Tecnológico de Costa Rica

Escuela de Ingeniería Electrónica

Programa de Licenciatura en Ingeniería Electrónica

Comparación de métodos modernos de segmentación semántica en imágenes de microscopía óptica de nematodos

Informe de Trabajo Final de Graduación para optar por el título de

Ingeniera en Electrónica con el grado académico de Licenciatura

Kimberly María Carvajal Méndez

Cartago, 24 de noviembre, 2022

Declaro que el presente documento de tesis ha sido realizado enteramente por mi persona, utilizando y aplicando literatura referente al tema e introduciendo conocimientos y resultados experimentales propios.

En los casos en que he utilizado bibliografía he procedido a indicar las fuentes mediante las respectivas citas bibliográficas. En consecuencia, asumo la responsabilidad total por el trabajo de tesis realizado y por el contenido del presente documento.

Kimberly María Carvajal Méndez

Cartago, 24 de noviembre, 2022

Céd: 1-0123-0456

Instituto Tecnológico de Costa Rica
Escuela de Ingeniería Electrónica
Trabajo Final de Graduación
Acta de Aprobación

Defensa de Trabajo Final de Graduación
Requisito para optar por el título de Ingeniero en Electrónica
Grado Académico de Licenciatura

El Tribunal Evaluador aprueba la defensa del trabajo final de graduación denominado *Comparación de métodos modernos de segmentación semántica en imágenes de microscopía óptica de nematodos*, realizado por la señora Kimberly María Carvajal Méndez y, hace constar que cumple con las normas establecidas por la Escuela de Ingeniería Electrónica del Instituto Tecnológico de Costa Rica.

Miembros del Tribunal Evaluador

M.Sc. José Miguel Barboza Retana
Profesor Lector

Dr. Adolfo Chaves Jiménez
Profesor Lector

Dr. José Pablo Alvarado Moya
Profesor Asesor

Cartago, 24 de noviembre, 2022

Resumen

Los nematodos son animales de aspecto vermiforme (forma de gusanos), semitransparentes y de tamaño microscópico. Hay especies de nematodos fitoparásitos y de vida libre, las cuales son dañinas o benéficas para los agroescosistemas respectivamente. Se realizan estudios de los nematodos con el fin de obtener conocimiento en cómo interactúan con los ecosistemas y cómo utilizarlos para el control biológico.

En este proyecto se desarrolla una comparación de los modelos de segmentación semántica STDC, PointRend y Segformer, proporcionados por la herramienta *MMSegmentation* basada en PyTorch, con el fin de identificar los píxeles correspondientes a nematodos en imágenes de microscopía óptica provenientes de una muestra de práctica cotidiana.

Se analizan los modelos en base a criterios de capacidad computacional, rendimiento y la columna del modelo que esta basada en redes convolucionales o transformadores. Se realizan experimentos para medir la estabilidad del modelo y el rendimiento con cuatro configuraciones de parámetros por modelo, con el uso de arquitecturas GPU para soportar la demanda de las redes de aprendizaje profundo.

Con el conjunto de datos disponible y las máscaras de segmentación generadas, los modelos analizados demuestran la capacidad de identificar nematodos en un 91 % de rendimiento en base a la métrica IoU.

Palabras clave: segmentación semántica, nematodos, MMSegmentation, transformers, redes convolucionales

Abstract

Nematodes are vermiform animals (worm-like), semitransparent of microscopic size. There are phytoparasites and free-living species, which are harmful or beneficial to agroecosystems, respectively. Studies of nematodes are conducted in order to gain knowledge on how they interact with ecosystems and how to use them for biological control.

This project compares the semantic segmentation models STDC, PointRend and Segformer, provided by the MMSegmentation toolbox based on PyTorch, in order to identify the pixels corresponding to nematodes in optical microscopy images from a sample of daily practice.

The models are analyzed based on criteria such as computational capacity, performance and the backbone's model, which is based whether on convolutional networks or transformers. Experiments are performed to measure model stability and performance with four parameter settings per model, using GPU architectures to support the demand of deep learning networks.

With the available dataset and the generated groundtruths, the analyzed models demonstrate the capacity to identify nematodes at 91 % performance based on the IoU metric.

Keywords: semantic segmentation, nematodes, MMSegmentation, transformers, convolutional networks

a mis queridos padres y a mi tío Keylor

Agradecimientos

Doy gracias a Dios y a mis padres Cinthya Méndez Rivera y Hermann Carvajal Carvajal, por acompañarme y guiarme por las etapas de mi vida que me permitieron llegar a donde estoy hoy.

A mami Jorley, mi abuela Zoraida, mis titos, padrinos y tíos que me apoyaron y brindaron consejos para seguir adelante, así como mi tío Keylor Ovares Rivera quién fue mi ejemplo a seguir.

A mis amigos del TEC, que mutuamente nos apoyamos en los estudios. Y a mi novio Jorge Brenes Alfaro quien me brindó su apoyo y compañía en todo momento.

Al Dr.-Ing. Pablo Alvarado Moya, por el conocimiento que me brindó como profesor y asesor de este proyecto, y por su disposición de ayudar ante cualquier duda.

Finalmente, al CeNAT por brindarme el acceso a la supercomputadora Kabré para la realización del proyecto.

Kimberly María Carvajal Méndez

Cartago, 24 de noviembre, 2022

Índice general

Índice de figuras

Índice de tablas

Lista de símbolos y abreviaciones

Abreviaciones

BiSeNet	Bilateral Segmentation Network
CWAGM	Color Watershed - Adjacency Graph Merge
DPT	Dense Prediction Transformer
ECC	Código con corrección de error (Error correction code)
FCN	Fully Convolutional Network
IoU	Intersección sobre Unión (Intersection over Union)
kNN	K-nearest neighbors
LEGION	Locally Excitatory Globally Inhibitory Oscillator Networks
LMBI	Local Maximum of Boundary Intensity
MLP	Multilayer Perceptron
OCRNet	Object-Contextual Representations Net
PCI	Interconexión de Componentes Periféricos (Peripheral Component Interconnect)
PCNN	Pulse Coupled Neural Network
RBF	Radial Basis Function Network
SIPLab	Laboratorio de Procesamiento de Señales e Imágenes
STDC	Short-Therm Dense Concatenate
SVM	Support vector machines

Capítulo 1

Introducción

1.1. Nematodos en la agricultura de Costa Rica

El sector agropecuario de Costa Rica influye en la economía nacional, el comercio, con la producción de alrededor del 70 % de la canasta básica alimentaria [1]. Los productores agropecuarios en el país se dedican a la producción de granos, hortalizas, frutas y café, así como empresas agropecuarias de gran escala dedicadas a la producción de banano, caña de azúcar, piña, naranja, entre otros productos [2].

La situación de la emergencia nacional por el COVID-19 a inicios de la década de 2020, provocó variaciones en la demanda y cambios en las cadenas de distribución debido a las medidas implementadas por la pandemia [1]; sin embargo, el sector agro cuenta con alrededor de 240.000 productores agropecuarios, contempla acciones y protocolos establecidos en el periodo 2020-2021 para continuar exportando y abasteciendo a los consumidores, de modo que se contempla una aportación económica cercana al 10 % del PIB [3]. En función de continuar y crecer con esta actividad económica, la aplicación de nuevas tecnologías es clave en la aceleración de la producción [4], así como en el control de plagas que afectan la agricultura costarricense, puesto que se han encontrado especies, como los nematodos, que son dañinas para productos como el café, el banano y otros [5].

Los nematodos son especies de aspecto vermiforme, lo que quiere decir que tienen forma de gusano. Se caracterizan por ser semitransparentes de tamaños microscópicos, midiendo como máximo 1 mm de longitud y 50 μm de ancho, necesitando de un microscopio para poder analizarlos [6], [7]. Para 2012 se contabilizaban 26.646 especies de nematodos [6]. En la figura 1.1 se muestra un ejemplo de un nematodo.

Estas especies se encuentran de manera abundante en el agua y suelos que tengan un nivel de humedad del 40 % al 60 %. Pueden ser tanto dañinos como benéficos para los agroecosistemas. Ejemplo de esto, es que se conocen alrededor de 4000 especies de nematodos que atacan las plantas, lo que facilita la entrada de otros agentes patógenos en el suelo y las raíces [8]. Sin embargo, entre las principales funciones de los nematodos está conservar la fertilidad del suelo, esto gracias a la movilización de nutrientes y la regulación

Figura 1.1: Imagen de un nematodo de una muestra de microscopía óptica.

de poblaciones de patógenos de plantas [5].

