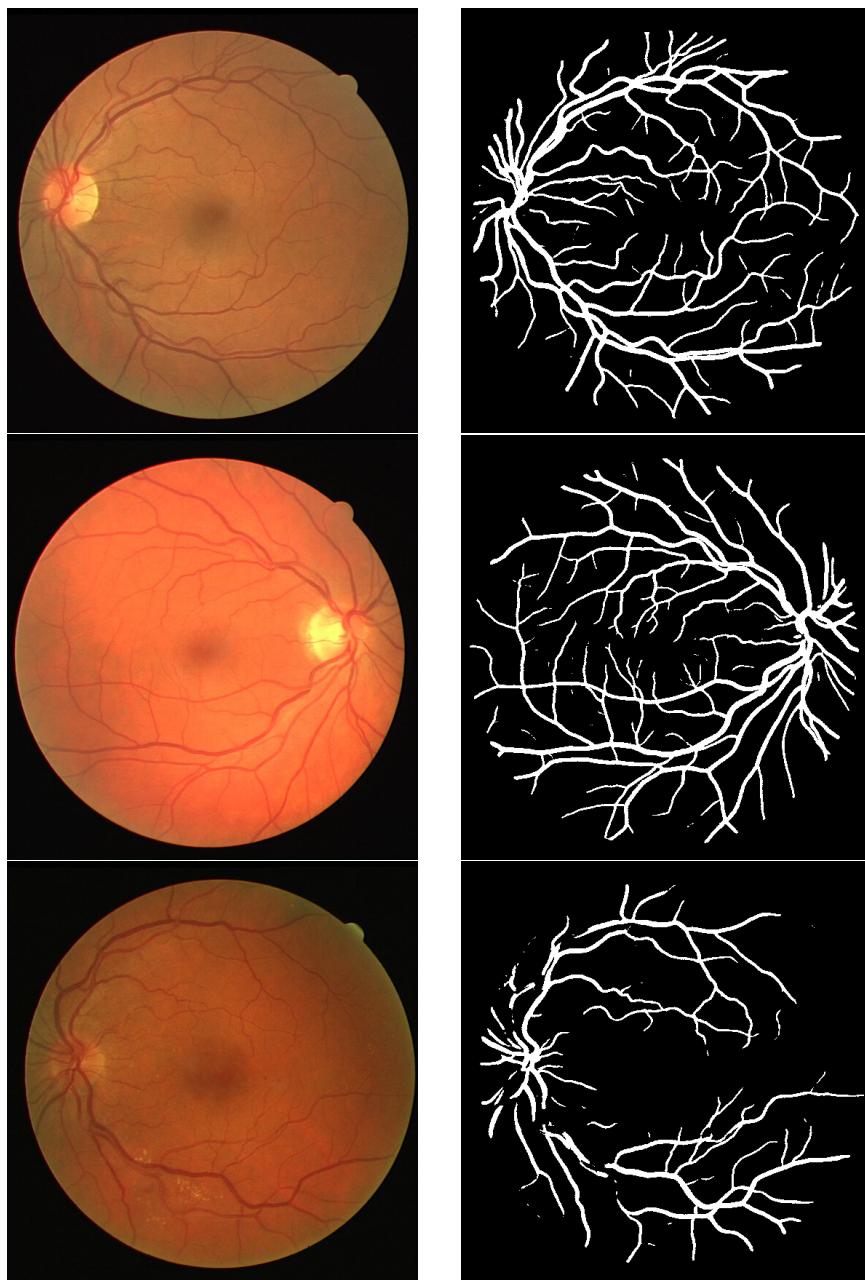
La primera línea de mejora planteada para tratar de conseguir el mejor resultado, no únicamente con esta arquitectura, sino de manera compartida por el resto de sucesivas aproximaciones, es tratar de afinar las técnicas de data augmentation.

### Explorando los límites del data augmentation

Con el modelo anterior, se han realizado las modificaciones de la Figura 12 a las imágenes de entrenamiento. Sin embargo estas transformaciones son bastante genéricas y no se diseñan para este problema en específico. Revisando la literatura [61], se encuentra que otros autores han realizado multitud de transformaciones a las imágenes de entrenamiento que, en particular, el modelo anterior 2, no contemplaba. Entre todas ellas el “zoom” sobre la imagen parece tener especial sentido especialmente si se observan los resultados de la Figura 14. Las predicciones, por lo general, parecen ser precisas sobre los vasos gruesos, sin embargo, el mayor nivel de dificultad se haya en la correcta clasificación de los pequeños vasos sanguíneos que se ramifican desde los conductos principales. Por otro lado en ocasiones algunos píxeles marcados como vasos, quedan como puntos aislados en la imagen sin estar conectados con ningún vecino, en este sentido parece que el posible ruido en la imagen podría afectar a las predicciones, por lo que se ha introducido, a su vez, en esta segunda iteración una transformación que añade ruido gaussiano sobre los píxeles para evitar esto.

Introduciendo estas nuevas transformaciones en el conjunto de entrenamiento, se ha reentrenado el modelo de la U-Net por segunda vez y se ha logrado obtener unos mejores resultados. La tabla 5 recoge estas métricas.

Los resultados de la tabla 4 muestran una mejora sustancial con respecto al modelo inicial 3 propuesto de la U-Net. Esta mejora se puede considerar significativa especialmente debido al alto escalado de posiciones en el ranking, situándose el modelo en una posición superior al percentil 90 % del total de modelos evaluados en el concurso.

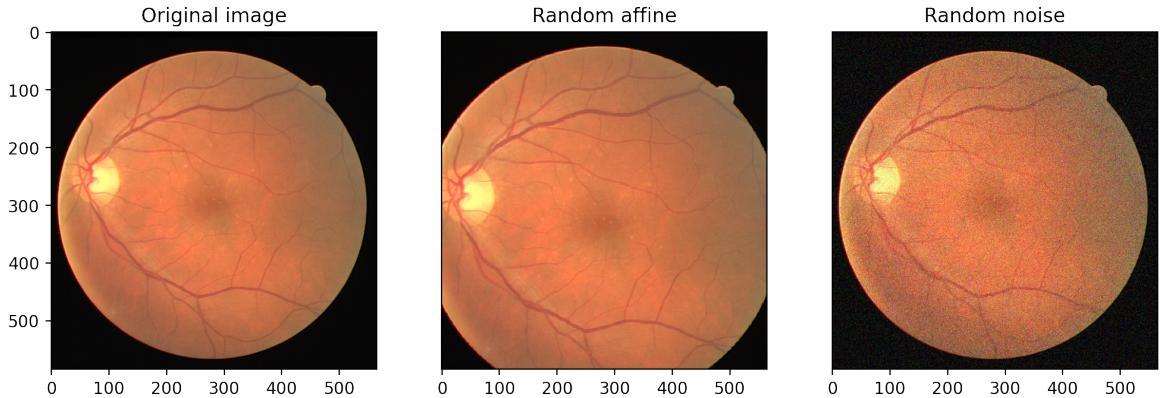


**Figura 14:** Predicciones del primer modelo basado en la U-Net sobre las 3 primeras imágenes del conjunto test. El modelo obtiene las métricas mostradas en la tabla 3, sin embargo, para estas 3 imágenes en concreto, el DICE obtenido es, empezando por la imagen superior, 0.774, 0.831 y 0.741 respectivamente.

Por otro lado, el acercamiento a las altas posiciones del ranking y la pequeña diferencia en el DICE entre los modelos con las mejores métricas, parece ser indicativo de estar aproximándose cada vez más a los límites para este problema. De hecho, revisando el *benchmark* [62] de la competición, en el que aparecen las “*submissions*” con mejores scores y cuya metodología ha sido publicada por sus autores, en particular para la U-Net [61], las métricas consideradas “techo” hasta la fecha son prácticamente iguales a las logradas con este segundo modelo 5.

Transformación	Parámetros	Descripción
Random affine	$\alpha \in [1, 1.2]$ $d = 10\%$	Realiza un zoom de $\alpha$ aumentos sobre la imagen, y desplaza la ventana sobre el eje vertical hasta un $d\%$ del alto de la imagen
Add random noise	$p = 30\%$ $\sigma = 0.1$ $\mu = 0$	Añade ruido a cada píxel en base a una distribución normal $N(\mu, \sigma)$ con una probabilidad “p”

**Tabla 4:** Nuevas transformaciones añadidas a las existentes [\[2\]](#) en esta segunda iteración sobre el modelo de la U-Net.



**Figura 15:** Representación de las transformaciones descritas en la Tabla [\[4\]](#) sobre una de las imágenes del conjunto de entrenamiento.

Val accuracy	Val AUC	Val DICE	Test DICE	Leaderboard
0.969	0.987	0.812	0.820	406

**Tabla 5:** Resumen general de los resultados obtenidos con el segundo modelo entrenado basado en la U-Net, el cual añade nuevas transformaciones [\[4\]](#) a los datos de entrenamiento

Por este motivo, al haber hallado un modelo capaz de resolver el problema y habiendo puesto al límite las capacidades del mismo, a continuación se plantea la búsqueda de una nueva arquitectura capaz de superar a esta última.

