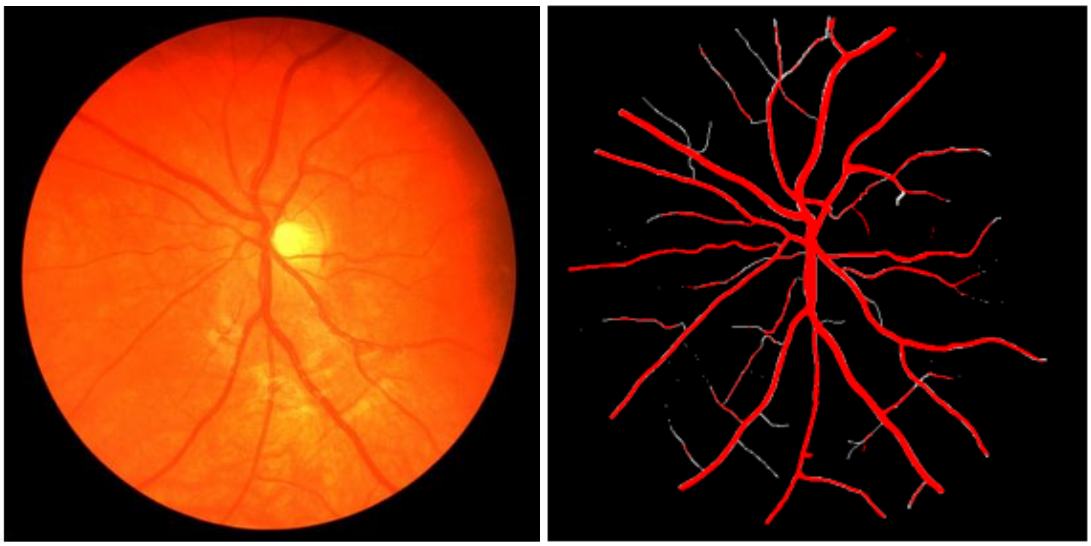
Trabajo Dirigido
Curso 2024/25

SEGMENTACIÓN DE VASOS RETINIANOS MEDIANTE U-NET POTENCIADA POR VISION GNN, IMÁGENES DE PERSISTENCIA Y FUNCIÓN DE COSTE TOPOLÓGICA



Javier Serrano Jodral
Estudiante del Máster Universitario en Ingeniería Biomédica y Salud Digital de la Universidad de Sevilla
Email de contacto: javierserranojodral@gmail.com

RESUMEN

Con el objetivo de mejorar la calidad de las segmentaciones de los vasos sanguíneos en la retina en imágenes 2D, se ejecuta un diseño de red neuronal profunda basado en una U-Net clásica, de estructura *autoencoder*, a la que se le acopla un módulo auxiliar extractor de características topológicas en forma de red neural de grafo, Vision GNN, típicamente usado en tareas de clasificación. También se propone la inclusión de imágenes de persistencia como información adicional, así como el empleo de una función de pérdida topológica que considere las componentes conexas, fomentando una conectividad coherente. El código elaborado está disponible en el repositorio https://github.com/javserjod/unet_vig_segmentation.

Palabras clave: Segmentación · Imágenes médicas · U-Net · Vision GNN · Topología

ABSTRACT

In order to improve the quality of blood vessel segmentation in 2D retinal images, we propose a deep neural network design based on a classical U-Net with an autoencoder structure, incorporating an auxiliary module for extracting topological features in the form of a Vision GNN, a graph neural network typically used for classification tasks. Additionally, we propose the inclusion of persistence images as extra information, along with the use of a topological loss function that considers connected components, promoting coherent connectivity. The developed code is available at https://github.com/javserjod/unet_vig_segmentation.

Keywords: Segmentation · Medical Imaging · U-Net · Vision GNN · Topology

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El estudio de los vasos sanguíneos retinianos es vital para el correcto y temprano diagnóstico de afecciones asociadas al órgano u otras patologías causantes de síntomas en este (cáncer, diabetes, migrañas, enfermedades cardíacas o neurológicas…) [4]. Para analizarlos en detalle, es imprescindible separarlos previamente del resto de componentes del globo ocular captados en la retinografía, angiografía o cualquier otra modalidad de imagen médica, en lo que se conoce como segmentación. Se trata de un proceso históricamente realizado manualmente por especialistas en el ámbito y que, dependiendo de la situación, puede suponer un enorme desafío. Esto se traduce en una alta demanda temporal, un problema que, sumado a la escasez de profesionales, podría desembocar en el colapso de ciertos servicios hospitalarios.

La necesidad de alcanzar una perfecta precisión se ve lastrada por la gran complejidad anatómica de los vasos retinianos: estructuras intrincadas y extremadamente finas, con infinidad de formas, tamaños y orientaciones, variaciones inter-pacientes… Además, las propias imágenes introducen barreras adicionales a la tarea: condiciones de iluminación variables a lo largo de la retina, artefactos y ruido, diferentes equipos de adquisición y sus ajustes de ángulos, de filtros, etc. Todo esto hace que una segmentación manual fidedigna sea realmente costosa de obtener, provocando la insuficiencia de estas.

En los últimos años, las potentes soluciones basadas en aprendizaje profundo han experimentado un considerable auge, implantándose en innumerables aspectos de la vida diaria y sectores gracias al incremento exponencial de la capacidad computacional, así como a la

simplificación en el acceso a *frameworks*, bibliotecas especializadas y fuentes de información y de datos. En este contexto, el ámbito sanitario ha sido enormemente beneficiado en todas sus facetas, siendo un claro ejemplo de aplicación el recién comentado. Para dicho caso específico, haciendo uso de una cantidad limitada de los caros datos procesados a mano, es posible entrenar un modelo diseñado a medida y conseguir, automáticamente y al instante, segmentaciones de calidad sin la intervención directa de un especialista, evitando la ardua labor de anotación tradicional y, por lo tanto, ahorrando recursos.

Con el propósito de perfeccionar la técnica de segmentación y trazar el camino del progreso, miles son los artículos científicos e investigaciones publicados recientemente, en especial en el contexto médico, que abordan el problema desde distintas perspectivas. Por ejemplo, en [UBZ21] se realiza un estudio del drástico efecto en el rendimiento de un conjunto de apropiadas transformaciones afines, elásticas y a nivel de píxel a la hora de segmentar los vasos retinianos. En [SCR20] se propone el diseño Image PI-Net para producir automáticamente imágenes de persistencia (PI), evitando el cuello de botella habitual de su cálculo, que más tarde pueden incorporarse a otro modelo de entrada múltiple para favorecer la actividad, o como sugiere [SJ24], con *Vector Stitching*, concatenando la imagen en bruto con su PI en el mismo canal. En cuanto a las arquitecturas, la amplia mayoría son variaciones de la U-Net centradas en su mejora. Por ejemplo, en [LSY23] se refuerzan aquellos aspectos más deficientes de la simétrica estructura codificador-decodificador para la segmentación de los vasos, conformando la ResDO-UNet, poseedora de una extracción de características más sólida, capas de agrupación con reducción de pérdidas y bloques de atención en las conexiones de salto.

Igualmente, en [WWZ21] se aborda de otra manera, incluyendo el ajuste dinámico de los campos receptivos, la fusión adaptativa de características de distinto nivel y un módulo para ponderar la influencia de la salida de cada nivel del decodificador. Con tan solo algunas de estas investigaciones, se evidencian las infinitas posibilidades de optimización a las que es susceptible el campo de la segmentación de imágenes médicas, así como los numerosos caminos que valdría la pena explorar.

En este documento, se proponen los cimientos de un nuevo marco de trabajo sencillo y veloz para tareas de segmentación de imágenes bidimensionales a color adquiridas por retinografía. Está compuesto por dos tipos de redes neuronales: una arquitectura base, la U-Net [RFB15], ya predominante en el estado del arte, junto a un módulo acoplado basado en una Vision GNN [HWG22], un novedoso modelo destinado originalmente a la clasificación que trabaja considerando las imágenes como grafos, alcanzando información global y oculta que se complementa con la espacial local capturada por la primera, ofreciendo en conjunto unos resultados superiores. Las pruebas son llevadas a cabo sobre el dataset CHASE_DB1 [FRH12], escogido por la adecuada resolución de las imágenes y por la inclusión de un predefinido set de testeo con las segmentaciones *ground truth*. Los resultados producidos se comparan, mediante métricas aptas, con otros modelos del estado del arte sobre la misma colección de datos.

En el apartado 2 se hablará del contexto teórico en el que se basa la Vision GNN. En la sección 3, se propone un diseño combinatorio de U-Net, Vision GNN, imágenes de persistencia y funciones topológicas, siendo este diseñado en el apartado 4 en una versión acorde a los recursos disponibles y, posteriormente, evaluado en la sección 5. Se concluye y se redactan líneas futuras de trabajo en la sección 6.

2. MARCO TEÓRICO DEL PROBLEMA

Dado que la novedad introducida en este proyecto nace de la combinación de una ya asentada arquitectura U-Net con otra basada en grafos, es propicio comentar brevemente la fundamentación teórica en la que se justifica el diseño de esta última, a fin de adquirir una comprensión más profunda de los procedimientos del desarrollo. Se siguen las mismas secciones del artículo original y se aportan comentarios adicionales críticos cuando corresponda.

El pionero artículo [HWG22] en torno al cual orbita el diseño del modelo propio toma un enfoque novedoso, sin antecedentes según los autores: tratar las imágenes como grafos para usarlos como entrada de una red neuronal de grafos (GNN) y logrando evitar la pérdida de información al avanzar a lo largo de esta.