En los ecosistemas agronómicos se encuentran nematodos fitoparásitos, los cuales dependen de las plantas debido a que se alimentan de raíces, tallos y hojas [7] causando daños considerables que pueden derivar en la muerte o debilitamiento de las raíces, en lesiones, abultamientos o deformaciones de los tallos y las hojas de los cultivos [6]. También se encuentran nematodos de vida libre, que pueden ser beneficiosos en los procesos ecológicos siendo uno de sus alimentos otros nematodos [9]. Estos últimos se estudian por su potencial para ser utilizados en el control biológico, como parte de estrategias de disminución del uso de plaguicidas [7].

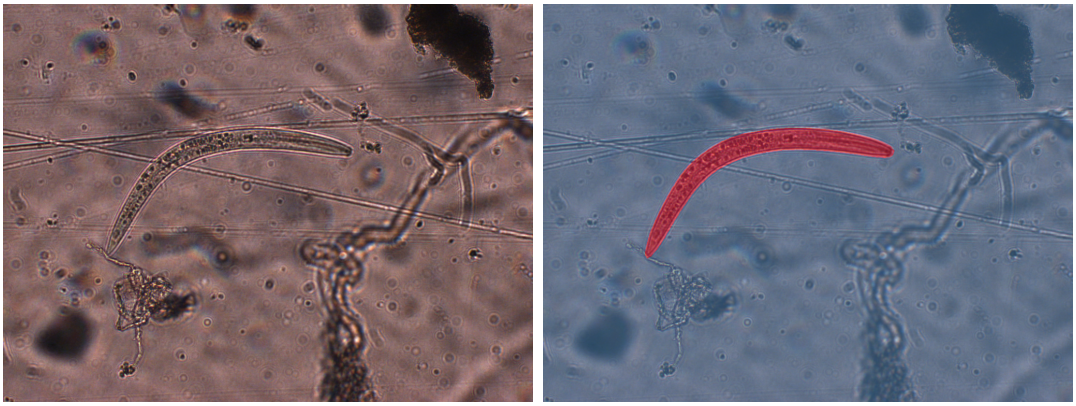
Debido al papel de los nematodos en la agricultura, se realizan estudios de áreas infestadas donde se puedan recolectar muestras que sean remitidas a un laboratorio para su estudio. Los estudios se realizan para identificar el género, la densidad de la población en una muestra e identificar patrones de comportamiento [7], lo que conduce a un mayor conocimiento en cómo interactúan con los ecosistemas y cómo utilizarlos para llevar a cabo el control biológico.

Un problema de este proceso es que, para llevar a cabo un análisis exhaustivo y con indicadores estadísticos de confianza aceptables, se deben procesar manualmente cientos de imágenes de muestras de suelo, lo cual se vuelve un trabajo tedioso, costoso y extenso debido a que el análisis de una sola muestra puede tardar desde 15 minutos hasta varias horas. Por ello, surge la necesidad de crear una herramienta de conteo e identificación capaz de optimizar el proceso de análisis de las imágenes microscópicas para disminuir el tiempo que toma realizar los estudios de manera manual.

1.2. Segmentación semántica de nematodos

La tarea de segmentación semántica es uno de los primeros pasos en el análisis automatizado de esas imágenes. Esta tarea consiste en asociar los píxeles de la imagen con etiquetas [10], esto con el fin de reconocer un conjunto de píxeles con características si-

milares, delimitando con precisión los elementos de una imagen [11]. Por ejemplo, las imágenes de microscopía óptica de nematodos se segmentan en regiones pertenecientes a los nematodos, el fondo y otras especies u objetos que se presenten en la muestra, como se ilustra en la figura 1.2. Una vez estimadas las regiones correspondientes a los nematodos, se acoplan otros módulos que permitan identificar las especies, contar los especímenes, o en secuencias de video, identificar comportamientos.



(a) Original **(b)** Segmentado

Figura 1.2: Ejemplo de un nematodo segmentado.

Para llevar a cabo una segmentación semántica, la imagen de entrada se somete a una disminución de resolución que permite llevar a cabo una extracción de características. Este proceso se puede llevar a cabo con un transformador o una capa convolucional con la ayuda de un módulo extra. De este proceso se obtiene un mapa de características de los píxeles de la imagen, el cual se reconstruye aumentando la dimensión al tamaño original, generando las etiquetas resultantes. Este proceso se resume en el diagrama de bloques de la figura 1.3.

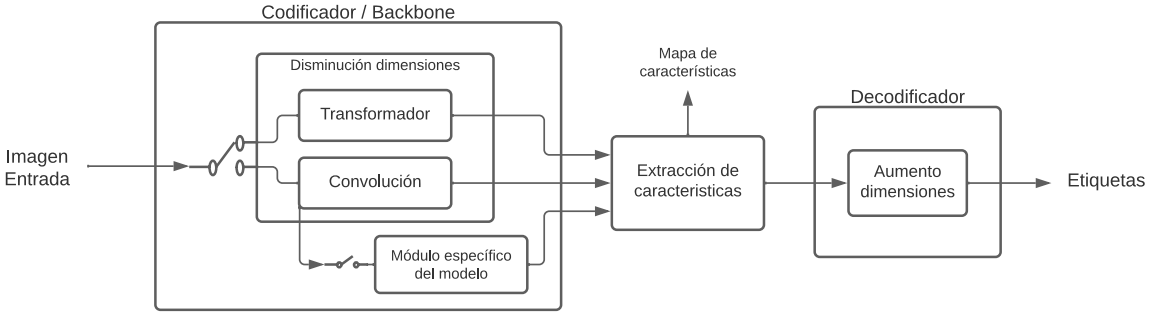


Figura 1.3: Diagrama de bloques de la segmentación semántica.

1.3. Antecedentes

El proyecto dentro del que se circunscribe el presente trabajo nace del "Programa interinstitucional de investigación en biodiversidad y ecología de organismos de suelo, con énfasis en sistemas de producción limpia y control biológico"[12] del Instituto Tecnológico de Costa Rica, con la propuesta de elaborar una herramienta computacional que permita agilizar la valoración de las imágenes de microscopía tomadas en el análisis de las muestras de suelo, raíz, etc.

Para atacar el problema de segmentación de nematodos, se divide el proyecto en tres ramas que comprenden: el análisis de forma con métodos clásicos de procesamiento digital de imágenes, el aumento de datos de forma sintética y la segmentación de imágenes.

Para el análisis de forma, se parte de la identificación de hitos o *landmarks* en el borde de los nematodos. Marín [13] utiliza modelos activos de forma (ASM) para definir la ubicación de un nematodo, dibujando un contorno a su alrededor. Seguidamente, Grüner [14] utiliza algoritmos de aprendizaje de variedades topológicas (*Active Dictionary Models*) para representar deformaciones no lineales, como las que exhiben los nematodos. Arroyo en [15] toma el trabajo realizado por Grüner en [14] como un subdominio de siluetas permitidas expresadas en hitos que se ajustan a las siluetas de nematodos en imágenes digitales, hasta hallar el mejor ajuste del nematodo en la imagen.

En la rama del aumento de datos se busca ampliar el conjunto disponible no solo con métodos clásicos como rotaciones, cambios de tamaño, entre otros. El enfoque en este caso, es la creación de imágenes sintéticas que emulen muestras realistas de microscopía óptica mediante redes generativas para evitar problemas de sobre-ajuste [16], [17].

Por último, en la rama de la segmentación, en primera instancia Gómez [18] prueba con modelos de segmentación clásicos en dos niveles, primeramente haciendo una comparación de algoritmos de extracción de características tales como PCNN, LEGION, CWAGM y *Meanshift* y segundo una evaluación de modelos clasificadores como MLP, RBF y kNN. Seguidamente Taylor [19] utiliza redes neuronales convolucionales con un conjunto de datos aumentado replanteado con una menor dimensión, donde se toman como datos segmentos de recta de una imagen de muestra, esto con el fin de detectar qué parte de este dato pertenece a un nematodo y definir el borde del espécimen en una imagen de muestra Por último, Jiménez [20] utiliza una combinación de redes convolucionales y modelos de forma con un conjunto de datos aumentado mediante la división de una imagen de muestra en "cuadros"de 40×40 píxeles.

El reto ahora es realizar una segmentación semántica donde se utilice un conjunto de datos con imágenes completas de muestras prácticas. En esta segmentación se busca segmentar el espécimen completo y no solo su borde. Debido a esto y a las métricas aplicadas, los trabajos realizados no son comparables entre sí; sin embargo, de estos trabajos previos se reutiliza el conjunto de datos de alrededor de 5000 imágenes, y de los bordes obtenidos en [19] se facilita la generación de etiquetas de los datos en máscaras de segmentación.

Los métodos implementados en el SIPLab, tienen una antigüedad al menos cinco años de haberse implementado. La visión por computadora es una rama que continúa avanzando y ha tenido en los últimos años propuestas con incrementos constantes en su desempeño de modo que existen nuevos aportes a la industria en relación al procesamiento y análisis de imágenes. La visión por computadora aporta nuevos avances en los últimos tres años en la tarea de segmentación, dichos avances aún no han sido probados y presentan potencial de aplicación en la detección y clasificación nematodos.