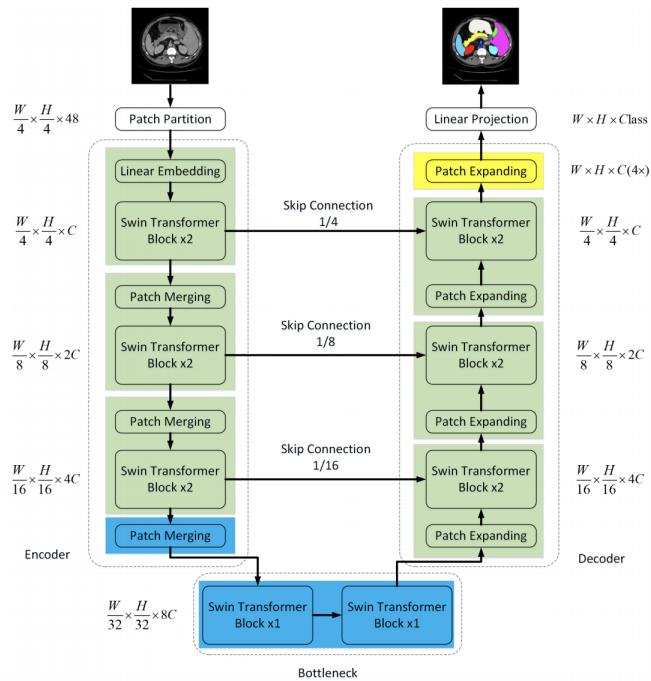
#### 2.4.2. Transformers

Con el fin de tratar de seguir mejorando las métricas en el problema a través de una nueva arquitectura más reciente, se plantea emplear un enfoque alternativo a las convoluciones usuales. La motivación tras el uso de los *transformers*, y más concretamente de los *vision transformers*, viene de la mano de su reciente popularidad en el campo del *computer vision* [\[63\]](#).

En particular se han probado dos diferentes arquitecturas basadas en *transformers* para tratar de mejorar los resultados obtenidos hasta ahora. Pese a que el esquemato común que comparten estas arquitecturas ha sido descrito con anterioridad [\[1.2.2\]](#),

existen diferencias entre ellas que conviene describir de cara a explicar los resultados obtenidos.

La primera arquitectura considerada se basa en el modelo llamado Swin-Unet [64]. Esta arquitectura imita la famosa estructura de tipo “U” característica de la U-Net, que tan ampliamente ha sido utilizado en aplicaciones médicas. Bajo el pretexto de la incapacidad de las CNNs de captar las dependencias semánticas a grandes escalas debido a la localidad de las operaciones de convolución, se propone el uso de *transformers* para dar cuenta de estas dependencias. En particular, se propone una estructura que se denomina ”*pure transformer*”, lo que significa que las operaciones de convolución quedan completamente eliminadas en esta arquitectura y únicamente se utilizan módulos o bloques basados en *transformers*.

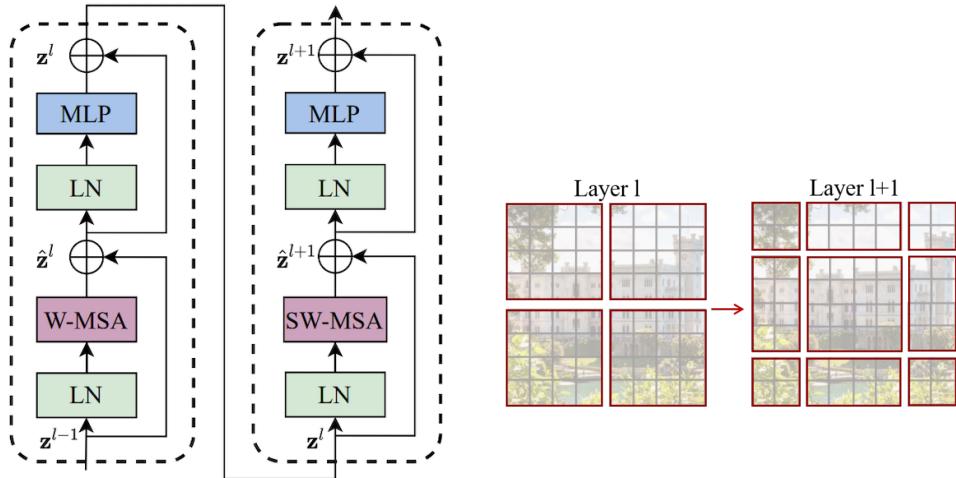


**Figura 16:** Esquema de la arquitectura general de la Swin-Unet sacado del *paper* original [64].

La Figura 16 ilustra la arquitectura de esta red. La estructura es idéntica a la de la U-Net 2 pero se presentan nuevos bloques llamados “Swin Transformer Blocks” que reemplazan a las convoluciones usuales. Las operaciones que se realizan en estos bloques fueron introducidas y descritas por el equipo de Microsoft [65] como mejora a la aproximación basada 1.2.2 en el cómputo del -attention sobre los *patches* de la imágenes de una resolución fija en los ViTs.

La imagen se divide inicialmente en *patches*, en el caso del *paper* original de tamaño 4, por lo que tenemos para cada uno de ellos un vector de tamaño  $4 * 4 * 3 = 48$ , siendo 3 el número de canales. Este vector se convierte a través de un embedding lineal en otro de dimensión  $C$  que, junto al resto de *patches*, serán la entrada a los bloques de atención.

En una primera iteración se calcula la atención en ventanas no solapadas de la forma



(a) Dos bloques *Swin* anidados que componen los pasos tanto en la parte de *downsampling* como en el *upsampling* en la Swin-Unet.

(b) Ilustración del proceso de *window shifting* por el que se introduce la vecindad en los mecanismos de *self-attention* al solapar las ventanas.

**Figura 17:** Esquema para explicar el funcionamiento de los bloques llamados “*Swin-Transformers*”.

usual<sup>4</sup> (*multi-head attention* 1.2.2), con las matrices Q, K y V y las operaciones pertinentes para cada ventana. Posteriormente se introduce esta nueva variante en la que, aprovechando el cálculo anterior, a través de lo que los autores llaman “*cyclic shift*”, se solapan los vectores de atención calculados parcialmente sobre otras ventanas, haciendo que sucesivas ventanas ya no sean independientes entre sí y estén conectadas a través de sus vecinos al recalcular la atención con estas ventanas solapadas. Esto mismo es lo que representa la Figura 17, donde este primer cálculo sobre las ventanas no solapadas se abrevia como *W-MSA*, mientras que *SW-MSA* se refiera al cálculo tras solapar las ventanas (Shift Window). El método de *cyclic shift* simplemente es la manera eficiente de aprovechar la computación en *batch* para realizar el cálculo sobre todas las ventanas.

Por tanto, dada la Figura 17a, en el paso a través de este bloque se producen las siguientes operaciones<sup>5</sup>:

$$\hat{z}^l = \text{W-MSA} (\text{LN} (z^{l-1})) + z^{l-1} \quad (2)$$

$$z^l = \text{MLP} (\text{LN} (\hat{z}^l)) + \hat{z}^l \quad (3)$$

$$\hat{z}^{l+1} = \text{SW-MSA} (\text{LN} (z^l)) + z^l \quad (4)$$

<sup>4</sup>Se incluye un término adicional en la función *softmax* llamado *Bias Matrix*, que tiene en cuenta el hecho de ser más flexibles al no forzar el uso de embeddings posicionales y, simplemente, dejar que el modelo aprenda de por sí la distancia entre los tokens.

<sup>5</sup>W-MSA y SW-MSA hacen referencia a *Window-based Multi-head Self Attention* y *Shifted Window-based Multi-head Self Attention* respectivamente. “LN” es una Normalización Lineal, mientras que MLP es un perceptrón de 2 capas con una función de activación no lineal GELU.

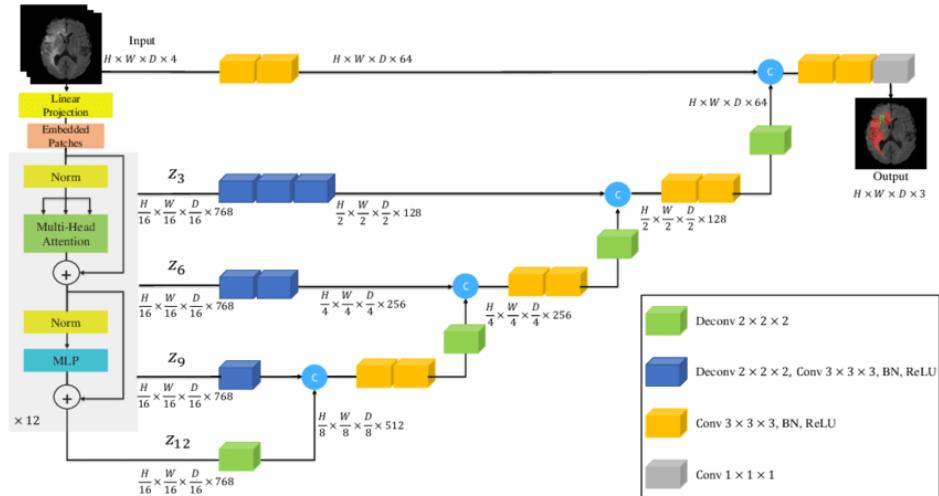
$$z^{l+1} = MLP \left( LN \left( \hat{z}^{l+1} \right) \right) + \hat{z}^{l+1} \quad (5)$$

Por último, las operaciones de *patch merging* en el *downsampling* y *patch expanding* en el *upsampling* son formas de, a través de una capa lineal de neuronas, reducir o aumentar respectivamente la dimensión de la concatenación de los vectores de los patches. Por ejemplo el vector producto de la concatenación inicial de dimensión 4C, pasa a reducir su dimensión a la mitad, 2C tras la primera unión.

La adaptación del código original [64] para esta red al problema de segmentación retinal es sencilla, simplemente se trabaja con imágenes de mayor dimensión, las cuales han sufrido “zero padding” por simplicidad hasta hacerlas cuadradas con una dimensión de 576x576 píxeles, además de especificar el número final de neuronas a uno al solo haber una clase como tal.