La GNN es más flexible que la hegemónica arquitectura de red neuronal convolucional (CNN), ya que supera las limitaciones de trabajar en un espacio euclídeo regular, de cuadrículas fijas (*grid*), donde solo se extrae información local, pues las convoluciones se ejecutan siempre entre píxeles vecinos inmediatos o muy próximos, según las dimensiones del *kernel* deslizante, y, por ende, no se desbloquean características globales ni relaciones entre objetos [LTZ22]. Algo similar sucede en los transformadores de visión (ViT): aunque puedan capturar vínculos entre los *patches* o bloques en los que se dividen las imágenes, no modelan explícitamente la conectividad real ni demás información estructural, pues dependen del mecanismo de *self-attention*. En cambio, las GNN sí adquieren expresamente información holística y contextual más compleja al explotar grafos geométricamente irregulares, moldeándose a la estructura de los objetos captados en la imagen, lo que encaja

perfectamente con anatomías complejas como los vasos retinianos del caso de estudio presente. Así, unos datos se propagan iterativamente entre nodos por medio de las aristas que los vinculan, pudiendo estos pertenecer a regiones distantes en la matriz original.

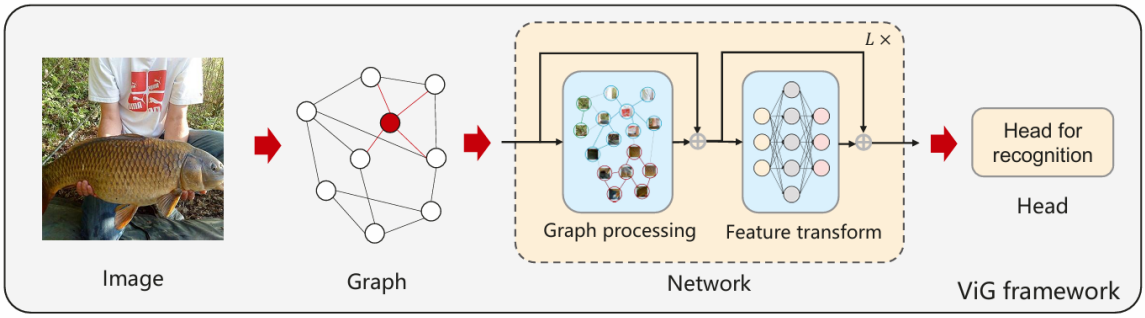


Ilustración 1. Framework original del modelo ViG. Fuente: [HWG22]

2.1. Descripción del enfoque de Vision GNN

A continuación, se comentan, sin entrar en aspectos técnicos detallados, los fragmentos funcionales en los que se divide la Vision GNN, sintetizados gráficamente en la Ilustración 1, quizás demasiado simplificada, poco aclarativa. Nótese que esta red ha sido desarrollada originariamente para tareas de clasificación, por lo que la arquitectura en cuestión simplemente es utilizada como *backbone* o extractor de características, dependiendo completamente de una *head* ajena para la emisión de las decisiones finales. Se adelanta que los autores del *paper* han sido sumamente concisos, dejando de lado algunas explicaciones o aclaraciones importantes. Afortunadamente, se han podido llenar los vacíos gracias a la consulta de las dos otras únicas fuentes de información escritas en la web que trabajan el mismo artículo: un estudio de replicación [Mir24] y una propuesta de U-Net con bloques ViG [JCT23], aún por describir.

2.1.1 Conversión de imagen a grafo

Vision GNN precisa un grafo como entrada. Sin embargo, solo se disponen de imágenes bidimensionales con tres canales. Por lo tanto, urge idear un método para la gestión de este inconveniente. La estrategia resolutiva es bastante lógica: cada uno de los nodos o vértices del grafo se corresponderá con un subconjunto de píxeles (*patch*), no siguiendo una relación 1 a 1 entre nodo y píxel por claras cuestiones de costo computacional. El tamaño de los *patches* varía en función del parámetro elegido por el usuario para el número de *patches* total en imágenes de dimensiones equitativas, *N*, y el tamaño de la imagen de la que se extraen.

Se indica, aunque muy escuetamente, que cada *patch* se convierte en un vector de características, ergo, en un punto o nodo representable en el espacio de características *D*-dimensional, siendo *D* el número de *features* calculadas. La forma en la que se extraen dichas *features* es omitida, por lo que queda a libre disposición del usuario. Para cada nodo, se buscan sus *K* vecinos más cercanos, según un criterio de proximidad o similitud, de nuevo, dejado a elección del usuario, y se trazan las aristas compartidas con ellos. Al hacer esto para cada nodo, se acaba construyendo un grafo que sirve de entrada a la red. Pero este no será el único momento en el que se construya el grafo: su estructura deberá ser recalculada en cada capa del modelo para siempre establecer las vecindades correctas, pues los vectores que definen a los nodos irán variando de valor gracias al paso de mensajes iterativo entre ellos (*message passing*).

2.1.2 Procesamiento a nivel de grafo

Y es que, en una GNN, con el recién mencionado paso de mensajes, lo que se predice o calcula en cada capa es el valor o *feature* de cada nodo en función de las *features* de sus vecinos (*aggregation*) y la suya propia (*update*) en la capa anterior. Las conexiones con los vecinos tienen un peso asociado que actúa como parámetro ajustable durante el entrenamiento de la red, al igual que existe un peso autoasignado al nodo en cuestión. Como en los alternativos tipos de redes neuronales, se precisan de unos sesgos en cada operación, aunque no se mencionan por simplicidad. Como se expresa en el artículo, se suelen aplicar unas funciones sobre el resultado de la operación lineal. Primero, al computar la agregación de los vecinos, en el *paper* se opta por un símil de *max pooling* entre la vecindad de cada nodo, desembocando en el valor máximo relativo de las características. Después, hipotéticamente, debería aplicarse otra función sobre el resultado anterior, simbolizando la actualización con el peso propio, pero parece ser que los autores la omiten, no quedando claro si es un error de redacción o la verdadera decisión de desarrollo. Generalmente, todo este proceso se llama convolución de grafo, y se produce en cada capa de una red convolucional de grafo (GCN).

Adicionalmente, los autores, en pos de lograr más diversidad en las *features*, modifican y sustituyen la convolución de grafo clásica ya comentada e introducen el concepto de *multi-head update*, una especie de mecanismo de atención, donde después de calcular la agregación con los vecinos para un nodo, se divide el vector resultado en *h* partes y a cada una de ellas se le aplica un peso de actualización entrenable diferente, concatenando todos los valores finales en el vector *feature* saliente.

2.1.3 Bloque ViG

Los cálculos anteriores, al sucederse repetidamente en las GCN más profundas, provocan el efecto conocido como *over-smoothing*, causando un decremento en la heterogeneidad de las *features* de los nodos y empeorando el rendimiento del modelo. Para paliar este problema, los autores diseñan el bloque ViG, la unidad básica de la arquitectura Vision GNN (ViG) propuesta. Cada bloque ViG estará compuesto por los módulos Grapher y red neuronal prealimentada (FFN), vitales para un mejor rendimiento, tal y como se prueba al final del documento en un estudio de ablación.

El módulo Grapher es el encargado de procesar el grafo, y no es más que una sucesión de capa lineal, la capa de convolución de grafo ya comentada (*aggregation* y *multi-head update*), función de activación no lineal (ya sea ReLU o GeLU) y otra capa lineal. A su salida se le suma su entrada, simulando una conexión residual, la cual agiliza el entrenamiento al servir como atajos por los que propagar hacia atrás el valor del gradiente. Al ser caminos más cortos, se reduce el problema del desvanecimiento de dicho gradiente, dictando así de manera efectiva la magnitud y dirección del cambio de los parámetros durante el entrenamiento.

Después, para fomentar la transformación de las *features* que salen del Grapher y reducir el *over-smoothing*, se utiliza una FFN de dos capas (o sea, con una capa oculta) en cada nodo, sobre cada vector *feature*. De nuevo, se suma la entrada de la FFN a la salida de esta. Se menciona que también se aplica *Batch Normalization* después de las capas completamente conectadas y de convolución de grafo, agilizando el entrenamiento al normalizar las entradas de las capas siguientes.

2.2. Arquitecturas de Vision GNN

En el artículo se proponen dos modelos de ViG: el isotrópico y el pirámide, con el propósito de compararse adecuadamente con los Transformers y CNNs, que, respectivamente, siguen la misma arquitectura. Para cada uno de ellos se definen varios submodelos según el número de parámetros, desde *tiny* hasta *base*.

El modelo isotrópico mantiene las *features* con el mismo tamaño y forma en toda la red, con h=4 por defecto en *multi-head update*, y a medida que se va adentrando en ella, el número de nodos considerados vecinos se aumenta linealmente de 9 a 18, habiendo en total 196 nodos. Los submodelos varían la profundidad, es decir, el número de bloques ViG, de 12 a 16, y las D features calculadas por nodo, de 192 a 640.

El modelo pirámide considera la propiedad multiescalar de las imágenes, obteniendo características invariantes a la escala, por lo que estas se reducen progresivamente en tamaño espacial a lo largo de la red, pero se van calculando más *features* por nodo. El número de vecinos para la GCN siempre es 9. Esta arquitectura posee más parámetros que la anterior, por lo que es más abusivo en cuanto a potencia computacional.

Adicionalmente, se añade una codificación posicional a la *feature* de cada nodo para dotarla de información sobre su posicionamiento en el espacio D-dimensional. Esta consiste en simplemente sumar un vector, aunque no se especifica el método de obtención. La inclusión de esta información en el artículo original está en una ubicación inadecuada que puede desorientar al lector, ya que está más vinculado al procesamiento a nivel de grafo que a las estructuras. Se

repite el mismo fallo de brevedad: no explica dónde ni cómo se origina el vector de codificación.

2.3. Experimentación con Vision GNN

Acto seguido, se llevan a cabo numerosos experimentos, excelentemente detallados y representados tabular y claramente, empleando distintas arquitecturas de ViG, parámetros y datasets, para problemas de clasificación de imágenes y detección de objetos, donde sorpresivamente demuestran superioridad, o al menos se comparan, tanto los submodelos isotrópicos como los de pirámide, con los modelos más representativos de cada arquitectura en el SOTA para clasificación, como ResNet [HZR15] (CNN), CycleMLP [CXG22] (Multilayer Perceptron, MLP) y Swin-T [LLC21] (Transformer).