1.4. Sistema de segmentación semántica

En este proyecto se realiza una comparación de la efectividad de tres métodos de segmentación semántica modernos aplicados a imágenes de microscopía de nematodos. El desarrollo de este trabajo aporta en la innovación y uso de avances tecnológicos recientes en la segmentación aplicados a especies microscópicas o de aspecto vermiforme.

Los modelos de aprendizaje profundo requieren gran cantidad de datos para su buen funcionamiento. Es por ello que, con el conjunto de datos actual, se requiere una etapa de aumento de datos. Esta etapa se encarga de ampliar el conjunto de datos llevando a cabo transformaciones a las imágenes existentes como rotación, cambio de tamaño, cambiar el contraste, entre otros. Otra etapa necesaria es la de evaluación, la cual usando métricas específicas de segmentación, se comparan los rendimientos de los modelos aplicados, de tal manera que se cumpla con el diagrama de la figura 1.4 que ilustra las etapas de desarrollo del proyecto.

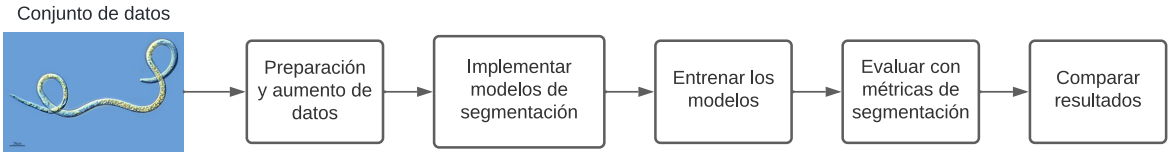


Figura 1.4: Etapas para clasificar imágenes mediante segmentación semántica.

1.5. Objetivos y estructura del documento

El objetivo de este proyecto es comparar experimentalmente métodos propuestos en los últimos tres años para la segmentación semántica, adaptándolos al caso de nematodos en imágenes de microscopía óptica. Conforme con lo planteado, se debe considerar la preparación del conjunto de datos disponible en el SIPLab, incluyendo la selección de técnicas de aumento de datos y separación del conjunto de datos. Previo a la implementación de los modelos se deben seleccionar al menos tres modelos de segmentación semántica del estado del arte de acuerdo a criterios que cumplan con las necesidades y limitaciones del

proyecto. Finalmente los modelos seleccionados se deben implementar utilizando un marco de trabajo de aprendizaje automático y se entrenan con el conjunto de datos preparado para finalmente evaluar mediante métricas de segmentación el desempeño de los métodos con datos reales.

La estructura del documento incluye en el capítulo 2 el análisis del estado del arte en el área de propuestas recientes de segmentación semántica y conceptos fundamentales de esta investigación. El capítulo 3 detalla la estrategia desarrollada y los algoritmos implementados en la solución propuesta. En el capítulo 4 se presentan y analizan los resultados obtenidos de los modelos aplicados. Por último, en el capítulo 5 se exponen las conclusiones derivadas del desarrollo del proyecto y recomendaciones de trabajos futuros.

Capítulo 2

Marco teórico

2.1. Segmentación de imágenes

El fin de la segmentación de una imagen consiste en identificar y clasificar los objetos contenidos en ella.

2.1.1. Tipos de segmentación

Para segmentar imágenes se utilizan métodos como la detección de objetos, la segmentación semántica y la segmentación de instancia, según el nivel de detalle.

La detección de objetos localiza los objetos de una imagen mediante un recuadro delimitador alrededor del objeto detectado [21]. El cuadro se compone de las coordenadas donde se encuentra el objeto en la imagen. A dicho cuadro corresponde la etiqueta de la clase correspondiente y el porcentaje de error.

Una alternativa de la detección de objetos es la segmentación semántica, la cual asocia una categoría a cada píxel de la imagen [10]. Como resultado, la imagen se particiona en regiones correspondientes a cada clase identificable en la segmentación. La segmentación se visualiza pintando los pixeles correspondientes a un objeto, con un color predefinido para la clase.

La segmentación de instancia en cambio, asigna una etiqueta independiente a cada objeto de la imagen aunque pertenezcan a una misma categoría [22]. Así, el resultado de la segmentación de instancia se diferencia de la segmentación semántica, en que cada región de la imagen tiene la clase de objeto a la que pertenece, pero además tiene un identificador para cada ente (o instancia) de la clase detectado. En la visualización de la segmentación de instancia se usa por lo general un color predefinido para cada instancia de cada clase.

En la figura 2.1 se visualiza el resultado de los tres métodos mencionados.

(a) Detección de objetos **(b)** Segmentación semántica **(c)** Segmentación de instancia

Figura 2.1: Tipos de segmentación de imágenes. Fuente: [23]

2.1.2. Segmentación semántica

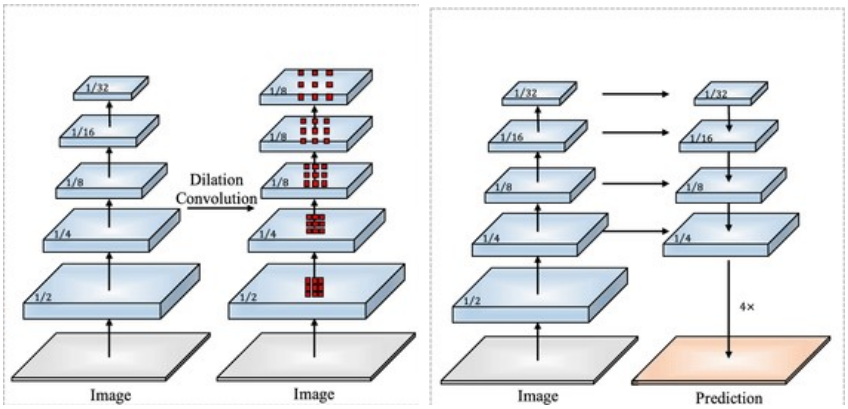
Como se definió anteriormente, el objetivo de la segmentación semántica es asignar una etiqueta semántica a cada uno de los píxeles en una imagen [24], [25]. Dichas etiquetas se visualizan con una máscara de colores [26].

Para realizar un método de segmentación semántica se necesita un conjunto de imágenes con la salida esperada, correspondiente a una máscara de segmentación o verdad de referencia (*groundtruth*) para cada una de las imágenes de entrada. La referencia es una imagen [27] con las anotaciones en cada píxel de la clase a la que pertenece, por ejemplo, "nematodo" o "fondo".

La mayoría de métodos de segmentación están basados en redes completamente convolucionales (FCN), cuyas capas convolucionales son capaces de extraer características a nivel de píxel. Sin embargo, en los últimos años, se ha comenzado a probar métodos basados en *Vision Transformers*, los cuales tienen módulos de autoatención (*self-atttention*) con resultados similares o sobresalientes al uso de redes convolucionales, sin embargo requieren una mayor capacidad computacional [25], [28], [29]. La precisión del método de segmentación depende de la red central o columna que el método utilice [30], la cual es la principal responsable de la extracción de características. Existen dos arquitecturas principales para llevar a cabo la segmentación: columna de dilatación y columna codificador/decodificador, siendo está última la más utilizada. Dichas arquitecturas se visualizan en la figura 2.2

En la arquitectura de dilatación se eliminan las operaciones de reducción y aumentan los filtros de convolución para mantener las características de alta resolución, como se muestra en la figura 2.2a. En esta arquitectura las convoluciones de dilatación toman mucho tiempo, además la eliminación del submuestreo implica mayor complejidad de cálculo y consumo de memoria [30].

La arquitectura codificador/decodificador se divide en dos partes: primeramente la codificación (*encoder*) basada en redes de clasificación [31] se encarga de aprender y extraer las características semánticas de la imagen de entrada, mediante la reducción de dimensiones (submuestreo) o un transformador [24], [28]. La etapa de decodificación (*encoder*), es la encargada de interpolar los mapas de características obtenidos en el proceso de codificación y reconstruir gradualmente la imagen en un mapa de segmentación [24], [26]. Dicha estructura se muestra en la figura 2.2b.



(a) Dilation Backbone **(b)** Encoder-Decoder Backbone

Figura 2.2: Arquitecturas de los *backbones*. Fuente: [30]

2.1.3. U-Net

La *U-Net* es un modelo de aprendizaje profundo aplicado en segmentación, caracterizado por su estructura en forma de U como se visualiza en la figura 2.3. Está conformado por dos procesos: primeramente se realiza un submuestreo para luego aplicar un sobremuestreo [32]. El submuestreo lo realizan capas convolucionales y de agrupación, capaces de disminuir el tamaño de la imagen y extraer las características de carácter semántico. El sobremuestreo al contrario, reconstruye y decodifica las características formando el mapa de segmentación, el cual consiste en la clasificación de cada píxel de la imagen según la clase a la que pertenezca. Además, *U-Net* usa la conexión de salto, la cual conecta capas paralelas del submuestreo y sobremuestreo para evitar la pérdida de datos que se genera en la reducción al eliminar la información espacial imposible de recuperar del vector de salida del submuestreo [33].