En paralelo al entrenamiento de esta red y su evaluación para el conjunto de datos de referencia, se ha considerado el uso de otra arquitectura que también utiliza este tipo de *transformers* para resolver este problema de segmentación. En concreto, la UNETR [66] es la segunda arquitectura que se ha entrenado dentro del campo de los *vision transformers*. Esta arquitectura presenta diferencias con respecto a la anterior, sin embargo, ambas son presentadas prácticamente al mismo tiempo, con excelentes resultados en tareas médicas de segmentación respectivamente, por lo que explorar ambas alternativas puede dar una visión más completa sobre los resultados dentro de los *vision transformers*, al no limitarse únicamente a un mismo tipo de arquitectura.

La principal diferencia de esta red con respecto a la anterior, es que la UNETR no es una arquitectura puramente basada en *transformers*, sino que conserva la parte convolucional usual en el *decoder*, mientras que emplea *transformers* únicamente para la extracción de características o *downsampling part*. De esta manera, se espera capturar las dependencias globales a lo largo de la extracción de características que se lleva a cabo en el encoder, mientras a su vez se mantiene el resto de la arquitectura de la UNET original que tan buenos resultados consigue en este tipo de problemas.



**Figura 18:** Esquema de la arquitectura general de la UNETR sacado del *paper* original [66]

La Figura 18 muestra el esqueleto de esta red de forma ilustrativa. Otras diferencias con respecto a la Swin-Unet como puede apreciarse en la estructura es precisamente el uso del sistema de *Multi-head Self Attention* basado en ventanas no solapadas exclusivamente, por lo que no incorpora la interacción entre ventanas vecinas en el sistema de atención, además de emplearse embeddings posicionales, por tanto, siguiendo una estructura en el encoder idéntica al *paper* original de los ViTs [45].

La adaptación del código en este caso es idéntica a la realizada para la puesta a punto de la Swin-Unet solo que, además, al estar esta red originalmente programada para imágenes 3D, se han cambiado las convoluciones a 2D para que pueda aceptar las imágenes de DRIVE. El esquema de la Figura 18 es válido para la adaptación de la red simplemente no considerando la dimensión adicional “D”.

Por tanto, considerando estas dos arquitecturas y sus modificaciones para adaptarlas al problema retinal, se han entrenado distintas configuraciones de ambas redes, manteniendo las técnicas que data augmentation que han demostrado su buen funcionamiento, modificando el resto de hiperparámetros como el tamaño de las ventanas sobre las que se calcula la atención o rediseñando funciones de coste que pudieran mejorar el entrenamiento de los modelos. Por ejemplo, se ha diseñado una función de coste que combina el DICE al uso con la versión con pesos de Binary Cross Entropy (BCE) bajo un parámetro ajustable para dar mayor, o menor, peso a una de las dos componentes según la fórmula:

$$Loss = \alpha * DICE + (1 - \alpha) * BCE_{logits} \quad (6)$$

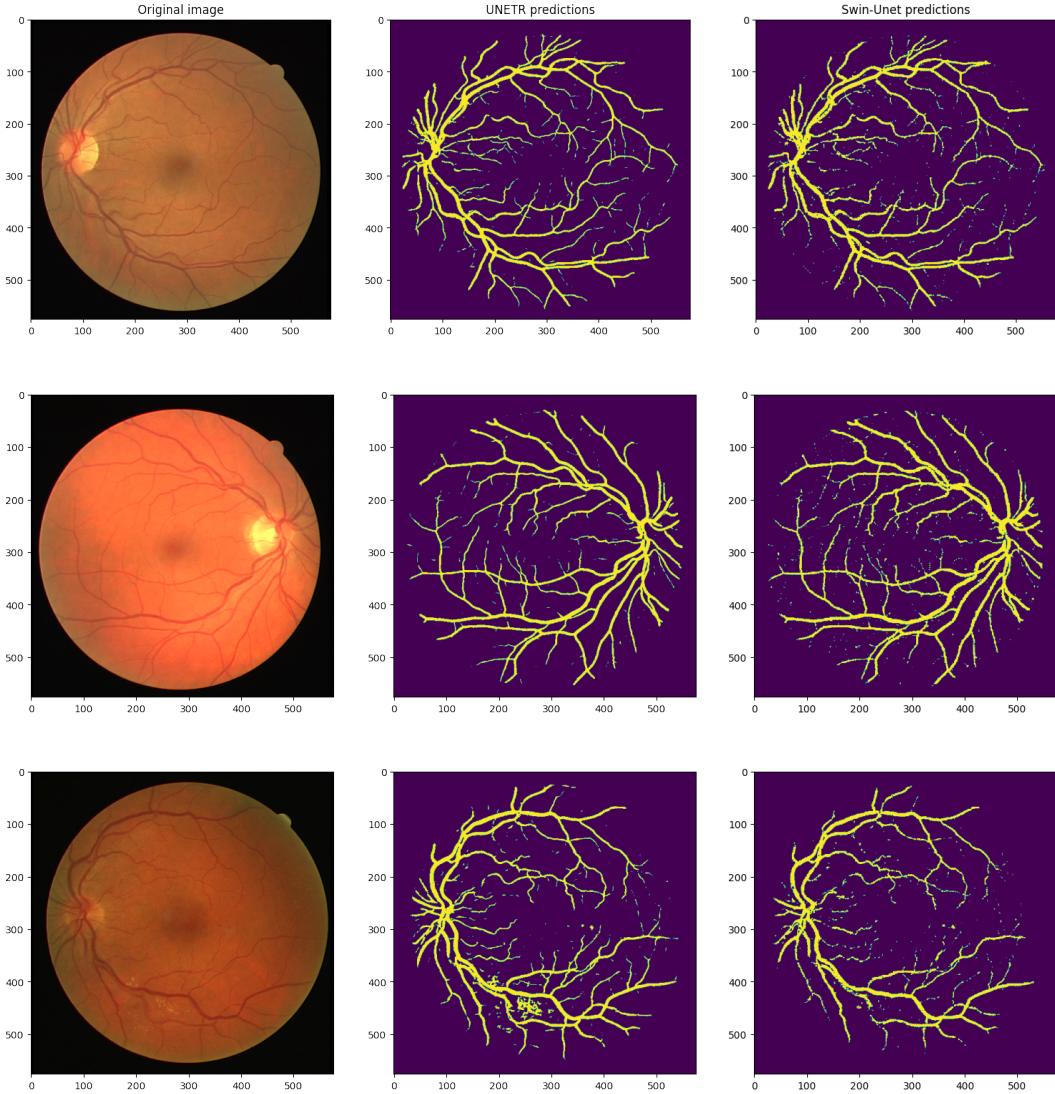
De esta manera, en el caso límite en el que  $\alpha$  toma el valor de 0, se usa el *Binary Cross Entropy* ponderado, mientras que si se fija a 1, la función de pérdida se convierte en el DICE.

Model	Val accuracy	Val AUC	Val DICE	Test DICE	Leaderboard
Swin-Unet	0.959	0.920	0.756	0.771	1737
UNETR	0.932	0.966	0.798	0.805	990

**Tabla 6:** Resumen de los resultados obtenidos con la selección de los mejores modelos para la Swin-Unet y la UNETR respectivamente.

La Tabla 6 muestra los resultados finales de los mejores modelos obtenidos tras probar distintas combinaciones con los elementos o hiperparámetros disponibles para modificar, mientras que la Figura 19 permite comparar las predicciones de los dos modelos sobre una muestra del test. La primera y más simple lectura que se puede hacer de los resultados es negativa en cuanto a no haber podido superar las métricas obtenidas con el mejor modelo de la U-Net (Tabla 5). Para ser capaces de explicar estos resultados, conviene detenerse a analizar con detalle los motivos por los cuales no se ha logrado obtener unas mejores métricas.

Comenzando por el modelo de la Swin-Unet, el índice DICE se sitúa significativamente por debajo del conseguido con el mejor modelo. La arquitectura de la red se basa en el *stack* de *transformers* a modo de *encoders* y *decoders* imitando la forma convencional



**Figura 19:** Predicciones del modelo entrenado de la UNETR (izquierda) frente a la Swin-Unet (derecha) sobre las tres primeras imágenes del conjunto test [14].

de la U-Net, pero en esta red no existe ninguna operación de convolución.

Uno de las principales factores que caracteriza a las CNNs es el verse afectadas por el denominado “*spatial inductive bias*” [67]. Estas redes asumen que existe una cierta relación espacial en las imágenes, lo cual introduce un sesgo. Éste puede ser beneficioso, ya que al introducir este conocimiento a priori se restringe de alguna manera la distribución espacial de los datos, los cuales mientras sean consistentes con estas asunciones, pueden inducir a mejores resultados de manera general sobre otros modelos que no poseen este sesgo, es el caso de las imágenes y las CNNs frente a los MLPs. Además, esto también permite a los modelos alcanzar un rendimiento superior con relativamente pocos datos (“*high floor*”), justamente como sucede con la U-Net.

La otra cara de la moneda se observa en el caso en el que se dispone de una buena cantidad de datos con la que “exprimir” la flexibilidad de los modelos. Al haber reducido la flexibilidad al introducir este sesgo, el modelo queda capado superiormente en cuanto

a rendimiento (“*low ceiling*”), perdiendo la capacidad de generalizar más allá de estas asunciones. En particular, las CNNs sufren de este sesgo, que no les permite capturar dependencias a grandes escalas debido a la intrínseca localidad de la operación de convolución.