Por último, tiene lugar un estudio de ablación de los distintos módulos del diseño, sirviendo únicamente para ratificar que el modelo, tal y como se ha descrito, es la mejor alternativa. Se explica que la elección de pocos vecinos reduce la información que intercambian para calcular las *features* y que un número demasiado elevado la diluye. Luego, se analiza el efecto del número de *heads* en *multi-head update*, siendo una única cabeza la que mejor precisión da a consta de un leve incremento de FLOPs, y sin embargo, se opta por un punto medio entre ambos, 4 *heads*.

3. PROYECTO INNOVADOR

Los métodos actuales de segmentación semántica se basan principalmente en sofisticadas redes neuronales convolucionales (CNN) y arquitecturas derivadas, como U-Net [RFB15], para la

extracción de características locales a nivel de píxel. Sin embargo, estas no terminan de explotar completamente la interacción entre celdas o conjuntos de píxeles, pues quedan restringidas al aprendizaje en la grilla de la imagen, con los vecinos físicos y, por ende, se pierden las relaciones productoras de características holísticas que, en muchas ocasiones, aportan información útil para la correcta interpretación automática de los datos y su anatomía en conjunto. En este documento se expone la idea de expandir la capacidad de las arquitecturas del estado del arte para la segmentación de imágenes bidimensionales de vasos retinianos mediante la inserción de módulos y técnicas basados en características topológicas. Principalmente, se recurre al teorema de la Teoría de la Información: la inclusión de información adicional relevante no perjudica al modelo, sino que mantiene su desempeño o lo mejora [SJ24].

En primer lugar, se reacondicionará la descrita arquitectura Vision GNN, diseñada exclusivamente para tareas de reconocimiento en imágenes y detección de objetos, para producir matrices 2D que sinteticen características topológicas extraídas en cada *patch* con la influencia de los *patches* vecinos. Estas imágenes, por medio de otras capas neuronales, serán refinadas a lo largo del entrenamiento hasta finalmente extraer la información adecuada de ellas. La salida resultante de esta será fusionada con otro modelo especializado en segmentación, y de nuevo, la salida se procesará con más capas. La posición exacta de la fusión se determina con ensayo y error, examinando las métricas obtenidas para los mismos datos. Cada uno de los hiperparámetros de la ViG deben ser configurados adecuadamente, pues vienen optimizados para las tareas mencionadas y en el dataset ImageNet [DDS09]. Asimismo, deben personalizarse algunos aspectos no concretados en el artículo

original, al ser este realmente sucinto en la fundamentación teórica del diseño, como la manera en la que se extraen las features de los *patches* o la forma en la que se determinan los *K*-vecinos.

Además del bloque anterior, se propone incluir un módulo de enriquecimiento explícito por homología persistente basado en imágenes de persistencia (PI), que no son más que la representación matricial en píxeles de los diagramas de persistencia conformada por sumas ponderadas de funciones gaussianas centradas en cada punto. Así, las imágenes 2D con las que se alimenta el modelo vendrán acompañadas de sus PI, es decir, información adicional de utilidad, ya sea como un nuevo canal o mediante otros métodos, como *Vector Stitching* [SJ24]. Dado que el cómputo de estas imágenes resulta extremadamente tardo, el entrenamiento se alargaría en exceso, por lo que convendría buscar alternativas que aceleren el cómputo. Una gran opción sería PI-Net [SCR20], una red convolucional que genera rápidamente las imágenes de persistencia sin seguir los pasos de su cálculo, aunque evidentemente requiere de un previo entrenamiento acorde, para el cual se debe contar de antemano con numerosas PI construidas manualmente. Si esta opción no convence, no es complejo hallar otras alternativas que apuran tiempos con respecto a las bibliotecas más populares, como puede ser el software Cubical Ripser [KSA20] para calcular la homología persistente en complejos cúbicos. De cualquier modo, elaborando las PI se consiguen representaciones topológicas robustas a variaciones de iluminación, FOV y otros condicionantes.

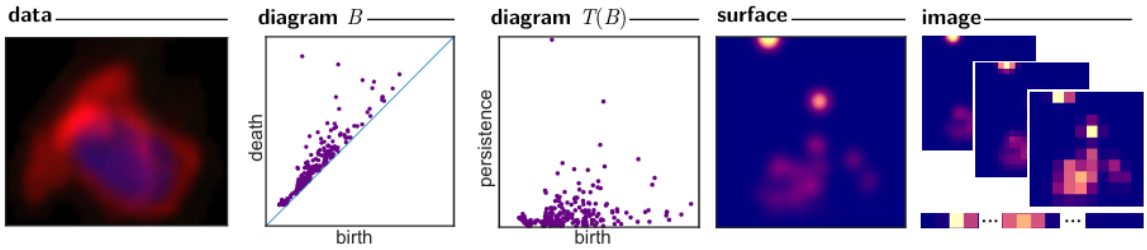


Ilustración 2. Ejemplo de obtención de imágenes de persistencia. Fuente: [ACE16]

Inclusive, se comprueba el efecto de añadir una función de coste en la que, en uno de sus términos, se involucre una comparación de características topológicas, como pueden ser la distancia *Bottleneck* o la distancia *Wasserstein*. No obstante, las anteriores necesitan de nuevo el cálculo previo del diagrama de persistencia de la imagen segmentada y de la verdad fundamental, por lo que en caso de exigir un proceso eficiente, la solución más sensata podría pasar por escoger la simple diferencia entre el número de Betti en dimensión 0, acerca de las componentes conexas. Su elección tiene un sentido lógico, ya que los vasos deberían conformar una única región conectada, alentando al modelo a unir aquellos trazos predichos separados de la estructura principal para minimizar la pérdida en entrenamiento y lograr resultados topológicamente similares en homología con los respectivos *ground truth*.

Por último, siguiendo la tendencia de considerar a la topología para evaluar la calidad de los modelos en labores de segmentación, se utilizan agregan métricas pertinentes a las típicamente usadas, pudiendo resultar en AUC para ROC o PR, F1-score, IOU, coeficiente de correlación de Matthews, ccDice [RMP24], etc.

En resumen, con este planteamiento se espera lograr segmentaciones de las ramificaciones muy cercanas al *ground truth* y equiparables o superiores a los modelos del SOTA, junto a una

arquitectura fácilmente extrapolable a cualquier otro tipo de problema de segmentación. Básicamente, los detalles locales a nivel de píxel de las retinografías serán capturados por las convoluciones de la U-Net u otro diseño, mientras que las características globales, contextuales, sobre la conectividad, serán grabadas por la GNN, procesando la imagen como un grafo, donde los nodos representantes de píxeles espacialmente alejados en la grilla pueden estar conectados e intercambiar información de regiones distantes. Ambas técnicas son complementarias y se potencian mutuamente. Al mismo tiempo, se desea insertar información topológica de la imagen de entrada por medio de diagramas o imágenes de persistencia, para mejorar aún más la segmentación, pues la GNN no necesariamente la computa, de nuevo enriqueciendo el proceso. Con el objetivo de garantizar la equivalencia topológica y la preservación de la estructura durante el entrenamiento, así como una segmentación coherente de los vasos en la validación y testeo, se propone también incorporar una función de pérdida donde, a parte de la habitual entropía cruzada binaria, se agregue un término ponderado con la distancia *Bottleneck*, la distancia *Wasserstein* o la diferencia entre el número de componente conexas. El valor que cada uno de los factores mencionados proyecta sobre el problema propuesto será evaluado mediante métricas adecuadas y rigurosas, típicamente utilizadas en el ámbito, como ROC-AUC, PR-AUC, F1-score, IOU o coeficiente de correlación de Matthews, además de otras que contemplen la topología, como ccDice.

4. DISEÑO DE ALGORITMO

Partiendo del planteamiento de la sección anterior, donde idealmente se han considerado recursos infinitos, se procede con el diseño de

una pequeña implementación acorde a la situación individual de desarrollo, con restricciones tanto de recursos como de tiempo, y que más tarde será objeto de las pruebas de evaluación.

Siguiendo las pautas establecidas, se decide implementar una arquitectura U-Net, en su forma original, sin modificaciones sustanciales respecto a las planteadas en el artículo [RFB15], salvo el uso de *padding* en las convoluciones para no perder filas y columnas de píxeles. Las razones de su preferencia son claras: es un diseño relativamente sencillo, con una reducida cantidad de parámetros, que produce resultados de segmentación sobresalientes. De hecho, en la clasificación mostrada en [2], relativa al problema de segmentación de vasos sanguíneos retinianos en el dataset CHASE_DB1 [FRH12], la U-Net se halla en el puesto undécimo, a pesar de haber sido lanzada hace ya 10 años, en 2015. Esta gran capacidad radica en su estructura *autoencoder*. El camino de la izquierda se dedica a la contracción (codificador), es decir, la aplicación de sucesivas capas de convolución, funciones de activación no lineales y capas de agrupamiento máximo, igual que el bloque de extracción de una CNN, desembocando en un conjunto creciente de mapas de características de cada vez menor resolución. Al llegar al final de este recorrido (*bottleneck*), donde se tienen las características de más alto nivel, se comienza un camino de expansión (decodificador), que hace lo opuesto al anterior: reduce el número de características y aumenta su resolución. Además, para preservar detalles finos se concatenan los mapas de estos niveles del decodificador con los correspondientes del codificador (*skip-connections*). El código pertinente de la U-Net de este TD ha sido factible gracias a la referencia de [Aro20].

Por otro lado, se quiere fusionar la salida de una Vision GNN con la máscara de segmentación preliminar generada por la U-Net anterior.

Para ello, se opta por la arquitectura ViG isotrópica ya explicada en su variante *tiny*, pues es la más sencilla de construir para esta mera demostración y la que menos parámetros requiere. Los hiperparámetros se establecen según se indica en el artículo original: 196 nodos, aumento lineal del campo receptivo o número de vecinos desde 9 hasta 18, a lo largo de 12 bloques ViG, siendo 192 el número de características por nodo calculadas con la intervención de la vecindad. Estas características no varían en tamaño, por lo que se facilita la escalabilidad y la aceleración *hardware*. La materialización de esta arquitectura no hubiera sido posible sin la ayuda de [Mir24], donde también se han señalado los problemas con la omisión de detalles del *paper* [HWG22]. Dentro de dichos obstáculos de reimplementación se destacan:

- Forma de extracción de características de los nodos. Se decide incluir una secuencia de capas de transformaciones lineales afines completamente conectadas, seguidas cada una de una normalización del lote y una función de activación GELU.
- Medición de la distancia para determinar una vecindad. El modelo original utiliza el algoritmo de *K* vecinos más próximos (KNN) dilatado, pero no especifica qué métrica de distancia se sigue. Para evitar problemas, en este TD se opta por la similitud por coseno y posterior selección de los *K* nodos más parecidos, pues se indica que es mucho más eficiente computacionalmente y alcanza una precisión parecida.