El modelo *U-Net* es actualmente uno de los más utilizados para la segmentación en imágenes de microscopía. Este modelo es robusto y capaz de entrenarse con un conjunto de datos pequeño y obtener altos valores de precisión [32]. Al día de hoy, *U-Net* es la arquitectura base que se ha utilizado para llevar a cabo una segmentación. Nuevos modelos de segmentación semántica basados en la arquitectura *U-Net* han innovado este campo.

2.2. OpenMMLab

OpenMMLab es una plataforma de código abierto de algoritmos de visión por computadora. Este proyecto provee una variedad de herramientas en las áreas de detección de objetos, clasificación de imágenes, segmentación semántica, entre otros. Este proyecto permite tanto a industrias como a entidades académicas el acceso a modelos innovadores y recientes de visión por computadora, ofreciendo reproducibilidad y bibliotecas para el manejo de los modelos disponibles.

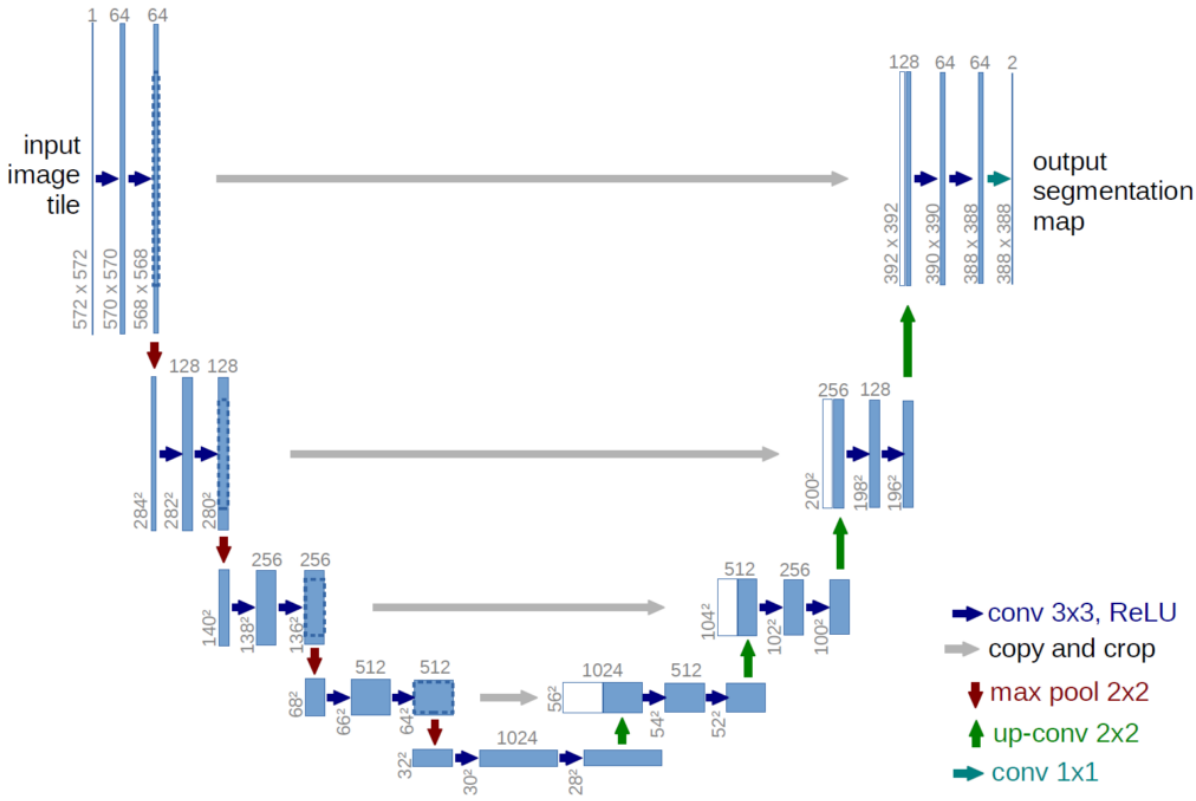


Figura 2.3: Arquitectura del modelo U-Net. Fuente:[34]

2.2.1. MMSegmentation

La herramienta *MMSegmentation* [35] proporcionada por el proyecto OpenMMLab permite la configuración de modelos de segmentación semántica. Esta herramienta basada en PyTorch, provee 34 modelos de segmentación semántica propuestos desde el año 2015 al 2022. Además, *MMSegmentation* descompone el marco de trabajo de la segmentación semántica en etapas, permitiendo a los usuarios construir modelos al combinar módulos como combinar pares de codificación/decodificación disponibles o elegir una de las trece columnas soportadas [36].

Esta herramienta dispone de un catálogo de técnicas de entrenamiento, permitiendo adaptarse a las necesidades del usuario. *MMSegmentation* soporta paralelización distribuida de datos, lo que beneficia a la velocidad de entrenamiento en comparación a otras plataformas [36]. Esta herramienta provee 34 de modelos para utilizar, y soporta conjuntos de datos como *Cityscapes, Coco Stuff, Pascal Context*, ADE20K, entre otros. Brinda además la capacidad de disponer de conjuntos de datos propios del usuario.

Entre los modelos disponibles más recientes se encuentra el ISANet [29], OCRNet [37], DNLNet [38], PointRend [39], CGNet [40], BiSeNetv2 [30], STDC [41], SETR [28], DPT [31], Segmenter [42], Segformer [42] y el K-Net [43].

En la tabla 2.1 se destacan los criterios de comparación que comprenden la capacidad computacional del modelo, el *backbone* utilizado en cada modelo, el decodificador utiliza-

do, los conjuntos de datos utilizados con sus respectivos resultados de la métrica mIoU, los parámetros de la red y las operaciones de coma flotante por segundo (GFLOPS).

Se destacan en la tabla 2.1 los cuatro *transformers* y las cuatro redes convolucionales de mejor rendimiento tomando en cuenta el número de parámetros del modelo, el número de operaciones en punto flotante requerido para una de segmentación y la métrica mIoU resultante, para el conjunto de datos correspondiente. En el apéndice A se encuentra una tabla con los 11 modelos investigados.

Tabla 2.1: Modelos de segmentación semántica disponibles en *MMSegmentation*. Los datos marcados con * son obtenidos de un *script* experimental de la herramienta. Fuente: [28], [30], [31], [37], [39], [41], [42], [44]

Modelo	Capacidad computacional	Backbone	Decoder	Datasets usados	mIoU	Params [M]	GFlops
Segformer	8 Tesla V100	Mix Transfomer MiT-B0	Lightweight All-MLP	ADE20K	37.4	3.8	8.4
				Cityscapes	76.2		125.5
Segformer	8 Tesla V100	Mix Transfomer MiT-B5	Lightweight All-MLP	ADE20K	51	84.7	183.3
				Cityscapes	82.4		1460.4
Segmenter	V100 GPU	MiT with patch tokens: Tiny	Mask Transformer (DETR)	ADE20K	39.03	6	-
Segmenter	V100 GPU	MiT with patch tokens: Large	Mask Transformer (DETR)	ADE20K	51.3	307	-
DPT	Nvidia RTX 2080 GPU	ViT bag-of-words	Three-stage Reassemble	ADE20K	49.02	343	-
				Pascal Context	60.46		
SETR	-	ViT with image sequentialization	MLA	ADE20K	48.64	310.6	-
				Pascal Context	54.87		
				Cityscapes	-		
STDC1	NVIDIA GTX 1080Ti GPU	U-net architecture with STDC module	Detail Guidance for low-level layers	Cityscapes	75.3	8.44	813 [M]
				CamVid	73		
STDC2	NVIDIA GTX 1080Ti GPU	U-net architecture with STDC module	Detail Guidance for low-level layers	Cityscapes	76.8	12.47	1446 [M]
				CamVid	73.9		
BiSeNetV2	NVIDIA GeForce GTX 1080Ti card	BiSeNetV2	-	Cityscapes	73.4	14.8*	21.15
				CamVid	72.4		
				COCO-Stuff	87.9		
PointRend	-	ResNet-101 with a PointRend module	-	Cityscapes	79.97	47.71*	521.69*
OCRNet	P40 GPU	HRNet-W48 with OCR module	Cross-attention module	Cityscapes	82	10.5	340
				ADE20K	45.97		
				Pascal Context	56.2		

2.3. Métricas de evaluación

2.3.1. Intersección sobre Unión (IoU)

La intersección sobre unión o índice de Jaccard es la métrica mayormente utilizada para comparar la similitud de dos formas. Es por ello, que se utiliza para evaluar resultados de modelos de visión por computadora como detección de objetos o segmentación. El IoU es invariante a la escala de las formas, ya que consiste en un cálculo normalizado centrado

en las áreas o volúmenes de ambas formas [45]. El valor del IoU está siempre entre los valores de 0 y 1, siendo 0 que las formas no se intersecan y 1 que se intersecan en su totalidad.