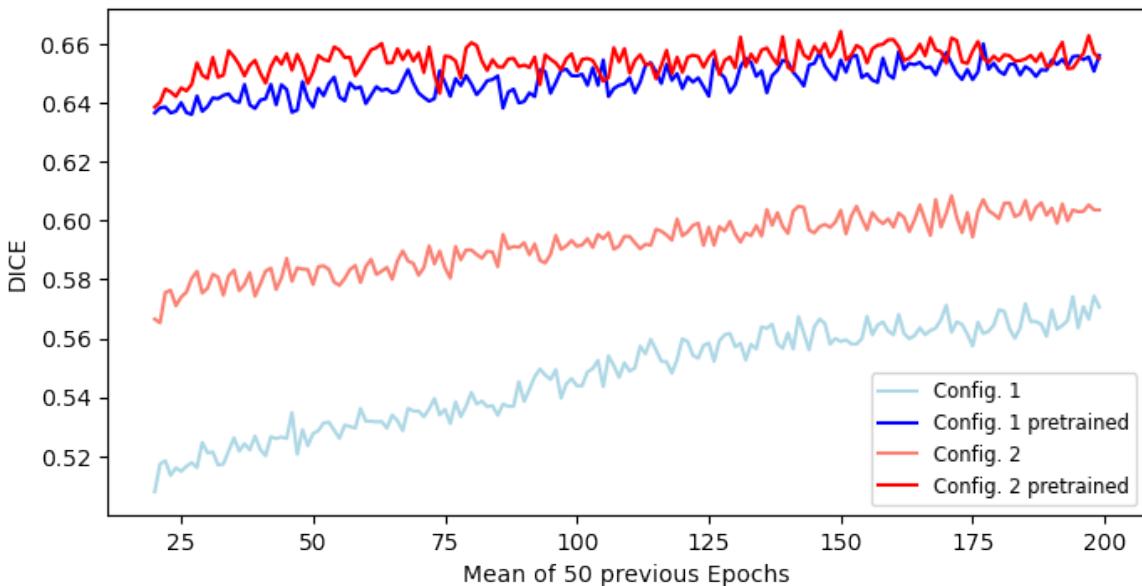
La introducción de los *transformers* viene motivada justamente con el fin de evitar este *bias* que presentan las convoluciones. De manera natural, los *transformers* no asumen esta localidad y presentan sesgos mínimos [68] (“*minimal inductive biases*”). Por ejemplo, en los embeddings posicionales de los mismos, los parámetros son aprendidos por la red y no necesitan de recurrencia, al contrario que en las LSTMs o RNNs que son sensibles a la posición de los tokens. Esta flexibilidad permite a éstos brillar en regímenes en los que hay una gran cantidad de datos con los que optimizar todos estos parámetros (*high ceiling*), pero a su vez, su rendimiento es muy limitado en las situaciones en las que escasean los datos (*low floor*), quedando por completo a merced de las técnicas de regularización y, sobretodo, *data augmentation*.

Por tanto, la alta dependencia en el número de muestras de entrenamiento en los *transformers* es un factor limitante de cara a plantear el entrenamiento en problemas con pocos datos de una de estas arquitecturas con el fin de superar a las redes convolucionales. Justamente éste es el motivo por el cual las métricas de estas arquitecturas no logran superar a la versión convolucional. De hecho, como indican Bhojanapalli et al. [69] “*When the training set is small, the ViTs are less robust compared to ResNets of comparable sizes, and increasing the size of the ViTs does not lead to better robustness*”.

Por tanto, puede parecer entonces poco recomendable el uso de estas arquitecturas en el contexto médico, siendo bastante común el no disponer de muchas imágenes etiquetadas (como en el conjunto de datos de DRIVE) debido al alto coste asociado a la obtención de las mismas. La forma más común de lidiar con este problema es precisamente el preentrenamiento de estas arquitecturas sobre un conjunto de datos, para luego, con los pesos inicializados, reentrenar sobre el conjunto de datos médicos en cuestión (“*fine-tunning*”).

Precisamente esto es lo que reportan los autores de la Swin-Unet [65], asociando los resultados obtenidos en su trabajo a un primer entrenamiento previo sobre la base de datos de ImageNet. Sin embargo, pese a que el preentrenamiento parece condición necesaria para lograr el rendimiento óptimo del modelo, debido al alto coste computacional que supone el incrementar el tamaño de la red hasta poder aceptar imágenes de la resolución del problema (584x565) junto al preajuste del modelo sobre las imágenes de ImageNet escaladas a este mismo tamaño, se hace inviable el preentrenamiento con el fin de exprimir las capacidades de esta arquitectura para este problema.

Pese a ello, con el fin de visualizar el efecto que tendría la inicialización de los pesos en la red preentrenada frente al entrenamiento desde cero, se han tomado los pesos de esta misma red que proporcionan los autores y se han escalado las imágenes del conjunto de entrenamiento junto a sus máscaras hasta una resolución compatible con la arquitectura preentrenada (224x224). Se han entrenado dos modelos con distintas configuraciones de hiperparámetros, ambos con su versión preentrenada en ImageNet y con los pesos inicializados aleatoriamente respectivamente.



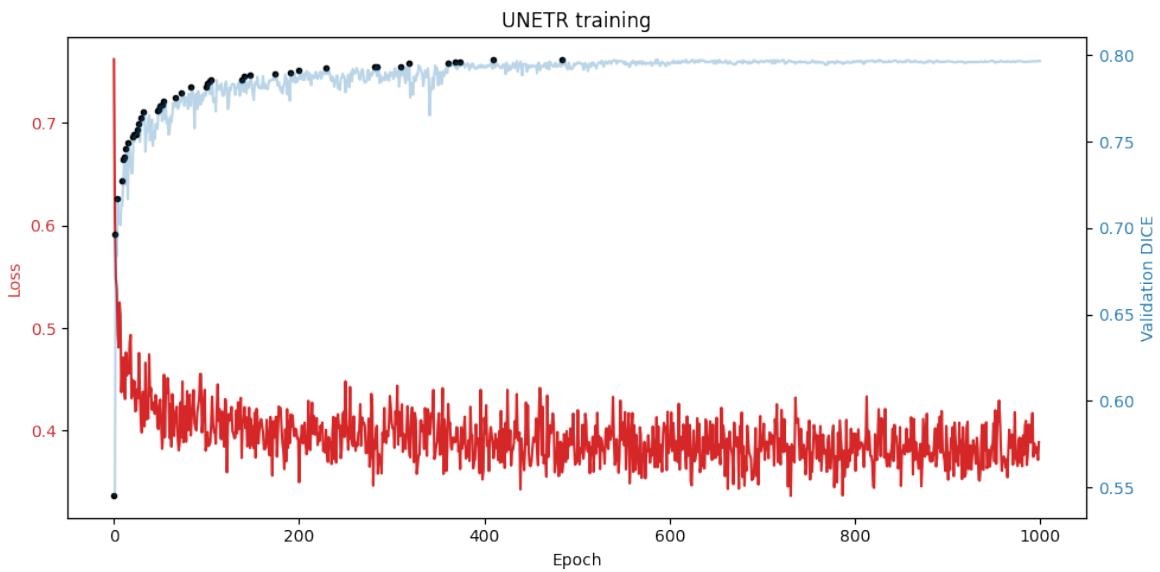
**Figura 20:** Representación del efecto del preentrenamiento en la Swin-Unet a través de las curvas de entrenamiento de dos configuraciones de hiperparámetros distintas. Para facilitar la visualización se ha omitido la parte inicial del entrenamiento y se ha tomado la media en ventanas de 50 epochs.

La diferencia en el DICE entre las curvas con y sin preentrenamiento es notable, por lo que el beneficio del preentrenamiento de la Swin-Unet es significativo. Partiendo de un modelo preentrenado y con un mayor número de imágenes de entrenamiento disponibles es seguro que se lograría mejorar las métricas en el problema, tal vez incluso superando a los resultados conseguidos con la U-Net.

La UNETR en cambio, no se sustenta por completo en *transformers* como sucede con la Swin-Unet, sino que combina éstos con convoluciones para obtener las correspondientes ventajas de cada arquitectura. Los resultados [6] muestran un mejor desempeño que el modelo de la Swin-Unet, precisamente por el uso de CNNs en la arquitectura. Al incorporar las convoluciones en el *decoder* del la red, se reduce la flexibilidad con respecto a la arquitectura basada en su totalidad en los *transformers*, esto parece llevar a mejores resultados, de hecho, los autores [66] afirman no necesitar el preentrenamiento en ImageNet para esta arquitectura ya que no conlleva un incremento significativo en los resultados.

Por tanto, pudiendo prescindir del preentrenamiento del modelo, el único factor que parece estar afectando al modelo para no poder superar a la U-Net es el bajo volumen de muestras de entrenamiento, lo cual es consistente con los resultados mostrados por los autores. Ellos emplean como referencia para presentar resultados varios conjuntos de datos [70] que forman parte de una competición con diversas tareas de segmentación. Entre ellas, se puede ver que en la segmentación del bazo (*Spleen CT segmentation*) consiguen superar ligeramente a la U-Net con muy poca diferencia, tan solo un 1% en el DICE. Esta diferencia se acentúa conforme más muestras de entrenamiento se dispone, por ejemplo, en la segmentación de tumores cerebrales en el que se dispone de +400 imágenes de entrenamiento esta diferencia asciende hasta casi un 10% en el DICE para la segmentación de regiones intratumorales, frente a la pequeña diferencia

en el problema anterior con una muestra de tan solo 41 imágenes de entrenamiento. No es de extrañar por tanto, que con tan solo 16 imágenes de entrenamiento, menos de la mitad que el conjunto de datos más pequeño con el que los autores presentan sus resultados, no se consiga aprovechar la flexibilidad de la arquitectura y pronto se alcance la saturación de la curva de entrenamiento pese a las técnicas de data augmentation y la regularización. La Figura 21 justamente muestra la curva de entrenamiento de la UNETR.