El cómputo de las imágenes de pérdida se descarta del diseño de esta prueba dado su elevadísimo coste computacional, que incrementa el tiempo de entrenamiento enormemente. La comentada red PI-Net [SCR20] requiere de un entrenamiento, igualmente laborioso, y la biblioteca Cubical Ripser [KSA20] lamentablemente resulta incompatible con la versión de algunas dependencias vitales.

En cuanto a la función de coste, se opta por aquella más eficiente: la mera comparación entre el número de componentes conexas halladas en la imagen predicha y las observadas en su verdad fundamental. Estas últimas pueden sustituirse por la información a priori de que deberían constituir un único elemento unido, reduciendo a la mitad los cálculos, aunque en tal caso no se recomienda emplear imágenes demasiado transformadas que desplacen los vasos fuera de los límites de la matriz, pues algunas ramificaciones podrían retornar y aparecer como nuevas componentes conexas que aumentarían el error.

La forma en la que se fusionan las salidas de la U-Net y de la Vision GNN se deja a elección del usuario. Puede, por ejemplo, inclinarse por usar un canal para cada imagen o, simplemente, por la multiplicación de una matriz por la otra, elemento a elemento. En la implementación realizada, se opta por esta última pues es la que mejores métricas ha provocado. La salida de esta operación deberá transmitirse por otras capas de convolución y funciones no lineales para una correcta interpretación del sistema con el paso de las épocas, finalmente generando una sola imagen de un canal, con las mismas dimensiones que la imagen en entrada, representando a la máscara de segmentación de los vasos retinianos.

El código de la implementación, convertido a un simplificado pseudocódigo más adelante, se puede acceder sin restricciones en el repositorio https://github.com/javserjod/unet_vig_segmentation.

Algoritmo 1. U-Net

*(B: *batch size* | C: canales, características | H, W: altura y anchura imagen)
Block = Secuencia de {Conv2D, BN, ReLU, Conv2D, BN, ReLU}
InputBlock = *Block*
EncoderBlock = Secuencia de {*Block*, MaxPool2D(2x2)}
DecoderBlock = Secuencia de {Interpolación, Concatenación, *Block*}
OutputBlock = Secuencia de {Conv2D, Sigmoide}

1: *InputBlock*. [B, 3, H, W] -> [B, 64, H, W]
2: *EncoderBlock*. [B, 64, H, W] -> [B, 128, H//2, W//2]
3: *EncoderBlock*. [B, 128, H//2, W//2] -> [B, 256, H//4, W//4]
4: *EncoderBlock*. [B, 256, H//4, W//4] -> [B, 512, H//8, W//8]
5: *EncoderBlock*. [B, 512, H//8, W//8] -> [B, 1024, H//16, W//16]
6: *DecoderBlock*. [B, 1024, H//16, W//16] -> [B, 512, H//8, W//8]
7: *DecoderBlock*. [B, 512, H//8, W//8] -> [B, 256, H//4, W//4]
8: *DecoderBlock*. [B, 256, H//4, W//4] -> [B, 128, H//2, W//2]
9: *DecoderBlock*. [B, 128, H//2, W//2] -> [B, 64, H, W]
10: *OutputBlock*. [B, 64, H, W] -> [B, 1, H, W]

Algoritmo 2. Vision GNN isotrópica *tiny* con cabeza ajustada

*(B: *batch size* | N: número de *patches* | C: canales, características | H, W: altura y anchura imagen | h, w: altura y anchura de cada *patch* | p: *patches* por dimensión. En imágenes cuadradas -> p = raíz de N)
Patch embedding = Secuencia de {Linear, BN, GELU} x6
Pose embedding = Suma de [N, C] número aleatorios en rango [0, 1]
TwoLayerNN = Secuencia de {Linear, BN, GELU, Linear}
GCN = Secuencia de {Agregación, Multi-head Update}
ViGBlock = Secuencia de {Creación de grafo, Guardar copia 1, *TwoLayerNN*, *GCN*, GELU, *TwoLayerNN*, Concatenar copia 1, Guardar copia 2, *TwoLayerNN*, GELU, *TwoLayerNN*, Concatenar copia 2}
Cabeza de la red = Secuencia de {Redimensionamiento [B, N, C] -> [B, C, p, p], Conv2D [B, C, p, p] -> [B, C//2, p, p], ReLU, Conv2D [B, C//2, p, p] -> [B, C//4, p, p], ReLU, Conv2D [B, C//4, p, p] -> [B, 1, p, p], Sigmoide, Interpolación [B, 1, p, p] -> [B, 1, H, W]}

1: *Patchifier*. [B, C, H, W] -> [B, N, C, h, w]
2: *Patch embedding*. [B*N, C*h*w] -> [B, N, C]
3: *Pose embedding.* [B, N, C] -> [B, N, C]
4: *ViGBlock* x12, elevar vecinos de 9 a 18 linealmente. [B, N, C] -> [B, N, C]
5: *Cabeza de la red*. [B, N, C] -> [B, 1, H, W]

Algoritmo 3. Combinación de Vision GNN isotrópica *tiny* y U-Net

*(B: *batch size* | C: canales, características | H, W: altura y anchura imagen)
Capas de procesamiento de fusión = Secuencia de {Conv2D [B, 1, H, W] -> [B, 1, H, W], ReLU, Conv2D [B, 1, H, W] -> [B, 1, H, W], ReLU}
Capa de salida = Secuencia de {Conv2D [B, 1, H, W] -> [B, 1, H, W], Sigmoide}

1: *U-Net* (Algoritmo 1).
2: *Vision GNN isotrópica tiny* (Algoritmo 2).
3: Fusión de salidas. Multiplicación, resultando en [B, 1, H, W]
4: *Capas de procesamiento de fusión*. [B, 1, H, W] -> [B, 1, H, W]
5: *Capa de salida*. [B, 1, H, W] -> [B, 1, H, W]

5. EXPERIMENTACIÓN

En este apartado se comienza describiendo la base de datos a utilizar en el entrenamiento, validación y fase de pruebas del modelo construido. Posteriormente, se ajustan los hiperparámetros de la arquitectura descrita en el apartado anterior y ya transcrita a código, teniendo presente las condiciones deficientes de desarrollo en cuanto a recursos materiales y temporales y experiencia previa. Se concluye analizando los resultados de diversas ejecuciones mediante las métricas adecuadas, sirviendo como método de comparación con otros modelos del estado del arte.

Se comenta que el modelo ha sido desarrollado en un entorno virtual Python gracias al administrador de paquetes Anaconda, dentro del IDE Visual Studio Code, donde se han descargado las dependencias PyTorch [1] requeridas, entre otras, además de CUDA Toolkit del fabricante NVIDIA, acorde con la GPU local a emplear, de tipo RTX 4070 Ti Super de 16 GB. Lograr el correcto funcionamiento final ha sido un desafío, debido a las incompatibilidades constantes entre distintas bibliotecas, algo que se ha logrado solventar consultando foros especializados como [3]. La aplicación de técnicas de preprocesamiento, el entrenamiento, la validación y las pruebas de evaluación se construyen en un cuaderno de Jupyter para facilitar la comprensión de cada fragmento de la secuencia operacional.

5.1. Conjunto de datos utilizado

Se ha descargado y desplazado al directorio de trabajo el dataset CHASE_DB1 [FRH12]. Este consta de 28 imágenes del fondo de ojo, es decir, retinografías. Fueron adquiridas durante un estudio llevado a cabo sobre ambos ojos de 14 estudiantes de primaria de diversas etnias de colegios en Inglaterra con el objetivo de demostrar la relación entre la tortuosidad de los vasos retinianos y factores de riesgo tempranos de afecciones cardiovasculares. La adquisición de dichas imágenes se realizó con una cámara de fondo de ojo Nidek NM-200-D ajustada a 30º de FOV (*field of view*) y con una resolución de 999x960 píxeles con 3 canales (RGB) de 8 bits cada uno, donde la intensidad de cada píxel expresa la cantidad de luz reflejada en cada plano de color. Dichas imágenes vienen en formato TIF, al igual que el primer conjunto de máscaras de segmentación binarias (1 canal, 8 bits) anotadas manualmente. El segundo conjunto de máscaras, en cambio, tiene la extensión PNG. Idealmente, los vasos se identifican por tener una menor intensidad que la retina, pero en este dataset las imágenes se caracterizan por una iluminación del fondo no

uniforme y un bajo contraste de los vasos con este. La división del dataset está predeterminada: 20 imágenes para entrenamiento (comprendiendo a las de validación) y 8 para las pruebas. A todas estas las acompañan sus respectivas segmentaciones *ground truth*, dos por cada imagen, ejecutadas manualmente por dos profesionales y supervisadas las del primero por el segundo, y viceversa, para una mayor rigurosidad. Estas máscaras de los vasos son vitales para el actual problema, pues la segmentación se encuentra dentro de la rama de aprendizaje supervisado.