Dicha métrica se obtiene a través de la unión y la intersección de los objetos a comparar:

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

En el caso de la segmentación semántica se obtiene la IoU para los nematodos y la IoU para el fondo, obtenidos estos dos se obtiene una media que indica el porcentaje de similitud de la referencia con la predicción obtenida del modelo, representada como mIoU.

2.3.2. Entropía cruzada

La entropía cruzada es una función de pérdida utilizada en redes neuronales. Esta se encarga de generar un vector de salida como una distribución de probabilidad para todas las clases [46], [47]. Se calcula de la forma:

$$EC = -\frac{1}{n} \cdot \sum_{i=1}^{n} [Y_i \cdot \ln(Pred_i) + (1 - Y_i) \cdot \ln(1 - Pred_i)] \tag{2.2}$$

donde EC es función de pérdida de la entropía cruzada, n es el número de muestras totales, Y es el vector de valores reales y $Pred$ es el vector de valores de las predicciones [46].

2.3.3. Entropía cruzada binaria

La entropía cruzada binaria es una función de pérdida utilizada en clasificaciones binarias y múltiples. Esta se encarga de generar un vector de salida como una distribución de probabilidad binaria para cada una de las clases [46], [47]. Se calcula de la forma:

$$ECB = -\frac{1}{n} \cdot \sum_{i=1}^{n} [Y_i \cdot \log(Pred_{ij}) + (1 - Y_i) \cdot \log(1 - Pred_i)] \tag{2.3}$$

donde ECB es función de pérdida de la entropía cruzada binaria, n es el número de muestras totales, Y es el vector de valores reales y $Pred$ es el vector de valores de las predicciones [46].

2.3.4. Frente de Pareto

Con el frente de Pareto se busca la optimización multiobjetivo. Esta optimización no se restringe a una única solución, sino que se da un conjunto de soluciones no dominadas. Cada solución se representa en un espacio de valores, donde las soluciones no dominadas conforman el frente de Pareto [48], como se visualiza en la figura 2.4.

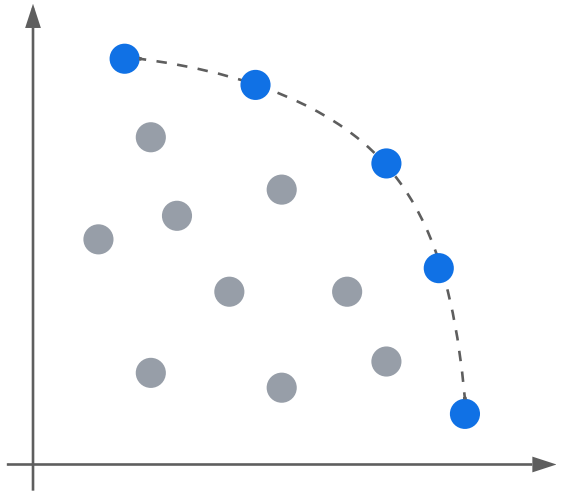


Figura 2.4: Frente de Pareto. Los puntos azules corresponden a las soluciones no dominadas.

Las soluciones no dominadas son aquellas que contenga valores aceptables para todas las funciones objetivo, por ejemplo la precisión y exhaustividad.

La precisión y exhaustividad corresponden a métricas formuladas con los valores obtenidos de la matriz de confusión. Dicha matriz es una herramienta para visualizar el desempeño del algoritmo de una red. Permite ver los aciertos y errores con los valores VP, FP, VN y FN [49].

- Verdadero Positivo (VP): El valor real es positivo y la predicción es positiva.
- Falso Positivo (FP): El valor real es negativo y la predicción es positiva.
- Verdadero Negativo (VN): El valor real es negativo y la predicción es negativa.
- Falso Negativo (FN): El valor real es positivo y la predicción es negativa.

La precisión es una referencia a la dispersión dede los valores obtenidos, cuanta menor sea la dispersión mayor será la precisión del modelo [49]. La precisión se calcula de forma:

$$P = \frac{VP}{VP + FP} \tag{2.4}$$

La exhaustividad es una referencia a la proporción de casos positivos predichos correctamente en el conjunto de datos [49]. La exhaustividad se calcula de forma:

$$R = \frac{VP}{VP + FN} \tag{2.5}$$

2.4. Métodos de segmentación semántica

2.4.1. STDC: Short-Term Dense Concatenate network

En [41] se propone una columna (*backbone*) eficiente y ligera que proporciona un campo receptivo escalable, sumado a un decodificador con una ruta única diseñada con una guía detallada de información para el aprendizaje de los detalles de bajo nivel. El objetivo de esta red es conseguir una mayor velocidad de predicción y un rendimiento competitivo a los otros métodos del estado del arte disponibles. La red se basa en la arquitectura de la red *U-net* a la cual se le concatena un módulo STDC, denominado módulo de concatenación densa a corto plazo.

La arquitectura del modelo STDC [41] está diseñada para reducir gradualmente la dimensión de los mapas de características y utilizar la agregación de los mapas para la representación de la imagen. Esta es una estructura novedosa y eficiente, ya que elimina redundancia. Dicha arquitectura se visualiza en la figura 2.5(a). Consiste de 6 etapas, exceptuando la capa de entrada y la de predicción. Las etapas de 1-5 se encargan de reducir la resolución espacial de la entrada con una apertura (*stride*) de 2, en el caso del modelo STDC2. La etapa 6 es la encargada de generar la predicción. Las etapas 1 y 2 son consideradas capas de bajo nivel en la extracción de características.

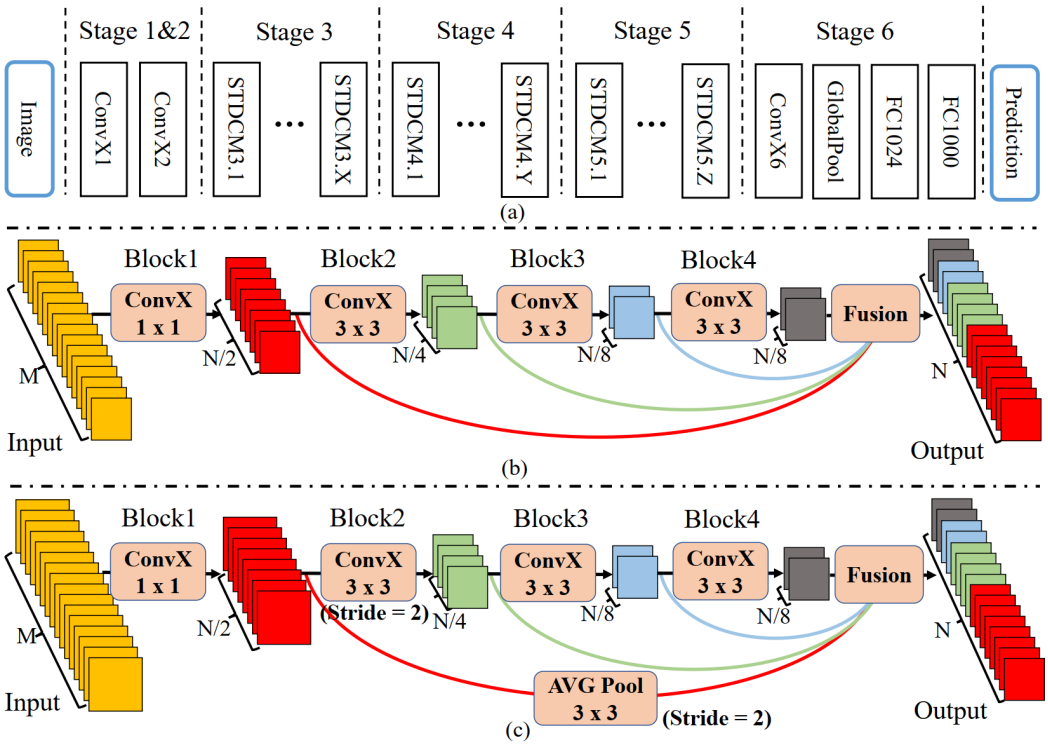


Figura 2.5: Arquitectura general del modelo y los módulos STDC. Fuente: [41]

El diseño del módulo STDC se representa en la figura 2.5, donde (b) y (c) consisten en el diseño del módulo STDC1 y STDC2 respectivamente. Cada módulo está separado en bloques y para denotar las operaciones del i-ésimo bloque se utiliza $ConvX_i$ y el cálculo

de la salida del bloque se realiza con

$$x_i = ConvX_i(x_{i-1}, k_i) \tag{2.6}$$

donde x_{i-1} denota la entrada, x_i denota la salida y k_i denota tamaño del kernel de la capa convolucional.