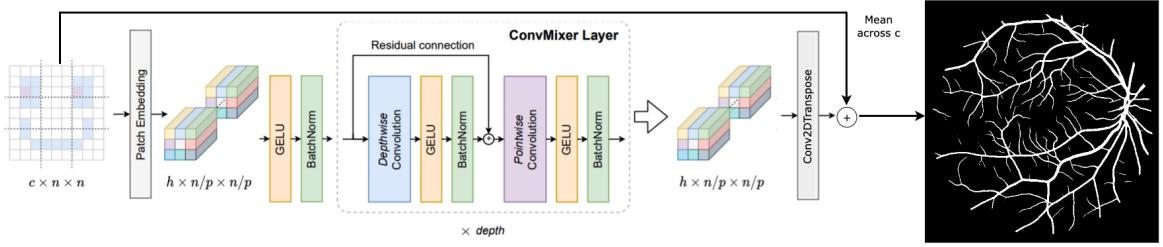
**Figura 21:** Curva de entrenamiento de la UNETR.

Habiendo explorado los *transformers* como alternativa para la segmentación de imágenes médicas, y, concretamente en el problema de DRIVE al no haber sido posible superar a la U-Net, se plantea el uso de una última arquitectura.

#### 2.4.3. ConvMixer

La estructura de la ConvMixer introducida anteriormente 1.2.3 no es completamente funcional de manera directa para el problema de DRIVE sobre el cual se ponen a prueba los modelos. Debido a que el problema tratado es de segmentación y no de clasificación, se han tenido que modificar los últimos módulos de la arquitectura (los bloques posteriores a las capas ConvMixer en la Figura 6) para poder obtener la máscara final con la imagen segmentada. Para ello, simplemente se ha añadido una última capa con una convolución traspuesta con la que se pasa de un número de canales  $h$  a un único canal, empleando un tamaño de kernel y *stride* igual al tamaño de los *patches* en los que se divide la imagen. Al resultado de esta operación, que es una máscara de dimensiones idénticas a la imagen original, se le añade una conexión residual mediante una operación de suma píxel a píxel con la imagen de entrada original promediado a través de los 3 canales. Finalmente se pasa una función sigmoide sobre el resultado para lograr la máscara real de la imagen con valores de 0 o 1. Estas modificaciones se ilustran en la Figura 22.

Con este *set\_up*, se han probado diferentes configuraciones en base a los parámetros



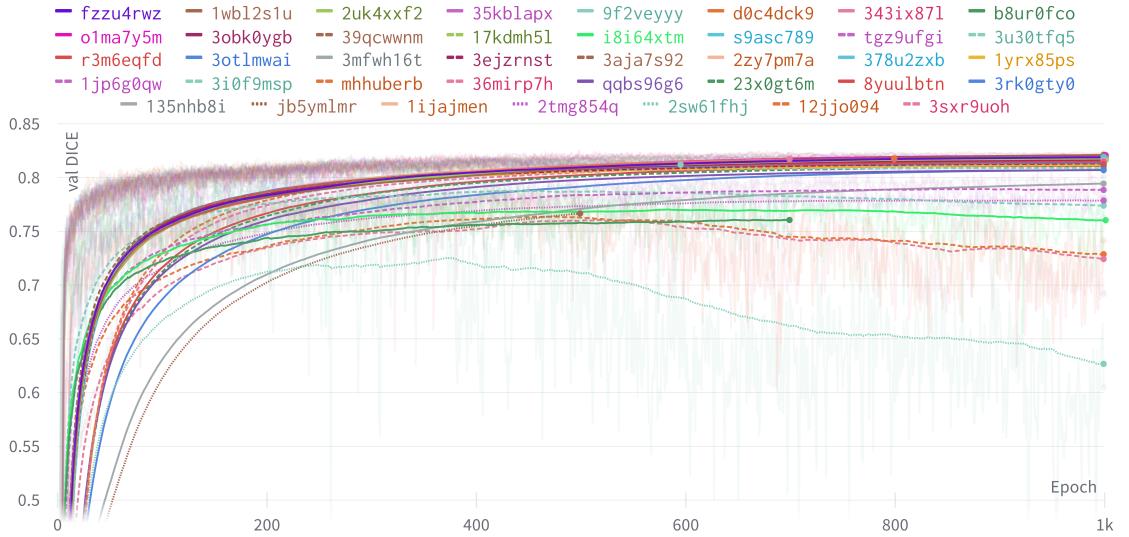
**Figura 22:** Esquema de la adaptación realizada a la arquitectura original de la ConvMixer [6].

modificables: tamaño de *patch*, dimensión del *embedding*  $h$ , tamaño del kernel  $k$  y número de módulos ConvMixer  $depth$ . Por otro lado, también se han modificado los parámetros del data augmentation pero sin añadir nuevas transformaciones, con la finalidad atribuir las posibles mejoras en las métricas a la propia arquitectura y no a haber aumentado la riqueza de la muestra de entrenamiento.

Para el entrenamiento de los modelos y la llegada al mejor de ellos se ha seguido una estrategia “*bottom up*”, empezando por una arquitectura simple, tratando de ajustar hiperparámetros y logrando consecutivamente mejores métricas antes de probar a incrementar el tamaño de la red en cuanto a número total de parámetros entrenables (ver Figura 23), ya que el hecho de incrementar el tamaño de la red no implica automáticamente lograr mejor resultado. Este proceder es especialmente recomendable teniendo en cuenta el coste asociado al entrenamiento de un modelo de gran número de hiperparámetros. En el artículo original de la ConvMixer [47], los autores evalúan dos modelos a los que llaman “ConvMixer-768/32” y “ConvMixer-1536/20” respectivamente, siendo estos números los hiperparámetros  $h/depth$ . El modelo más ligero de éstos, para nuestro problema (suponiendo un tamaño de imagen de (576,576) y *patches* de (192, 192) con tamaño de kernel  $k = 3$ ) presenta un número ligeramente superior a 19 millones de parámetros, lo cual es comparable a los parámetros que posee la U-Net ( $\sim 30$  millones). Sin embargo, estos tamaños no son nada prácticos a la hora de tunear hiperparámetros, especialmente sin un conocimiento a priori sobre posibles rangos en los hiperparámetros en los que pueda encontrarse la combinación que mejor funcione, y más siendo esta arquitectura tan novedosa, habiendo sido aplicada a la tarea de clasificación de imágenes genéricas, que dista del problema de segmentación para estas imágenes tan particulares.

Por este motivo se partió de una arquitectura con un número reducido de hiperparámetros y, poco a poco, se fue incrementando el tamaño de la red hasta ser capaces de superar los resultados obtenidos con la U-Net [5] de esta forma, las ejecuciones son rápidas y, por ende, la tarea de exploración del efecto asociado al cambio de cada uno de los hiperparámetros de la red.

Una herramienta especialmente útil para la optimización de los hiperparámetros y el seguimiento en general de los modelos es el software de Weights & Biases [71]. Esta herramienta permite llevar un control completo sobre el entrenamiento de cada modelo, almacenando sus hiperparámetros y métricas, e integrando todo ello en diversos *dashboards* interactivos especialmente útiles y customizables. También integra otras funcionalidades como alertas ante eventos o elaboración de reportes automáticos o



**Figura 23:** Evolución de las variantes de los modelos entrenados de la ConvMixer, recogida en uno de los *dashboards* de Weights & Biases.

semi-automáticos. El modelo ha sido entrenado empleando la versión PRO de Google Colab, que permite el acceso a GPUs más potentes de manera más frecuente. Como en las máquinas virtuales de Google Colab se puede instalar la librería python de Weights & Biases, desde un único cuaderno en Colab se puede tanto entrenar los modelos, como utilizar Google Drive para almacenar los pesos de los mismos, todo ello mientras se registra el seguimiento del entrenamiento de cada modelo.

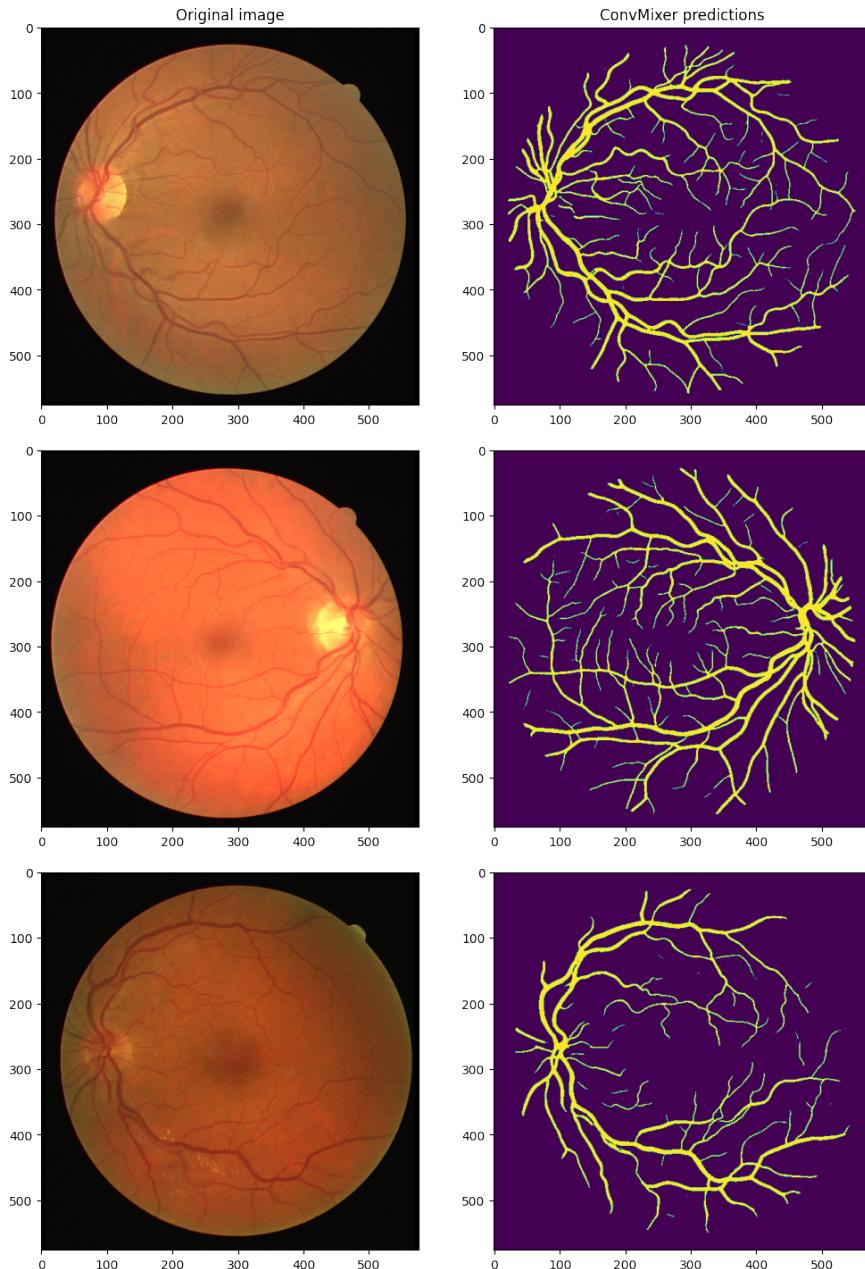
Los resultados obtenidos con el mejor modelo se muestra en la Tabla 7, junto a las predicciones generadas por el mismo en la Figura 24.