Ya que el número de muestras es diminuto, el modelo elaborado memorizaría las características del set de entrenamiento durante el periodo de ajuste de parámetros, causando el llamado *over-fitting* o sobreajuste, desembocando en un mal desempeño en el testeo. Para corregir este contratiempo, se introducen artificialmente más muestras y, por ende, mayor variedad de datos que el modelo puede aprender. Por medio de la técnica llamada *data augmentation*, las imágenes originales son transformadas mediante unos criterios definidos por el usuario. En este trabajo, se establecen ciertas transformaciones siguiendo como guía el detallado análisis elaborado en [UBZ21] y modificando en función de los resultados obtenidos hasta alcanzar un punto de equilibrio entre eficiencia y eficacia. Se han acabado empleando, tanto para las imágenes de entrenamiento como para sus máscaras, volteos horizontales y verticales, rotaciones de cualquier ángulo, translaciones laterales y verticales y reducción y aumento de la escala. Los recortes aleatorios, transformaciones elásticas y ruido gaussiano provocaban una disminución considerable de la calidad. Lógicamente, las transformaciones deben ser parejas para imagen y su pertinente máscara, de lo contrario estarían incorreladas. Por otro lado, también se aplica a las imágenes por separado unos reajustes de nitidez y alteraciones de brillo, contraste,

saturación y tono. Finalmente, por limitaciones de rendimiento en el entorno de trabajo, la resolución de las imágenes se establece en 512x512, según sugiere [GAD22], acelerando así todos los procesos, a expensas de capturar detalles no tan refinados, lo cual no es un problema porque la verdadera intención es exhibir el potencial de la mezcla de información topológica con espacial. Las mencionadas transformaciones son accesibles fácilmente gracias a la biblioteca PyTorch. Esta opera con tensores, por lo que las imágenes deben ser convertidas a dichas estructuras antes de introducirlas en la red. Después, durante la operación, con un comando específico se mueven a la GPU donde se encuentre el modelo, aprovechando así el paralelismo que brinda. Al usar un DataLoader, se aplican las transformaciones a los datos al momento de ser requeridos en cada lote de cada época, de manera automática. El hiperparámetro que define la probabilidad con la que se transforman los datos se establece en 85%. Asimismo, se deciden dividir las muestras originalmente de entrenamiento en dos grupos: un 80% para entrenamiento real y el 20% restante para validación, una acción llamada *data splitting*. Una muestra del aumento de datos seguido se imprime en la Ilustración 3.

El tamaño de lote, *batch size*, a adoptar será fijado a 4, tras comprobar empíricamente el retroceso en el rendimiento al usar los cercanos valores de 2 y 8. Esto quiere decir que los parámetros de la red son actualizados según el valor de la función de pérdida computada al inyectarle 4 imágenes aumentadas en tiempo real.

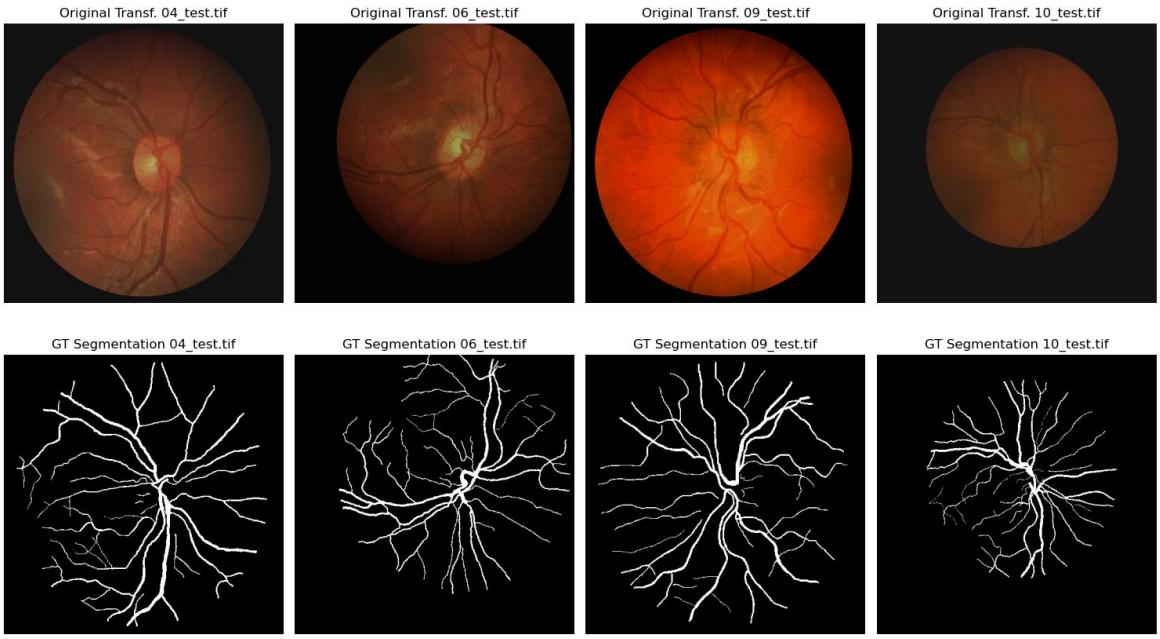


Ilustración 3. Ejemplo del *data augmentation* aplicado

5.2. Función de pérdida y métricas de evaluación

En las imágenes 2D obtenidas en retinografías, tan solo en torno a un 10% de los píxeles se corresponde con vasos sanguíneos. Por ende, el ratio de objeto a segmentar y fondo está muy desbalanceado [LJY21]. Escoger una función de coste que no contemple esta característica puede ser fatal para el entrenamiento, siendo el modelo subsecuente propenso a caer en un mínimo local rápidamente, donde la segmentación final pierda gran parte de los píxeles vasculares o, directamente, sea completamente fondo. Igual sucede con las métricas de evaluación y comparación de resultados, generando impresiones incorrectas de los modelos.

En pos de evitar el primero de estos inconvenientes, se comprueban los resultados al emplear tres tipos de función de coste: una basada en la pérdida Dice, otra que combina la entropía cruzada binaria con la pérdida Dice y una tercera que añade a los dos términos anteriores la diferencia entre componentes conexas. Los términos de estas

funciones de coste son ponderados según la importancia que se les quiera dar y en vista de los resultados generados.

A su vez, se han buscado las métricas más apropiadas para problemas de segmentación binaria con gran desbalanceo de clases, todas operando con la matriz de confusión resultante. El coeficiente Dice, equivalente en un contexto binario a F1-score y calculado desde la precisión (*precision*) y exhaustividad (*recall*), evalúa la superposición de la segmentación predicha y la verdad fundamental. **F1-score** se usa habitualmente cuando el problema es desbalanceado, como es el caso, pues ignora los verdaderos negativos. Además, permitirá comparar con las mejores arquitecturas del SOTA para el mismo dataset, listadas en [2], por lo que será la métrica de referencia principal para un umbral fijo. Se buscará el umbral de binarización de las probabilidades predichas para los píxeles que maximice este valor F1. Adicionalmente, para aportar mayor contexto, en el código se agrega el coeficiente de correlación de Matthews (MCC), que normalmente tiende a aumentar con F1. La representación de la curva ROC junto al área bajo esta, **ROC-AUC**, permite evaluar la capacidad de discriminación entre clases del modelo, entre fondo y vasos, a medida que se varía el umbral de decisión. De igual manera, la curva PR (*Precision-Recall*) posee gran utilidad en situaciones donde la clase positiva es mucho menor que la negativa. Su área bajo la curva, **PR-AUC**, permite cuantificar la capacidad de detección correcta de los positivos, lo cual interesa en este ejercicio. Aunque se ha comentado que la segmentación de vasos retinianos es un problema desbalanceado, se agrega al código el cálculo de la exactitud (*accuracy*) para ver los pequeños incrementos positivos, un valor no tan riguroso como los anteriores, ya que aun produciendo una máscara completamente negra alcanza valores superiores al 90%.

5.3. Pruebas realizadas

Los entrenamientos se ejecutan hasta que se produce la convergencia del modelo o se superan las 1250 épocas (en torno a unos 45 minutos). En caso de no reducir el error de validación tras un número de épocas suficiente, se fuerza una detención temprana (*early-stopping*). Este límite de tiempo se conoce como paciencia y para el actual desarrollo se ha fijado en 100.

Con la herramienta torchsummary de Pytorch es sencillo conseguir el número de parámetros entrenables de cada modelo para las entradas de dimensiones 3x512x512, a fin de poseer otro criterio con el que compararlos. La U-Net utilizada consta de 31M, mientras que su fusión con la Vision GNN alcanza los 41.2M de parámetros, suponiendo un aumento de casi un 33%.

En lo referente a la experimentación con el modelo del artículo base, se sabe que, generalmente, la red Vision GNN, por sí sola, es incapaz de realizar una segmentación a nivel de píxel, ya que trabaja con grafos donde los nodos son agrupaciones de píxeles, superpíxeles. La única excepción sería la elección de parches del tamaño de un solo píxel, una práctica que dispararía el costo computacional, junto a un complejo mecanismo de reconstrucción a partir de las predicciones nodales. El entrenamiento de una red compuesta por una ViG al uso conlleva segmentaciones completamente negras, es decir, puro fondo, debido al comentado desbalanceo de clases. Es por ello por lo que no se centran los experimentos en la segmentación con la ViG por sí sola, si no como un módulo acoplado capaz de expandir la información con la que trabaja la red de la que es parte. Esta red superior, conformada por la U-Net y dicha Vision GNN, sí que es razonable y merece ser evaluada en segmentación.

Antes de proseguir, debe hacerse un apunte acerca de los datos. Se mencionó que CHASE_DB1 incluía dos grupos de segmentaciones de verdad fundamental, cada una anotada por un especialista diferente. Tras varias pruebas, se concluye que entrenar y testear con la segunda, de nombre *2nd_manual*, desemboca en unas métricas superiores que las cosechadas con la primera, *1st_manual*. Por ello, en los siguientes apartados, aunque no se mencione expresamente, las máscaras *ground truth* son extraídas del conjunto *2nd_manual*.