En el módulo STDC [41] el bloque 1 tiene un kernel de 1 a diferencia del resto de los bloques, cuyo tamaño de kernel es de 3. Dado N, el número de canales de salida del módulo STDC, se define el número de filtros de la capa convolucional del i-esimo bloque como $N/2^i$, exceptuando el filtro de la última capa convolucional, que se define como el mismo de la capa anterior. Para reducir la capacidad computacional, el tamaño de los bloques del filtro se reduce gradualmente en forma de progresión geométrica.

El proceso de submuestreo se lleva a cabo solamente en el bloque 2, y para enriquecer la información de las características de una imagen se concatenan los mapas de características de la entrada x_1 hasta la x_n de todos los bloques, como la salida del módulo STDC, obteniendo asimismo una representación de características multi-escala.

En [41] se propone un decodificador con un módulo de agregación de detalles (*Detail Aggregation*), cuyo objetivo principal es adoptar una guía de detalles (*Detail Guidance*) para las capas de bajo nivel en el aprendizaje de la información espacial de una manera única, en vez de una ruta adicional como en el modelo BiSeNet [50] que consume tiempo extra. Primeramente el módulo *Detail Aggregation* genera un mapa de referencias detallado a partir de las referencias dadas en el set de entrenamiento. Seguidamente se emplean la pérdidas *binary cross entropy* y *dice* para optimizar el aprendizaje de la información detallada, esto solamente en el proceso de entrenamiento. Por último, la información espacial de las capas de bajo nivel y la información semántica de las capas profundas se fusionan, esta función es utilizada para predecir los resultados de la segmentación.

La generación de los los mapas de referencias detallados se da mediante la aplicación de un operador Laplaciano a las referencias, y se inserta un encabezado de detalles (*Detail Head*) en la etapa 3 para generar el mapa de características detalladas. Seguidamente se utiliza el el mapa de referencias detallado como guía para que las capas de bajo nivel aprendan las características de los detalles espaciales. Por último, se lleva a cabo la fusión que permite predecir los resultados.

El operador Laplaciano consiste de un kernel convolucional en 2D (kernel Laplaciano) visualizado en la figura 2.6. Este operador produce mapas de características de detalles finos con tres aperturas (stride) para obtener una representación de características multi-escala. Seguidamente se lleva a cabo un aumento de dimensiones del mapa de características detallado a su tamaño original y se fusiona con una convolución 1×1 para una distribución de pesos dinámica. Finalmente se adopta un umbral 0.1 para convertir los detalles predichos en un referencia binaria detallada final, que contiene información de los límites y esquinas de los objetos en la imagen.

En [41] aseguran que el modelo STDC tiene dos ventajas: primero, el tamaño de los filtros de los bloques se disminuyen gradualmente en forma de progresión geométrica, lo

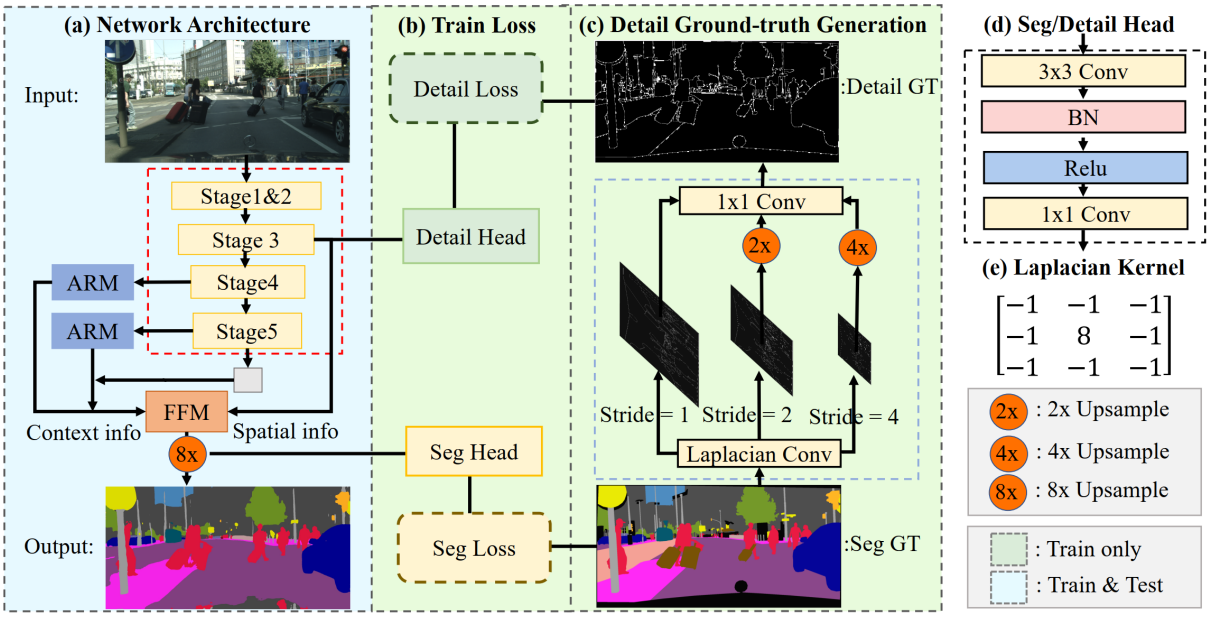


Figura 2.6: Estructura de la red de segmentación STDC. Fuente: [41]

que reduce significativamente la complejidad de cálculo del modelo. Segundo, la salida del módulo STDC es una concatenación de todos los bloques, lo que preserva los campos respectivos escalables y la información multi-escala.

Uno de los parámetros de mayor peso en la complejidad de cálculo medida en FLOPS, es la cantidad de bloques en el módulo STDC. Mediante un análisis realizado en [41] se obtiene que la cantidad de bloques que brinda un mejor rendimiento de la red, es 4 bloques. Una mayor cantidad de bloques no genera beneficio significativo y es contraproducente para los cálculos en paralelo y el desempeño en cuadros por segundo (FPS).

El módulo STDC ha demostrado en [41] resultados competitivos en precisión y la tasa cuadros por segundo (FPS) en tareas de clasificación de imágenes. Al usar la red STDC como una columna se logra obtener un equilibrio entre la velocidad y la precisión al usarlo en tareas de segmentación semántica en tiempo real.

2.4.2. PointRend: Point-based Rendering

En [39] se propone otro acercamiento a la segmentación de una imagen, y es tratarla como un problema de renderizado. Para adoptar esta perspectiva se agrega un módulo *PointRend*, donde se adaptan procesos clásicos de renderizado para generar mapas de segmentación de alta calidad.

El módulo PointRend [39] utiliza una estrategia de subdivisión. Esta estrategia se encarga de seleccionar, de forma adaptativa, un conjunto de puntos no uniforme para realizar predicciones de segmentación en dichos puntos. El módulo se compone de tres componentes: primeramente la estrategia de selección de puntos, encargada de seleccionar una pequeña cantidad de puntos de la imagen, evitando cálculos excesivos para todos los píxeles en

la imagen. El segundo componente corresponde a un extractor de una representación de características para cada punto, esto mediante la interpolación bilineal de los mapas de características obtenidos al realizar un submuestreo. Esto permite utilizar la información de subpíxeles codificada en la dimensión de la capa del mapa de características y predecir un mapa de segmentación de mayor resolución. Por último, una subred neuronal llamada *Point Head* predice las etiquetas de salida a partir de la representación de características para cada punto utilizando un MLP (*Multilayer Perceptron*) que distribuye los pesos a lo largo de todos los puntos.

Para un mejor rendimiento, los puntos seleccionados se distribuyen prefiriendo áreas de alta frecuencia como los bordes de los objetos. Sin embargo, la selección de estos puntos difiere en los procesos de inferencia y de entrenamiento. Para la etapa de inferencia, los autores proponen utilizar una técnica de subdivisión adaptativa mostrada en la figura 2.7, la cuál consiste en calcular la representación de características solamente de los puntos donde haya alta probabilidad que su valor sea diferente a puntos vecinos. En el caso del resto de puntos, los valores de predicción se obtienen de la interpolación de los valores previamente calculados. En cada iteración se aumenta el tamaño de la imagen y se seleccionan los N puntos más inciertos, siendo N un hiperparámetro del modelo. Se finaliza el proceso iterativo al obtener una predicción del tamaño de la imagen de entrada.