Model	Val DICE	Test DICE	Leaderboard	# Params
ConvMixer best	0.820	0.825	202	2.970.417

**Tabla 7:** Resultados obtenidos con el mejor modelo de la ConvMixer. En concreto, para esta arquitectura se ha fijado:  $\text{patch\_size} = 3$ ,  $h = 375$ ,  $depth = 20$  y  $k = 3$ .

A través del mejor modelo de la ConvMixer se ha logrado superar los resultados conseguidos con la U-Net [5]. La mejora en el DICE, pese a parecer sutil, atendiendo al *leaderboard* son multitud de puestos que se han conseguido escalar en el *ranking*, por lo que la mejora es considerable. Otro punto destacable es la reducción de parámetros en este modelo, 2.970.417 millones en la ConvMixer frente a los 31 millones entrenados con la U-Net.

A la hora de entrenar los modelos se ha visto que, en el caso particular de las imágenes retinales, por lo general es preferible una división inicial en *patches* relativamente grandes, de tamaño (192,192) para un  $\text{patch\_size}=3$ , al contrario que con el tamaño del kernel  $k$  de la convolución *depthwise*, que cuando es más pequeño se alcanzan mejores resultados (contrariamente en ambos casos a lo que sugieren los autores). Esto mismo también se había notado en el entrenamiento de la UNETR [21] y se podría atribuir a las peculiaridades de este problema. Al ser el tamaño de las venas en las imágenes bastante reducido puede parecer razonable el uso de kernels pequeños que permitan



**Figura 24:** Predicciones obtenidas con el mejor modelo de la ConvMixer sobre las 3 primeras imágenes del conjunto de test.

recoger el detalle de las mismas y, pese a introducir probablemente un *bias* de localidad de “corto alcance” y pérdida de contexto, para este problema en particular el sacrificio no parece ser tan relevante como la propia información contenida en los canales de las proximidades del píxel a predecir. Esto puede tener sentido al ser una cuestión simplemente de escala y explicaría, en parte, los pobres resultados obtenidos con los ViTs en conjunto con el resto de factores ya comentados.

Recuperando la línea de la reducción del número de parámetros entrenables, con el modelo de la ConvMixer se logra reducir en un factor 10 el número de parámetros entrenables, lo cual es un aspecto sumamente positivo teniendo en cuenta el coste de entrenamiento de los modelos más pesados. De hecho, hasta la llegada de la arquitectura

propia del modelo mencionado, en el diseño de las numerosas arquitecturas intermedias se han explorado otras redes con incluso menos parámetros que parecían tener potencial como para superar a la U-Net. El caso más relevante es el haber logrado entrenar un modelo al que bautizamos como *ConvMixer light*, el cual con menos de 300k parámetros entrenables es capaz de lograr mejores resultados que la propia U-Net, esto es una reducción de un factor 100 con respecto al tamaño de la red. Los resultados de este modelo *light* se pueden observar en la Tabla 8.

Model	Val DICE	Test DICE	Leaderboard	# Params
ConvMixer light	0.822	0.822	356	268.702

**Tabla 8:** Resultados obtenidos con el modelo *light* de la ConvMixer, que cuenta con un número reducido de parámetros entrenables. En concreto, se ha fijado: `patch_size = 3, h = 128, depth = 14` y `k = 3`.

Model	Val DICE	Test DICE	Leaderboard	# Params (M)
UNET	0.812	0.820	406	31
Swin-Unet	0.756	0.771	1737	27
UNETR	0.798	0.805	990	104
ConvMixer	0.820	0.825	202	3
ConvMixer-light	0.822	0.822	356	0.3

**Tabla 9:** Comparativa de los resultados finales de todos los modelos de deep-learning explorados a lo largo de este trabajo. Las métricas sobre el *leaderboard* pueden haber cambiado ligeramente a la hora de consultar este trabajo debido a nuevas *submissions* por parte de otros participantes.

### 3. Conclusiones

Se ha realizado un recorrido generacional por las diferentes técnicas de segmentación de imágenes en el contexto médico. Concretamente, se han testeado diferentes algoritmos sobre el conjunto de datos de DRIVE (Digital Retinal Images for Vessel Extraction) con la finalidad de examinar las principales ventajas y desventajas de los modelos a través de unas métricas que respalden estas conclusiones.

En primer lugar se han probado algunas de las técnicas consideradas clásicas, en el sentido de ser algoritmos más antiguos que históricamente eran populares antes de la llegada de deep learning al campo médico. En esta categoría se pueden incluir los resultados obtenidos con el método de thresholding e incluso los algoritmos de clustering. Hablando primeramente sobre el thresholding, los resultados son pobres en el conjunto de datos sobre los que se comparan los resultados a lo largo de este trabajo. Tal y como se muestra en la Figura 8, aunque se diera con un umbral óptimo que permitiera separar por completo los vasos sanguíneos de la retina, los parámetros que definirían esta segmentación serían altamente sensibles a cambios en la intensidad de cualquiera de los canales de la imagen, por lo que la mínima alteración en parámetros como el brillo estropearía la automatización de la segmentación, y por ende, se tendría que recalibrar el threshold adecuado para cada imagen. Además se puede observar que en este problema las imágenes oculares presentan variaciones locales por ejemplo en el brillo, habiendo zonas más iluminadas que otras, por lo que pese a que un cierto threshold pudiera funcionar bien en una región del ojo, los vasos sanguíneos de extremo opuesto pueden presentar unos valores en la intensidad de los canales R,G y B que sean considerablemente diferentes.

El clustering por su parte también presenta varios inconvenientes, el más general y que comparten todos los algoritmos es el hecho de no estar diseñados para esta tarea en particular. Por un lado el algoritmo K-means presenta como principal ventaja el poder especificar el número de clusters formados, además de ser rápido y sencillo. Sin embargo esto último para el dataset en cuestión es un problema ya que implícitamente el K-means tiende a formar clusters geométricamente sencillos (formas esféricas), lo cual es incompatible con la estructura de los vasos sanguíneos, que además, junto al resto de la imágenes presentan un ruido al cual este algoritmo es a su vez sensible. Por otro lado, algoritmos más complejos como el DBSCAN, preparado para lidiar con ruido en los datos, no permite fijar el número de clusters producto del algoritmo lo cual no permite hacer el paso de un problema de clustering a uno de segmentación. Mediante un ajuste de los hiperparámetros es posible lograr una configuración en la que se forme el número de clusters deseado, en este caso 2, e incluso se puede lograr una máscara (Figuras 10a y 10a) sobre la cual obtener métricas y compararlas con las de otros modelos. El principal problema, como se adelantaba anteriormente, es que estos hiperparámetros no garantizan la formación del número de clusters deseado para toda imagen del conjunto de datos, por lo que en una cierta imagen pueden llegar a formarse un número mayor o menor de clusters (Figura 10e), imposibilitando el uso de este algoritmo para la tarea de segmentación.

Lo métodos de thresholding y clustering ocupan un lugar relevante dentro de la segmentación de imágenes en el contexto médico. Sin embargo, este dominio lo conforman

problemas que no requieren una complejidad técnica demasiado elevada en cuanto a la parte algorítmica. Por ejemplo la detección de lesiones cutáneas que presentan patrones similares y un alto contraste con el tono de la piel [72], o en general la segmentación de imágenes médicas poco ruidosas [73], cuyos elementos a segmentar están bien aislados entre sí y son claramente separables de las regiones de no interés. Pese a que estos algoritmos, en parte por su simplicidad, sean buenos candidatos para esta clase de escenarios, en una gran parte de los problemas de segmentación no se van a poder dar las condiciones apropiadas para lograr unos resultados aceptables con estas aproximaciones, sin ir más lejos en los resultados obtenidos para el problema abordado (Figura 9).

El siguiente paso que se ha dado en este trabajo para mejorar los resultados ha sido explorar el terreno de los algoritmos de machine learning, en línea con la propia evolución histórica de las técnicas de segmentación de imágenes en el ámbito médico. Concretamente, dentro de la familia de los algoritmos tabulares, apropiados para problemas en los que los datos se pueden expresar en forma de tabla, pese a que es posible encontrar multitud de alternativas, algunas de ellas se han consolidado como las más comunes a la hora de abordar un problema de estas características. Algoritmos en general de bagging o de boosting suelen ser las opciones más populares dentro de la comunidad de *data science*, por lo que se ha escogido el Random Forest como representante en esta sección.