En la Tabla 1, se comparan los mejores valores para las métricas de evaluación escogidas al experimentar con esta y con la U-Net individual con el mismo conjunto de hiperparámetros, en igualdad de condiciones. Puede observarse cómo el diseño propio, aun sin refinar en exceso, obtiene una métrica F1 máxima superior. Esto la situaría en posición bastante más alta que la U-Net en el ranking [2] siguiendo el criterio de F1, y levemente por encima según la ROC-AUC, aunque su posición exacta queda indeterminada al no indicarse ninguna regla ni condiciones de clasificación (como el set de testeo, porcentaje de *data splitting*, resolución de las imágenes, etc.). En concreto, la subida de F1 máxima es de un 1.15%, algo nada despreciable en este ámbito. El que la curva ROC y su AUC varíen mínimamente de un modelo a otro implica que la capacidad discriminativa frente a las probabilidades predichas no ha cambiado significativamente, o sea, la separabilidad de las clases a medida que se varía el umbral de decisión es casi idéntica de un modelo a otro. En cambio, el incremento de la F1 máxima sugiere un mejor balance entre precisión y exhaustividad para el umbral óptimo, (concretamente, el umbral 0.4902), es decir, un mejor equilibrio entre falsos positivos y falsos negativos.

Tabla 1. Métricas de evaluación para los modelos estudiados en testeo

Modelo	F1 máx.	ROC-AUC	PR-AUC
U-Net	0.8181	0.9889	**0.8964**
U-Net + ViG	**0.8276**	**0.9890**	0.8913

Se comenta que la función de pérdida que mejora el rendimiento respecto a F1 máximo y ROC-AUC del modelo para los casos anteriores es aquella compuesta por la métrica Dice y la entropía binaria cruzada, esta última ponderada a 0.5. La otra integrante planteada, que penaliza las diferencias entre el número de componentes conexas, únicamente empeora o no altera el rendimiento de dichas métricas, posiblemente por un defectuoso diseño acentuado por el incongruente número de componentes conexas del *ground truth* y por la deformación en el reescalado de las imágenes, por lo que se omite y se sugiere una revisión de esta de cara al futuro. Esto desemboca en una homología diferente de la del *ground truth*, con conectividad imperfecta, por lo que métricas como ccDice tienen malos resultados, rondando siempre el valor 0.1. Como puntualización de lo anterior, se destaca que algunas imágenes *ground truth*, como las ilustraciones previas, presentan vasos desconectados del conjunto, lo cual puede haber entorpecido a la función planteada, enfocada en conseguir una sola componente conexa.

Durante la experimentación, se ha vislumbrado de primera mano el efecto negativo en el desempeño de incrementar el número de *heads*, al igual que sugieren los autores en el artículo original, a cambio de reducir los FLOPs, algo que, a juicio propio, no justifica la decisión, pues la carga computacional ahorrada es imperceptible. Esto concuerda con la revisión de este vista en [Mir24]. Además,

como se ha comentado, el modelo usado de ViG es el isotrópico *tiny*, y no por capricho: se comprueba que al elevar el número de características computadas por nodo y el número de módulos Grapher más FFN, aumenta el tiempo y costo computacional, y por si fuera poco, se reduce el rendimiento, por lo que no se consigue ningún beneficio. También se prueban numerosas combinaciones de hiperparámetros como, por ejemplo, alteraciones en la ponderación de las componentes en la función de pérdida. Esto es infructuoso, al igual que la comprobación del efecto de inyectar la información proporcionada por la Vision GNN a partir de la imagen original en diversos puntos del modelo U-Net, como en el propio *bottleneck* de la U-Net o incluso en niveles más superiores del nivel expansivo, con lo que se procedería más tarde con su interpolación conjunta con los mapas de características de alto nivel de la U-Net.

Se alcanzan resultados esclarecedores al incrementar el número de *patches* por imagen, lo que equivale a no combinar tantos píxeles en una región. Debido a lo que se cree que es un *bug* en el código, el máximo que se puede emplear es 900, desde los 196 predeterminados por los autores del Vision GNN original. Los resultados se imprimen en la Tabla 2. A causa a las reiteradas limitaciones, solo se ha podido realizar una prueba para cada combinación, aunque empleando siempre la misma semilla para asegurar la repetibilidad. El mejor resultado obtenido es aquel basado en 625 nodos, lo cual, en imágenes cuadradas como las que se lleva manipulando, se traduce a 25 nodos por lado. Se adelanta que este mismo es el mejor resultado alcanzado en cuanto a máximo F1-score y ROC-AUC de toda la experimentación, aunque probablemente no el mejor posible para la arquitectura, pues se posponen muchas pruebas consecuencia del tiempo limitado.

Tabla 2. Efecto de incrementar el número de nodos

Modelo	F1 máx.	ROC-AUC	PR-AUC
U-NetViG 196 nodos	0.8276	0.9890	0.8913
U-NetViG 625 nodos	**0.8290**	**0.9893**	**0.8934**
U-NetViG 900 nodos	0.8227	0.9884	0.8886

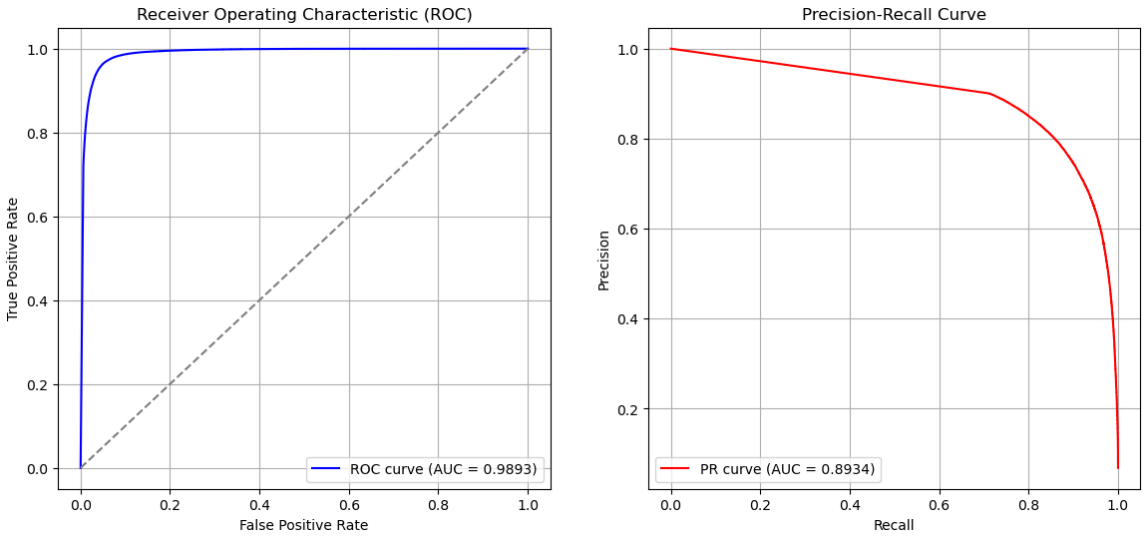


Ilustración 4. Curvas ROC y PR, junto a sus AUC, para el mejor resultado obtenido, mostrado en Tabla 2

En las Ilustración 5, cada columna se corresponde con una de las 4 primeras muestras del conjunto de testeo de CHASE_DB1, reescaladas a 512x512, precisamente mostradas en la primera fila de esta. Cada par proviene de un único paciente, siendo la primera de ellas para el ojo izquierdo y la segunda para el derecho. En la segunda fila se representan las segmentaciones *ground truth* del segundo especialista. En la tercera fila se trazan las segmentaciones producidas por el mejor modelo de fusión U-Net con ViG, indicado en la Tabla 2, con 625 nodos. Ya en la Ilustración 6 e Ilustración 7 se superponen las respectivas segmentaciones generadas (en rojo) sobre sus verdades fundamentales (en blanco), cada par en su propia

fila. Se aprecia una casi perfecta segmentación de los vasos principales, más gruesos y largos, pero claras deficiencias en ciertas ramificaciones finas donde radica la verdadera dificultad de esta actividad consecuencia de las ya comentadas imperfectas condiciones de la imagen original.

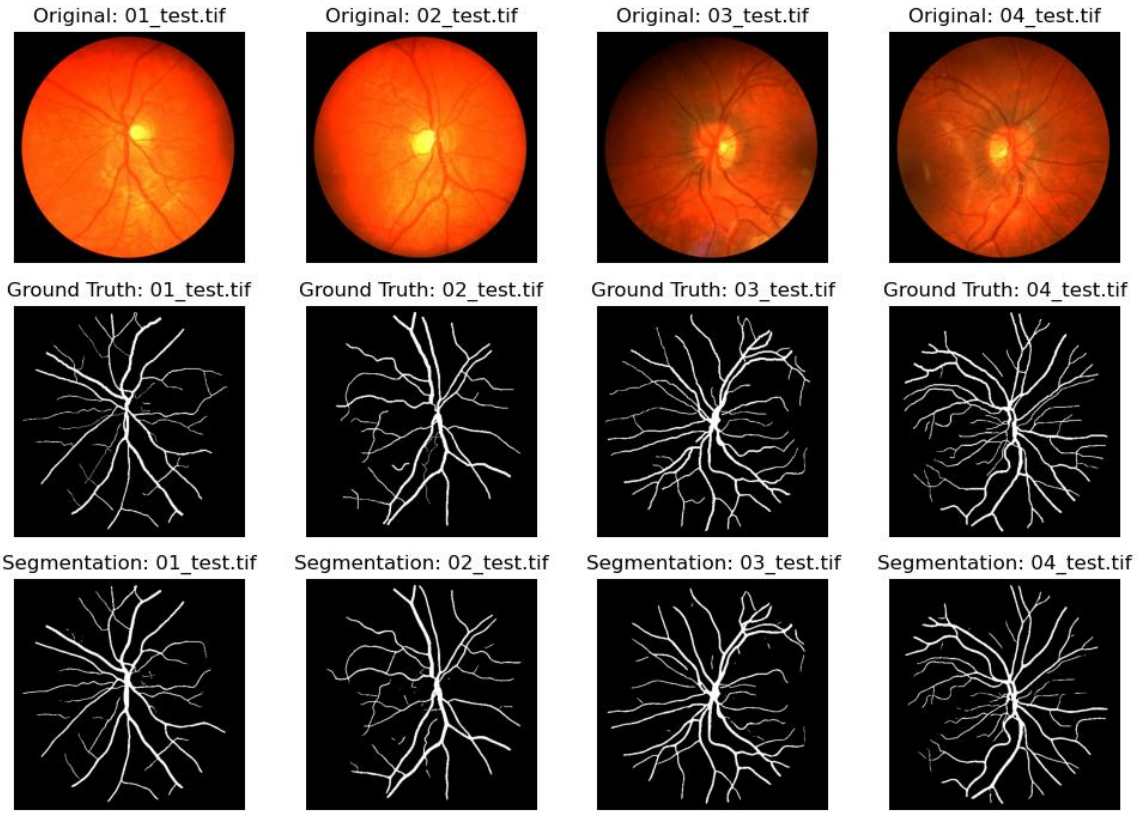


Ilustración 5. Primeras 4 muestras de testeo de CHASE_DB1, junto a sus *ground truth* y las segmentaciones generadas por la fusión de U-Net y Vision GNN