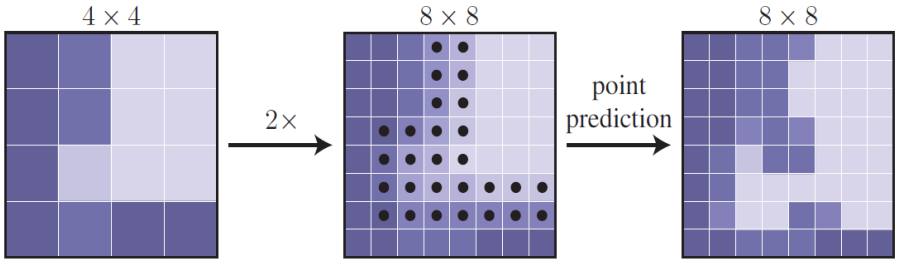


Figura 2.7: Pasos de la subdivisión adaptativa. Fuente: [39]

En la etapa de entrenamiento se utiliza una estrategia no iterativa basada en muestreo aleatorio. Aquí, la selección de puntos se sesga hacia las regiones inciertas, manteniendo al mismo tiempo un grado de cobertura uniforme al utilizar los principios de sobre generación, muestreo de importancia y cobertura. En el proceso de sobre-generación, se generan puntos candidatos para el conjunto al muestrear aleatoriamente kN puntos de una distribución uniforme. El muestreo de importancia se centra en los puntos con predicciones inciertas, interpolando las predicciones en todos los kN puntos y calculando una estimación de la incertidumbre. La cobertura comprende el muestreo de los puntos restantes $(1 - \beta)N$ a partir de una distribución uniforme. Para un mejor rendimiento en el entrenamiento, se muestrea de manera sesgada solamente una pequeña cantidad de puntos por región, con un $N = 14^2$ $k = 3$ y $\beta = 0{,}75$. Para ello los autores realizaron una comparación de varias configuraciones como se observa en la figura 2.8.

Cada estrategia de predicción y entrenamiento, tiene como hiperparámetro un valor particular de N dado por el usuario. En el caso de tener objetos con bordes complejos utilizar una mayor cantidad de puntos puede ser beneficioso. Tras las pruebas realizadas en [39] se

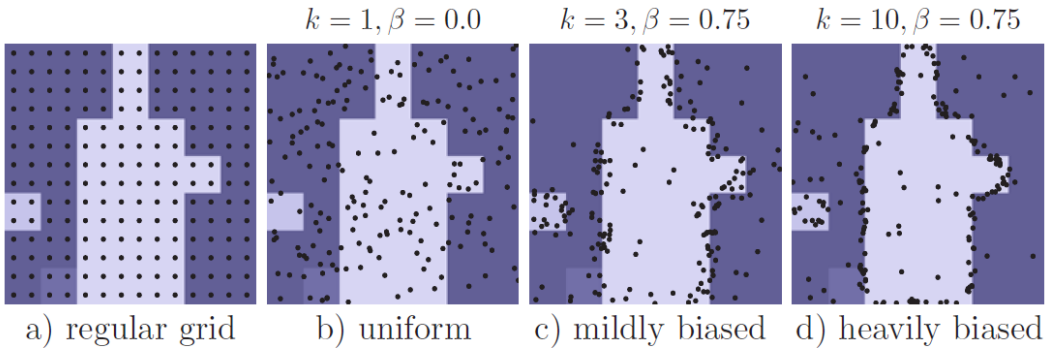


Figura 2.8: Comparación del muestreo de puntos en el proceso de entrenamiento. Fuente: [39]

estima que con $N = 14^2$ se obtiene un entrenamiento eficiente en cálculos y memoria. El uso de mayor cantidad de puntos no mejora significativamente los resultados. Este método *PointRend* se puede aplicar tanto a segmentación semántica como a segmentación de instancia.

2.4.3. Segformer

Segformer [44] es una arquitectura que combina transformadores *(Transformers)* con decodificadores MLP ligeros, que conducen a una segmentación eficiente y robusta. A partir de esta arquitectura se establecen seis modelos que van desde Segformer-B0 hasta el Segformer-B5, con un rendimiento y eficiencia significativamente ascendente en los modelos.

La entrada de esta arquitectura consiste en una imagen de tamaño $H \times W \times 3$, la cuál se divide en fragmentos de 4×4 para favorecer la tarea de predicción densa. Estos fragmentos se utilizan como entradas del codificador *transformer* jerárquico para generar características multinivel, específicamente a los niveles $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ del tamaño de la imagen original. Estas características se pasan al módulo decodificador ligero *All-MPL* que fusiona las características multinivel del codificador para crear las máscaras de segmentación de salida a un tamaño de $H \times W \times N_{cls}$, con N_{cls} como la cantidad de clases. En la figura 2.9 se visualiza la arquitectura de este modelo.

El codificador del modelo Segformer esta basado en un *Mix Transformer* (MiT) donde los modelos del MiT-B0 al MiT-B5 tiene la misma arquitectura con la diferencia de ser de tamaños diferentes. Esto convierte a MiT-B0 como el modelo más ligero para una inferencia rápida, y el modelo MiT-B5 como el modelo más grande para un mejor rendimiento.

La capa de autoatención (*self-attention*), estimada con

$$\text{Attention}(Q,K,V) = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_{head}}}\right)V \tag{2.7}$$

es la que más ralentiza el cálculo computacional. En el proceso original Q, K y V tienen las mismas dimensiones $N \times C$, donde $N = H \times W$ es la longitud de la secuencia. Este

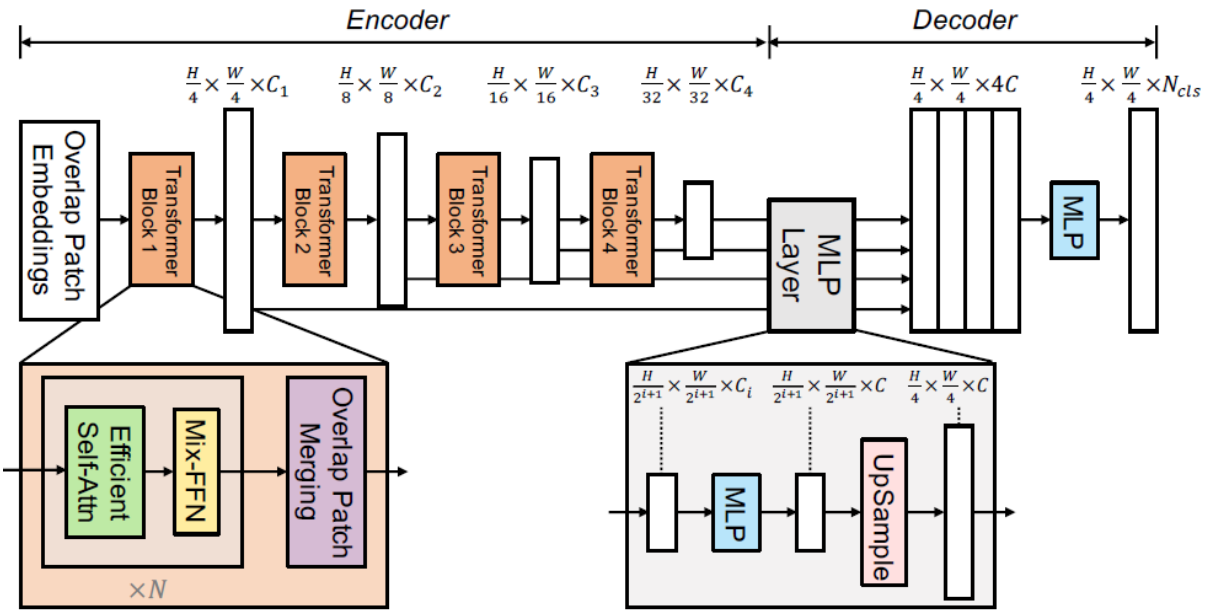


Figura 2.9: Arquitectura del modelo Segformer. Fuente: [44]

proceso tiene una complejidad computacional de $O(N^2)$ haciendo inasequible el uso de imágenes de alta resolución.

Para contrarrestar la complejidad de esta capa, se utiliza un proceso de reducción de secuencias introducido en [51], el cual utiliza la reducción en una tasa R para reducir la longitud de la secuencia de modo

$$\hat{K} = \text{Reshape}\left(\frac{N}{R}, C \cdot R\right)(K) \tag{2.8}$$

$$K = \text{Linear}(C \cdot R, C)(\hat{K}) \tag{2.9}$$

donde $\text{Reshape}(\frac{N}{R}, C \cdot R)(K)$ significa rehacer el tamaño de K al tamaño $\frac{N}{R} \times (C \cdot R)$, y $\text{Linear}(C_{in}, C_{out})(\cdot)$ corresponde a una capa lineal que toma un tensor de C_{in} dimensiones como entrada y genera un tensor de C_{out} dimensiones de salida. Utilizando este proceso la complejidad de la capa de autoatención se reduce de $O(N^2)$ a $O(\frac{N^2}{R})$.