Pese a que los resultados [11] obtenidos implican una mejora con respecto a los métodos clásicos, la realidad es que con este algoritmo el resultado queda lejos de representar una máscara limpia con la que un profesional médico pueda diagnosticar cualquier afección. Aún habiendo podido explorar otros modelos (por ejemplo XGBoost o similares) con el objetivo de alcanzar mejores resultados, viendo las salidas del modelo se entiende que, partiendo únicamente de la información contenida en los canales de la imagen a modo de variables explicativas para cualquiera de estos modelos, jamás se va a poder superar a los modelos de CNNs diseñados para tratar con imágenes, donde la dependencia local entre los píxeles es captada a través de los kernels. Este es el motivo por el que se abandonan este tipo de alternativas y, siguiendo la senda hasta resolver el problema de la segmentación retinal, se opta por entrar en el paradigma del deep learning.

Dentro del ámbito de la visión por ordenador mediante deep learning, las redes convolucionales juegan un papel fundamental desde que demostraron su potencial en el problema de clasificación de imágenes con ImageNet. En el campo de la segmentación, y más concretamente en el contexto médico, la U-Net es una de las arquitecturas más conocidas y empleadas, ya que suele lograr una buena segmentación con pocos datos de entrenamiento si se le complementa con las técnicas de data augmentation adecuadas. Se ha entrenado esta arquitectura en el problema de DRIVE y se ha comprobado a través de los resultados obtenidos (Tabla 3 y Figura 14) que esta arquitectura es completamente funcional. Por primera vez se es capaz de devolver una máscara que claramente separa los vasos sanguíneos del fondo de globo ocular, por lo que se puede considerar que a través de esta arquitectura el problema de la segmentación retinal quedaría resuelto.

A partir de este punto, el enfoque seguido no se ha centrado en encontrar otras alternativas que puedan ser comparables a la U-Net, sino en buscar nuevas y más recientes

arquitecturas (siguiendo con el avance generacional) que demuestren ser capaces de superar a ésta última. Las métricas a la hora de comparar modelos pasan a ser el factor más importante, por lo que el formato estilo concurso del *challenge* asociado al *dataset* que se emplea, es de gran utilidad para disponer de un conjunto de imágenes *test* objetivo y no manipulado. Siguiendo esta misma línea se ha exprimido al máximo la arquitectura de la U-Net, diseñando nuevas transformaciones como parte del *data augmentation*, para lograr una métrica que actúe como cota superior alcanzable mediante esta red. Los resultados obtenidos en la Tabla 5 ponen de manifiesto el buen funcionamiento de esta arquitectura al complementarse con las transformaciones adecuadas en el proceso de entrenamiento. Habiendo establecido este *baseline* con la arquitectura estándar de la U-Net, las consiguientes mejoras en la métrica de referencia se plantean desde el diseño y retoque de arquitecturas más modernas.

Ante el éxito de los *transformers* en el mundo del NLP, su traslado al campo de la visión por ordenador recientemente, pese a no estar igual de consolidado, es representado por varias publicaciones con diferentes arquitecturas que resultan prometedoras. La introducción de los mecanismos de atención en los ViTs los sitúan en el estado del arte de la segmentación semántica en el último año, por lo que a priori parecen buenos candidatos para lograr batir a la U-Net en el problema elegido. Se han evaluado por tanto dos arquitecturas diferentes que emplean estos mecanismos, los resultados en cambio han estado lejos de tan siquiera acercarse al lugar que ocupa la U-Net.

Con los ViTs el principal problema con el que se ha tenido que lidiar es la escasez de imágenes de entrenamiento. Las arquitecturas que emplean *transformers* requieren de una gran cantidad de datos para optimizar correctamente los mecanismos de atención de manera que comiencen a ser efectivos. Este fenómeno se ha podido comprobar en el entrenamiento de las dos arquitecturas, especialmente en la Swin-Unet [16] al excluir por completo las convoluciones en su arquitectura. El bajo índice de funcionamiento (Tabla 6) de estos modelos con respecto a la estándar U-Net en el problema se ha atribuido a este factor principalmente. Para reforzar este argumento, se ha demostrado (Figura 20) que el efecto del preentrenamiento de este tipo de arquitecturas es un “must”, especialmente en aquellas que basan su funcionamiento exclusivamente en *transformers*, tal y como apuntan los autores de las mismas. Por otro lado, estas redes por lo general cuentan con un elevado número de parámetros entrenables, lo cual es poco conveniente a la hora de explorar el desempeño de una arquitectura sobre unos nuevos datos, pero sobre todo el factor más limitante es la complejidad a la hora de establecer los hiperparámetros propios de las arquitecturas. Al ser tan recientes estas arquitecturas, no se han encontrado unos “guidelines” generales que orienten la optimización de hiperparámetros como el número de *patches* en los que dividir la imagen o el número de módulos de atención que fijar entre muchos otros. Aunque los autores en los *papers* hagan comparativas variando algunos de ellos, en el entrenamiento de los modelos se ha comprobado que, patrones como la mejora de los resultados al incrementar sistemáticamente un cierto hiperparámetro, no son trasladables en este caso al problema de DRIVE, ya que tanto las dimensiones de la imagen como su contenido es diferente, por lo que esta tarea se convierte en algo especialmente difícil, más aún le sumamos el coste de entrenamiento asociado al tamaño de las mismas tal y como se ha destacado antes. A la vista de los resultados obtenidos con la alta especificidad y complejidad de estas arquitecturas, se ha optado por cambiar de dirección y explorar modelos que puedan ser más sencillos, sin sacrificar la condición de estado del arte,

para tener así alguna oportunidad de superar a la U-Net.

ConvMixer se presenta como la alternativa que justamente parece cumplir estas dos condiciones. Con una arquitectura aparentemente simple, los autores reportan superar a los ViTs y a los modelos clásicos de convoluciones cuando se comparan las variantes de los mismos con un tamaño similar. Se ha logrado no solo adaptar la arquitectura para que funcione en los problemas de segmentación sino que, a través de la misma, se han obtenido finalmente unas métricas sobre las predicciones que superan claramente a las de la U-Net.

El modelo con mejores métricas [7] consigue batir a la U-Net por un margen considerable, aunque aún más interesante es el hecho de que lo haga con tan solo 3 millones de parámetros entrenables, una reducción de un factor 10. Este sorprendente resultado pone aún más por encima a la ConvMixer frente al resto de modelos ya que demuestra el potencial de esta red para aplicaciones que requieran de modelos más ligeros. No es un secreto que el TinyML [74] está ganando cada vez más popularidad, impulsado por aplicaciones industriales como por ejemplo el IoT (Internet of Things), presente en multitud de sistemas y expandiéndose cada día más con la mejora de los sistemas de telecomunicaciones (5G). El campo de la medicina por su parte también está viviendo este fenómeno, con la llegada de nuevas aplicaciones basadas en dispositivos equipables que recolectan datos de los pacientes [75].

Habiendo cumplido con el propósito de superar el *baseline* marcado por la U-Net, se ha pretendido en último lugar el profundizar en la capacidad de esta red de obtener un rendimiento superior con relativamente poco parámetros. Durante el proceso de entrenamiento hasta lograr el modelo [23], se han entrenado varias arquitecturas que con un número aún menor de hiperparámetros parecían alcanzar unos resultados comparables a la U-Net. Como representante de esto último se ha presentado el modelo llamado *ConvMixer-light*, que tan solo con algo menos de 270k parámetros entrenables es también capaz de superar a la U-Net, esta vez se consigue una reducción de un factor 100 en este número.

Sin duda, estos resultados son la constatación del potencial que tiene la ConvMixer, en primer lugar para aplicaciones en el campo de la segmentación semántica en general, en segundo lugar para el contexto médico en particular, y por último, en el contexto de las aplicaciones que requieren de TinyML en cualquiera de los dos casos anteriores. Con los resultados obtenidos se da pie a la explotación de esta arquitectura en nuevos problemas, la exploración de las configuraciones óptimas para los mismos e incluso el diseño de nuevas arquitecturas que tengan en cuenta las bondades de este modelo para lograr aún mejores resultados.

El recorrido generacional por la segmentación semántica en el contexto médico desemboca en la demostración de la superioridad de la ConvMixer en el problema escogido. Nuevos estudios sobre otros conjuntos de datos deberían ser complementarios a éste para poder desbancar a la U-Net como el recurso principal en el contexto médico ante los problemas de segmentación. La posibilidad de que esto suceda queda manifestada en este trabajo.

## Bibliografía

- [1] Niall O' Mahony, Sean Campbell, Anderson Carvalho, Suman Harapanahalli, Gustavo Adolfo Velasco-Hernández, Lenka Krpalkova, Daniel Riordan, and Joseph Walsh. Deep learning vs. traditional computer vision. *CoRR*, abs/1910.13796, 2019.
- [2] Rafael C. Gonzalez and Richard E. Woods. Digital image processing (3rd edition). chapter 10,11,12. Prentice-Hall, Inc., USA, 2006.
- [3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012.
- [4] Song Yuheng and Yan Hao. Image segmentation algorithms overview. *arXiv*, abs/1707.02051, 2017.
- [5] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.
- [6] Joe McGlinchy, Brian Johnson, Brian Muller, Maxwell Joseph, and Jeremy Diaz. Application of unet fully convolutional neural network to impervious surface segmentation in urban environment from high resolution satellite imagery. In *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 3915–3918, 2019.
- [7] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L Baker. Design of an image edge detection filter using the sobel operator. *IEEE Journal of solid-state circuits*, 23(2):358–367, 1988.
- [8] Jan Kristanto Wibisono and Hsueh-Ming Hang. Traditional method inspired deep neural network for edge detection. *2020 IEEE International Conference on Image Processing (ICIP)*, pages 678–682, 2020.
- [9] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-8(6):679–698, 1986.
- [10] Jaskirat Kaur, Sunil Agrawal, and Vig Renu. A comparative analysis of thresholding and edge detection segmentation techniques. *International Journal of Computer Applications*, 39:29–34, 02 2012.
- [11] Vamsi Krishna Madasu and Prasad Yarlagadda. An in depth comparison of four texture segmentation methods. In *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications (DICTA 2007)*, pages 366–372, 2007.