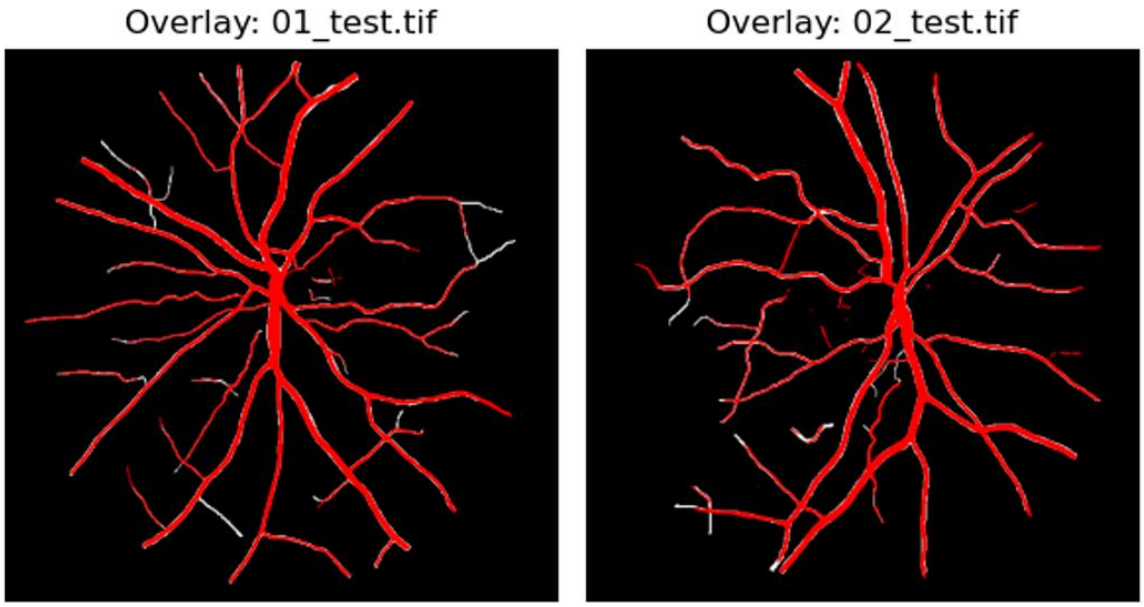


Ilustración 6. Primer paciente: superposición de segmentaciones del mejor modelo (en rojo) sobre *ground truth* (en blanco)

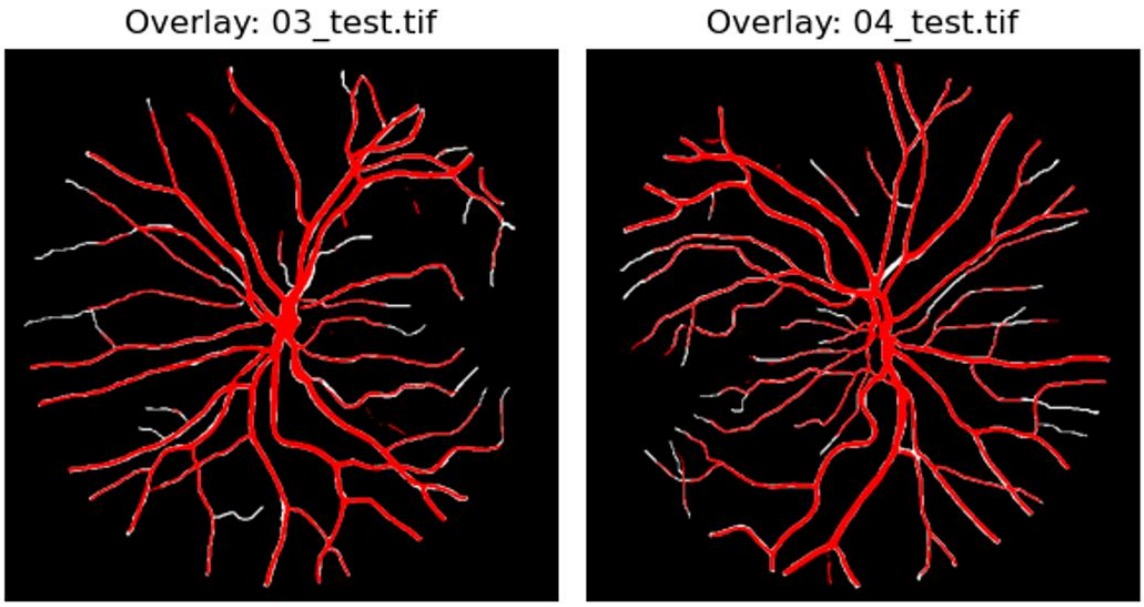


Ilustración 7. Segundo paciente: superposición de segmentaciones del mejor modelo (en rojo) sobre *ground truth* (en blanco)

Asimismo, si se extrema el *data splitting* ejercido para separar el set de entrenamiento original en el subset de entrenamiento y subset de validación, cambiándolo del hasta ahora usado ratio 80-20 hasta 90-10 y activando la misma semilla, las diferencias entre el modelo U-Net básico y la propia fusión U-Net con Vision GNN se acrecientan. En la Tabla 3 se aprecia cómo la fusión propuesta tiene una capacidad de generalización muy superior, reflejada en ROC-AUC, pues se ve beneficiada al ser capaz de extraer más patrones complejos en una situación de igualdad con la U-Net donde se tiene más variedad en entrenamiento y menos en validación. Con la máxima F1, métrica que no considera los verdaderos negativos, se observa que la media armónica óptima entre *precision* y *recall*, el equilibrio entre estas, se mantiene mucho mejor, ergo, la calidad de la segmentación de la clase de interés, los vasos, es sorprendentemente superior a la del *autoencoder*, que sí que se ve mermada.

Tabla 3. Efecto de emplear *data splitting* 90-10

Modelo	F1 máx.	ROC-AUC	PR-AUC
U-Net	0.7847	0.9781	0.8555
U-Net + ViG	**0.8164**	**0.9874**	**0.8883**

Con estas tres pruebas adjuntas, se puede confirmar que el módulo de Vision GNN aporta información topológica útil auxiliar, ya que se reducen los falsos negativos y los falsos positivos, traducido a un mejor *recall* y mejor *precision*, respectivamente, ambos plasmados en la métrica F1-score, cuyo valor máximo se ha constatado considerablemente superior al de la U-Net para todos los casos vistos.

6. CONCLUSIONES

A lo largo de este trabajo, a la par que se analizaba un artículo de referencia y se daba una opinión crítica a cada uno de sus aspectos más reseñables, se ha explorado una fracción del potencial que subyace en el planteamiento combinatorio de información espacial y topológica en el contexto planteado de segmentación de vasos retinianos en imágenes bidimensionales. No obstante, debido a las limitaciones de tiempo y de recursos materiales, no ha sido posible explorar todas las variantes, optimizaciones y experimentaciones posibles. Existen múltiples direcciones futuras que podrían potenciar aún más su rendimiento, incluyendo ajustes en la parametrización, refinamientos en la integración con otros enfoques y evaluaciones en escenarios más amplios. Estos aspectos quedan abiertos para estudios posteriores, con el objetivo de explotar al máximo las capacidades de la arquitectura y profundizar en su impacto dentro del área de aplicación. Por ejemplo, sustituir el modelo Vision GNN isotrópico por uno en pirámide sería conveniente para explorar la propiedad multiescalar de las imágenes como ya se hace en la U-Net. Igualmente, un estudio de ablación similar al realizado en [HWG22] sería revelador acerca de la influencia de cada componente de la ViG en la actual arquitectura, activando y desactivando las capas totalmente conectadas en el Grapher y la contigua FFN para refinamiento de características. Nuevos mecanismos de fusión de las redes quizás traen consigo una mejora del rendimiento (concatenación en *bottleneck* o tras la U-Net, ViG en cada escalón del camino de contracción…), así como otras funciones de pérdida más pulidas. Insertar las imágenes de persistencia como un nuevo canal o con *Vector Stitching* igualmente debería suponer un aumento positivo de las métricas de evaluación, y también entrenar con imágenes en resolución nativa. En otras palabras, la cantidad de alternativas

posibles es inabarcable dentro de las limitaciones impuestas por el trabajo. Aun así, los resultados experimentales, por sí solos, constituyen evidencia suficiente para esclarecer plenamente el alcance de la idea propuesta. Este análisis preliminar puede abrir la puerta a estudios más detallados y exhaustivos en el futuro dentro del campo de segmentación de imágenes médicas.