- [12] Prasantha S, Dr.Shashidhara.H.L, Kannamedi Balasubramanya Murthy, and Latha Madhavi. Medical image segmentation. *International Journal on Computer Science and Engineering*, 2, 07 2010.
- [13] Wei Yu, Jason Fritts, and Fangting Sun. A hierarchical image segmentation algorithm. volume 2, pages 221 – 224 vol.2, 02 2002.
- [14] Nameirakpam Dhanachandra, Khumanthem Manglem, and Yambem Jina Chanu. Image segmentation using k -means clustering algorithm and subtractive clustering algorithm. *Procedia Computer Science*, 54:764–771, 2015. Eleventh International Conference on Communication Networks, ICCN 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Data Mining and Warehousing, ICDMW 2015, August 21-23, 2015, Bangalore, India Eleventh International Conference on Image and Signal Processing, ICISP 2015, August 21-23, 2015, Bangalore, India.
- [15] Liying Zheng, Jingtao Zhang, and Qianyu Wang. Mean-shift-based color segmentation of images containing green vegetation. *Computers and Electronics in Agriculture*, 65(1):93–98, 2009.
- [16] Huikai Wu, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu. Fastfcn: Rethinking dilated convolution in the backbone for semantic segmentation, 2019.
- [17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [18] Jieneng Chen, Yongyi Lu, Qihang Yu, Xiangde Luo, Ehsan Adeli, Yan Wang, Le Lu, Alan L. Yuille, and Yuyin Zhou. Transunet: Transformers make strong encoders for medical image segmentation. *CoRR*, abs/2102.04306, 2021.
- [19] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.
- [20] Wen-Xiong Kang, Qing-Qiang Yang, and Run-Peng Liang. The comparative research on image segmentation algorithms. In *2009 First International Workshop on Education Technology and Computer Science*, volume 2, pages 703–707. IEEE, 2009.
- [21] Ronald Kline. Cybernetics, automata studies, and the dartmouth conference on artificial intelligence. *IEEE Annals of the History of Computing*, 33:5–16, 04 2011.
- [22] L Ralph Baker, James J Burke, and B Roy Frieden. Progress in digital image processing, 1969. Technical report, Optical Sciences Center, University of Arizona (Tucson, Arizona), 1970.

- [23] P Vasuda and S Satheesh. Improved fuzzy c-means algorithm for mr brain image segmentation. *International Journal on Computer Science and Engineering*, 2(5):2010, 1713.
- [24] Frederic P. Miller, Agnes F. Vandome, and John McBrewster. *AI Winter*. Alpha Press, 2009. Book. ISBN: 6130079133.
- [25] S. Papert. *The Summer Vision Project*. Number 100 in AI memo. 1966. <https://books.google.es/books?id=qOh7NwAACAAJ>.
- [26] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.
- [27] KB Shaw and MC Lohrenz. A survey of digital image segmentation algorithms. Technical report, NAVAL OCEANOGRAPHIC AND ATMOSPHERIC RESEARCH LAB STENNIS SPACE CENTER MS, 1995.
- [28] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.
- [29] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [30] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6:107–116, 04 1998.
- [31] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. volume 37, pages 29–43, 12 2003.
- [32] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.
- [33] Wen-Xiong Kang, Qing-Qiang Yang, and Run-Peng Liang. The comparative research on image segmentation algorithms. In *2009 First International Workshop on Education Technology and Computer Science*, volume 2, pages 703–707, 2009.
- [34] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018.
- [35] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440, Los Alamitos, CA, USA, jun 2015. IEEE Computer Society.
- [36] Shervin Minaee, Yuri Boykov, Fatih Porikli, Antonio Plaza, Nasser Kehtarnavaz, and Demetri Terzopoulos. Image segmentation using deep learning: A survey, 2020.

- [37] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation, 2015.
- [38] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation, 2016.
- [39] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition, 2020.
- [40] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
- [41] Jay Alammar. The illustrated transformer. *The Illustrated Transformer—Jay Alammar—Visualizing Machine Learning One Concept at a Time*, 27, 2018.
- [42] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. Huggingface’s transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771, 2019.
- [43] Ozan Oktay, Jo Schlemper, Loïc Le Folgoc, Matthew C. H. Lee, Matthias P. Heinrich, Kazunari Misawa, Kensaku Mori, Steven G. McDonagh, Nils Y. Hammerla, Bernhard Kainz, Ben Glocker, and Daniel Rueckert. Attention u-net: Learning where to look for the pancreas. *CoRR*, abs/1804.03999, 2018.
- [44] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *CoRR*, abs/1906.05909, 2019.
- [45] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.
- [46] Ilya Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, Mario Lucic, and Alexey Dosovitskiy. Mlp-mixer: An all-mlp architecture for vision, 2021.
- [47] Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv*, abs/2201.09792, 2022.
- [48] Francois Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [49] J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken.

- Ridge-based vessel segmentation in color images of the retina. *IEEE Transactions on Medical Imaging*, 23(4):501–509, 2004.
- [50] Radboud University Medical Center. DRIVE: Digital Retinal Images for Vessel Extraction - Grand Challenge.  
<https://drive.grand-challenge.org/>. Website of the Grand Challenge.
- [51] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. KDD’96, page 226–231. AAAI Press, 1996.
- [52] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
- [53] D.H. Wolpert and W.G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
- [54] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.
- [55] Benchmark object detection. <https://paperswithcode.com/task/object-detection#benchmarks>. Website accessed: 2021-12-25.
- [56] Joe McGlinchy, Brian Johnson, Brian Muller, Maxwell Joseph, and Jeremy Diaz. Application of unet fully convolutional neural network to impervious surface segmentation in urban environment from high resolution satellite imagery. In *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pages 3915–3918. IEEE, 2019.
- [57] Xiaodan Hu, Mohamed A Naiel, Alexander Wong, Mark Lamm, and Paul Fieguth. Runet: A robust unet architecture for image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019.
- [58] Michael Njeru, Ciira Maina, and Kibet Langat. Mammalian species detection using a cascade of unet and squeezeNet. In *2021 IEEE AFRICON*, pages 1–5. IEEE, 2021.
- [59] Binary Cross Entropy asignando pesos en las clases. <https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html>. Website accessed: 2022-01-15.
- [60] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [61] Enes Uysal, M. Bilici, B. Zaza, M. Özgenç, and Onur Boyar. Exploring the limits of data augmentation for retinal vessel segmentation. *arXiv preprint arXiv:2105.09365*, 05 2021.
- [62] Benchmark de la competición de DRIVE. <https://paperswithcode.com/sota/retinal-vessel-segmentation-on-drive?metric=F1%20score>.

- [63] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022.
- [64] Hu Cao, Yueyue Wang, Joy Chen, Dongsheng Jiang, Xiaopeng Zhang, Qi Tian, and Manning Wang. Swin-unet: Unet-like pure transformer for medical image segmentation, 2021.
- [65] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *CoRR*, abs/2103.14030, 2021.
- [66] Ali Hatamizadeh, Yucheng Tang, Vishwesh Nath, Dong Yang, Andriy Myronenko, Bennett Landman, Holger Roth, and Daguang Xu. Unetr: Transformers for 3d medical image segmentation, 2021.
- [67] Stéphane d’Ascoli, Hugo Touvron, Matthew L. Leavitt, Ari S. Morcos, Giulio Biroli, and Levent Sagun. Convit: Improving vision transformers with soft convolutional inductive biases. *CoRR*, abs/2103.10697, 2021.
- [68] Katelyn Morrison, Benjamin Gilby, Colton Lipchak, Adam Mattioli, and Adriana Kovashka. Exploring corruption robustness: Inductive biases in vision transformers and mlp-mixers. *CoRR*, abs/2106.13122, 2021.
- [69] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification, 03 2021.
- [70] Michela Antonelli, Annika Reinke, Spyridon Bakas, Keyvan Farahani, AnnetteKopp-Schneider, Bennett A. Landman, Geert Litjens, Bjoern Menze, Olaf Ronneberger, Ronald M. Summers, Bram van Ginneken, Michel Bilello, and et al. Bilic. The medical segmentation decathlon, 2021.
- [71] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.
- [72] Ashmita Gupta, Ashish Issac, Malay Kishore Dutta, and Hui-Huang Hsu. Adaptive thresholding for skin lesion segmentation using statistical parameters. *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pages 616–620, 2017.
- [73] Amlan Dutta, Abhijit Pal, Mriganka Bhadra, Md Akram Khan, and Rupak Chakraborty. An improved k-means algorithm for effective medical image segmentation. In Jyotsna Kumar Mandal, Somnath Mukhopadhyay, Aynur Unal, and Santanu Kumar Sen, editors, *Proceedings of International Conference on Innovations in Software Architecture and Computational Systems*, pages 169–182, Singapore, 2021. Springer Singapore.
- [74] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al. Benchmarking tinyml systems: Challenges and direction. *arXiv preprint arXiv:2204.07118*, 2022.

*arXiv:2003.04821*, 2020.

- [75] Vasileios Tsoukas, Eleni Boumpa, Georgios Giannakas, and Athanasios Kakarountas. A review of machine learning and tinyml in healthcare. In *25th Pan-Hellenic Conference on Informatics*, PCI 2021, page 69–73, New York, NY, USA, 2021. Association for Computing Machinery.