7. REFERENCIAS

7.1. Referencias principales

"En este trabajo, se ha hecho uso de la IA ChatGPT (GPT-4) para labores de 1. Resumen y Síntesis de secciones, 2. Explicación de Conceptos Complejos, ~~3. Creación de Introducciones o Contextos~~, ~~4. Interpretación y Análisis de Resultados~~, ~~5. Generación de Preguntas de Investigación~~, 6. Redacción de Conclusiones y Discusión, 7. Paráfrasis y Reformulación, de problemas, ideas y conceptos, ~~8. Creación de Presentaciones o Resúmenes Visuales~~, ~~9. Generación de Ejemplos o Casos Prácticos~~, ~~10. Creación de Notas o Material Didáctico~~, 11. Soporte en la Escritura de secciones o Propuestas"

[Aro20] Arora, A. (2020). *U-Net A PyTorch Implementation in 60 lines of Code*. https://amaarora.github.io/posts/2020-09-13-unet.html

[FRH12] Fraz, M. M., Remagnino, P., Hoppe, A., Uyyanonvara, B., Rudnicka, A. R., Owen, C. G., & Barman, S. A. (2012). An ensemble classification-based approach applied to retinal blood vessel segmentation. *IEEE Transactions on Biomedical Engineering, 59*(9), 2538–2548. https://doi.org/10.1109/TBME.2012.2205687

[GAD22] Galdran, A., Anjos, A., Dolz, J., & et al. (2022). State-of-the-art retinal vessel segmentation with minimalistic models. *Scientific Reports, 12*(6174). https://doi.org/10.1038/s41598-022-09675-y

[HWG22] Han, K., Wang, Y., Guo, J., Tang, Y., & Wu, E. (2022). Vision GNN: An image is worth a graph of nodes. *arXiv*. https://arxiv.org/abs/2206.00272

[Mir24] Mirzaei, M. (2024). *Vision Graph Neural Networks: A Replication Study*. McMaster University. https://github.com/mirzaim/VisionGNN

[RFB15] Ronneberger, O., Fischer, P., & Brox, T. (2015). U-Net: Convolutional Networks for Biomedical Image Segmentation. *arXiv*. https://arxiv.org/abs/1505.04597

7.2. Referencias secundarias

[ACE16] Adams, H., Chepushtanova, S., Emerson, T., Hanson, E., Kirby, M., Motta, F., Neville, R., Peterson, C., Shipman, P., & Ziegelmeier, L. (2016). Persistence images: A stable vector representation of persistent homology. *arXiv*. https://arxiv.org/abs/1507.06217

[CXG22] Chen, S., Xie, E., Ge, C., Chen, R., Liang, D., & Luo, P. (2022). CycleMLP: A MLP-like architecture for dense prediction. *arXiv*. https://arxiv.org/abs/2107.10224

[DDS09] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[HZR15] He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. *arXiv*. https://arxiv.org/abs/1512.03385

[JCT23] Jiang, J., Chen, X., Tian, G., & Liu, Y. (2023). ViG-UNet: Vision Graph Neural Networks for Medical Image Segmentation. *arXiv*. https://arxiv.org/abs/2306.04905

[KSA20] Kaji, S., Sudo, T., & Ahara, K. (2020). Cubical Ripser: Software for computing persistent homology of image and volume data. *arXiv*. https://arxiv.org/abs/2005.12692

[LJY21] Li, Z., Jia, M., Yang, X., & Xu, M. (2021). Blood Vessel Segmentation of Retinal Image Based on Dense-U-Net Network. *Micromachines, 12*(12), 1478. https://doi.org/10.3390/mi12121478

[LLC21] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. *arXiv*. https://arxiv.org/abs/2103.14030

[LSY23] Liu, Y., Shen, J., Yang, L., Bian, G., & Yu, H. (2023). ResDO-UNet: A deep residual network for accurate retinal vessel segmentation from fundus images. *Biomedical Signal Processing and Control*, 79, 104087. https://doi.org/10.1016/j.bspc.2022.104087

[LTZ22] Liu, Z.-Q., Tang, P., Zhang, W., & Zhang, Z. (2022). CNN-Enhanced Heterogeneous Graph Convolutional Network: Inferring Land Use from Land Cover with a Case Study of Park Segmentation. *Remote Sensing*, *14*(19), 5027. https://doi.org/10.3390/rs14195027

[RMP24] Rougé, P., Merveille, O., & Passat, N. (2024). ccDice: A topology-aware Dice score based on connected components. *Workshop on Topology- and Graph Informed Imaging*

Informatics (TGI3@MICCAI) (pp. 11-21). https://doi.org/10.1007/978-3-031-73967-5_2

[SCR20] Som, A., Choi, H., Ramamurthy, K. N., Buman, M., & Turaga, P. (2020). PI-Net: A deep learning approach to extract topological persistence images. *arXiv*. https://arxiv.org/abs/1906.01769

[SJ24] Stolarek, A., & Jaworek, W. (2024). Preserving information: How does topological data analysis improve neural network performance? *arXiv*. https://arxiv.org/abs/2411.18410

[UBZ21] Uysal, E. S., Bilici, M. Ş., Zaza, B. S., Özgenç, M. Y., & Boyar, O. (2021). Exploring the limits of data augmentation for retinal vessel segmentation. *arXiv*. https://arxiv.org/abs/2105.09365

[WWZ21] Wu, H., Wang, W., Zhong, J., Lei, B., Wen, Z., & Qin, J. (2021). SCS-Net: A scale and context sensitive network for retinal vessel segmentation. *Medical Image Analysis*, 70, 102025. https://doi.org/10.1016/j.media.2021.102025

Páginas web

[1] PyTorch. (s.f.). *PyTorch*. https://pytorch.org/

- Documentación de la librería empleada en la construcción de la red neuronal.

[2] Papers With Code. (s.f.). *Retinal vessel segmentation on CHASE_DB1*. Papers With Code. https://paperswithcode.com/sota/retinal-vessel-segmentation-on-chase_db1

- Ranking de las mejores redes neuronales para segmentar imágenes bidimensionales de vasos retinianos extraídas del dataset CHASE_DB1.

[3] Stack Overflow. (s.f.). *Stack Overflow*. https://stackoverflow.com/
- Foro especializado con el que se han resuelto problemas relativos al código.

[4] Fernández-Vega, A. (2017). *Enfermedades que se manifiestan a través de los ojos*. Clínica Oftalmológica Fernández-Vega. https://fernandez-vega.com/blog/enfermedades-se-manifiestan-traves-los-ojos/
- Blog de un especialista en el que se señala las patologías que pueden detectarse mediante el análisis de la retina.

APÉNDICE: CRONOLOGÍA DE ELABORACIÓN DEL TD

Se adjunta un cronograma a modo de registro diario de cada una de las actividades llevadas a cabo a lo largo del proyecto.

Fecha	Hora inicio	Hora fin	Actividad	Comentarios
23/12/2024	18:00	22:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
26/12/2024	19:00	20:30	Comprensión de requerimientos del TD	Interpretación de lo que realmente se pide en el TD
	20:30	22:00	Acondicionamiento del documento Word	Preparación de apartados según las indicaciones recibidas
	22:00	23:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
28/12/2024	19:00	22:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
29/12/2024	19:00	23:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
30/12/2024	14:00	16:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
	18:00	20:00		
31/12/2024	13:00	16:00	Consulta de fuentes de	Recopilación de artículos o vídeos relacionados y

			información	búsqueda de base del TD
1/1/2025	19:00	23:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
2/1/2025	18:00	22:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
4/1/2025	19:00	23:30	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
5/1/2025	14:00	15:30	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
	19:00	23:30		
6/1/2025	00:30	2:30	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
	18:00	23:00		
7/1/2025	14:00	16:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
	19:00	23:30		
8/1/2025	00:30	3:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
	13:00	15:30		
	19:00	23:30		
10/1/2025	19:00	23:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
11/1/2025	19:00	23:00	Consulta de fuentes de información	Recopilación de artículos o vídeos relacionados y búsqueda de base del TD
16/1/2025	19:00	23:00	Preparación de	Configuración del entorno

			experimentación	e IDE, descarga de librerías y otro software
17/1/2025	19:00	23:00	Preparación de experimentación	Configuración del entorno e IDE, descarga de librerías y otro software
18/1/2025	18:30	23:30	Programación de entrenamiento con PyTorch y U-Net	Conversión a código del *paper* de U-Net e implementación de código para entrenar
19/1/2025	18:00	23:50	Corrección de errores en código y programación de pruebas y métricas	Solución de errores, parametrización, pruebas en set test, métricas ROC-AUC, y Dice score
20/1/2025	14:00	16:00	Corrección y mejora del código, implementación de otras métricas	Se refina el código hecho hasta ahora y se añaden métricas MCC y ccDice
	19:00	23:00		
21/1/2025	14:00	16:00	Implementación *data augmentation* y funciones de pérdida	Transformaciones a retinografías y *ground truths*. Funciones de pérdida: Dice y combinación BCE-Dice
	19:00	23:30		
22/1/2025	12:00	15:00	Implementación de ViG	Se codifica la estructura de una Vision GNN
	19:00	23:00		
23/1/2025	18:00	22:00	Adaptación de ViG y comprobación de funcionamiento	Se adapta la ViG al problema de segmentación y se prueba el rendimiento
24/1/2025	18:00	22:00	Conexión ViG y U-	Se busca la manera

			net, pruebas de rendimiento	óptima de conectar ambos modelo para mejorar métricas
25/1/2025	18:00	23:00	Adaptación de ViG y comprobación de funcionamiento	Se barajan alternativas de diseño y acoplamiento de la ViG con la U-Net
26/1/2025	12:00	16:00	Adaptación de ViG y comprobación de funcionamiento	Se barajan alternativas de diseño y acoplamiento de la ViG con la U-Net
	18:00	23:00		
27/1/2025	11:00	15:00	Ejecución de pruebas	Se prueba la arquitectura actual con diferentes parámetros
	18:00	23:00	Redacción del documento del TD	Redacción de introducción y ajuste de secciones
28/1/2025	19:00	23:00	Ejecución de pruebas	Se prueba la arquitectura actual con diferentes hiperparámetros
29/1/2025	19:00	23:00	Ejecución de pruebas	Se prueba la arquitectura actual con diferentes hiperparámetros
31/1/2025	19:00	23:00	Redacción del documento del TD	Redacción de marco teórico y proyecto innovador
1/2/2025	19:00	23:00	Redacción del documento del TD	Redacción de diseño y experimentación
2/2/2025	18:00	23:00	Ejecución de pruebas	Se prueba la arquitectura actual con diferentes hiperparámetros

3/2/2025	19:00	22:00	Redacción del documento del TD	Redacción de experimentación y conclusiones
4/2/2025	12:00	15:00	Revisión y corrección del documento	Se repasa lo previamente redactado y se corrigen incongruencias
5/2/2025	12:00	16:00	Refinamiento de documento y redacción de conclusiones	Se completan algunos apartados y se redactan las conclusiones
	19:00	23:00		