Los modelos que aplican en su codificador un tranformador para visión (*vision transformer* (ViT)) usan codificación posicional, PE en sus siglas en inglés, para introducir la información de localización con un tamaño fijo. Cuando el tamaño en la inferencia y en el entrenamiento difieren se produce una caída de la precisión. En Segformer se utiliza un módulo llamado *Mix-FNN* que filtra la información de localización con el efecto de relleno cero, esto utilizando una capa convolucional 3×3 que evita la pérdida de precisión con

$$x_{out} = \text{MLP}(\text{GELU}(Conv_{3\times3}(\text{MLP}(x_{in})))) + x_{in} \tag{2.10}$$

donde MLP es el *Multilayer Perceptron*, GELU es la función de activación *Gaussian Error Linear Units* y x_{in} es la característica de salida del módulo de autoatención.

El decodificador del modelo Segformer consiste solamente de capas de un MLP ligero que aprovecha las características originadas del *transformer*, donde las capas de baja dimensión mantienen una atención local y las capas de mayor dimensión mantienen una atención global. El MLP agrega información de varias capas y combina la atención local y global para obtener representaciones potentes de un decodificador.

El funcionamiento del decodificador tiene cuatro pasos. Primeramente, las características multinivel F_i del codificador MiT pasan por una capa MLP unificando la dimensión del canal. Segundo, las características se sobremuestrean de 1/8 de dimensión a 1/4 y se concatenan. Tercero, se utiliza una capa para fusionar las características F. Por último, otra capa toma las características fusionadas para inferir la máscara de segmentación resultante M con una resolución $\frac{H}{4} \times \frac{W}{4} \times N_{cls}$. Las ecuaciones

$$\hat{F}_i = \text{Linear}(C_i,C)(F_i),\forall i \tag{2.11}$$

$$\hat{F}_i = \text{Upsample}\left(\frac{W}{4} \times \frac{W}{4}\right)(\hat{F}_i),\forall i \tag{2.12}$$

$$F = \text{Linear}(4C,C)(Concat(\hat{F}_i)),\forall i \tag{2.13}$$

$$M = \text{Linear}(C,N_{cls})(F), \tag{2.14}$$

resumen los pasos anteriores, donde $\text{Linear}(Cin; Cout)(\cdot)$ se refiere a una capa lineal con C_{in} y C_{out} como vectores.

En conclusión, el modelo más ligero correspondiente a Segformer-B0 demuestra ser un modelo compacto, aún así eficiente y con un rendimiento competitivo. Este método resulta eficiente en aplicaciones en tiempo real. Por otro lado, el modelo más grande Segformer-B5, obtiene resultados sobresalientes en el estado del arte en los conjuntos de datos probados en [44], mostrando el potencial de la arquitectura así como que al incrementar el tamaño del codificador se obtienen mejoras constantes.

En cuanto al rendimiento del decodificador, los autores analizan la influencia de la dimensión del canal, el cual configurado en C=256 provee un rendimiento y un costo computacional competitivo. Si se aumenta la dimensión del canal el modelo se vuelve más grande y menos eficiente. Sin embargo, con dimensiones de canal superiores a 768 el rendimiento se estabiliza, por lo que modelos de Segfromer-B0 al Segfromer-B0 utilizan $C = 256$ y el resto $C = 768$.

2.5. Unidad de procesamiento gráfico (GPU)

Una unidad de procesamiento gráfico o GPU, por sus siglas en inglés, es un chip de silicio que contiene millones de transistores diseñado inicialmente para la computación intensiva y paralela, requerida en la generación de gráficos por computador, pero que se aprovecha en la actualidad para realizar procesamiento de propósito general. Dicha unidad trabaja en conjunto con la unidad central de procesamiento (CPU), quien controla el flujo y almacenamiento de los datos [52].

Las tarjetas GPU están diseñadas para entrenamientos e inferencias de modelos de aprendizaje profundo, analítica de datos, la computación científica, la genómica, renderizado de gráficos, entre otras [53]. Debido a la demanda de capacidad computacional que supone el uso de redes neuronales de aprendizaje profundo para la tarea de segmentación semántica, tomando en cuenta las redes convolucionales totalmente conectadas (FCN) y *transformers*, se utilizan varias GPU.

2.5.1. NVIDIA Tesla K80

La GPU NVIDIA Tesla K80 es un módulo de computación con un acelerador dual GPU basado en la arquitectura Kepler; es decir, posee dos tarjetas GPU Tesla GK210 incorporadas. Dicho acelerador está diseñado para servidores, con una frecuencia base de trabajo de 562 MHz y 12 GB de memoria GDDR5 por GPU formando un total de 24 GB, con un ancho de banda de memoria de 240 GB/s por GPU [54]. La tarjeta K80 acelera las cargas de trabajo debido al poder de cómputo de simple y doble precisión que corresponde a 8.74 TFLOPS y 2.91 TFLOPS respectivamente [55].

La tarjeta K80 fue lanzada en el año 2014 y cuenta con interconectores PCI *Express* 3.0 de doble ranura con formato Tesla. Además, incorpora código de corrección de errores (ECC), el cual está activado por defecto con el fin de proteger los archivos de registro, la caché y la DRAM. Por último, la K80 sólo está disponible con un disipador térmico pasivo, que requiere un flujo de aire externo para su refrigeración [55], [56].

En la tabla 2.2 se resumen las características principales de dicha tarjeta.

Tabla 2.2: Especificaciones generales de la GPU NVIDIA Tesla K80. Fuente: [55], [56]

Modelo	Tesla K80
Número de GPU	2 Tesla GK210
Arquitectura	Kepler
Memoria	12GB por GPU
Núcleos NVIDIA CUDA	2.496 por cada GPU
Ancho de banda de memoria	240 GB/s por GPU
Potencia máxima	300 W
Operaciones en simple precisión	8.74 TFLOPS
Operaciones en doble precisión	2.91 TFLOPS
Interfaz del sistema	PCI Express 3.0

2.5.2. NVIDIA A100 Tensor Core GPU

La tarjeta GPU NVIDIA A100 lanzada en el año 2020, tiene una versión de 40 GB y otra de 80 GB de memoria HBM2 [57]. Está basada en la arquitectura *Ampere* GA100 que ofrece un ancho de banda de 1555 GB/s. El poder de cómputo de esta tarjeta admite cálculos

de doble precisión (FP64) de 9.7 TFLOPS, de simple precisión de 19.5 TFLOPS y a media precisión de 78 TFLOPS [53], [57]. La A100 incorpora algoritmos ECC encargados de proteger la interfaz de memoria y la memoria DRAM al detectar y corregir errores simples, dobles o de bits impares. Para su conexión la A100 cuenta con PCI *Express* 4.0 de doble ranura. Además, incorpora un disipador pasivo bidireccional, el cuál requiere de un flujo de aire para que funcione dentro de los límites térmicos [57].

La targeta A100 innova con la función GPU Multi-Instancia (MIG), la cual permite la virtualización y la partición de hasta 7 instancias de la GPU NVIDIA A100. Cada partición es totalmente aislada con su propia memoria de ancho de banda, caché y núcleos de cálculo. Esta función es principalmente utilizada en la creación de servidores, clústeres y supercomputadoras para el uso de una o varias GPU [53].

En la tabla 2.3 se resumen las características principales de dicha tarjeta.

Tabla 2.3: Especificaciones generales de la GPU NVIDIA A100. Fuente: [53], [57]

Modelo	NVIDIA A100 Tensor Core
GPU	A100
Arquitectura	Ampere
Memoria	40 GB
Núcleos NVIDIA CUDA	6,912
Ancho de banda de memoria	1555 GB/s
Potencia máxima	250 W
Operaciones en media precisión	78 TFLOPS
Operaciones en simple precisión	19.5 TFLOPS
Operaciones en doble precisión	9.7 TFLOPS
Interfaz del sistema	PCI Express 4.0

2.5.3. Cluster Kabré

La supercomputadora Kabré del CeNAT es un clúster compuesto por 24 arquitecturas en paralelo llamadas nodos, Uno de ellos es el nodo de aprendizaje automático (nukwa) constituido por una GPU NVIDIA Tesla V100, 24 núcleos CPU de 2.20GHz, 2 hilos por núcleo y contienen una memoria de 32 GB de RAM.

La V100 disponible en el clúster Kabré, cuenta con intercorectores PCI 3.0 y la memoria GPU disponible es de 32GB. La tarjeta V100 lanzada en el año 2017, está diseñada bajo una arquitectura NVIDIA Volta y ofrece un ancho de banda de memoria de 900 GB/sec. Esta tarjeta está protegida por algoritmos ECC y admite cálculos de simple y doble precisión[58].

En la tabla 2.4 se resumen las características principales de dicha tarjeta.

Tabla 2.4: Especificaciones generales de la GPU NVIDIA V100. Fuente: [58]

Modelo	NVIDIA V100 Tensor Core
GPU	V100
Arquitectura	Volta
Memoria	32 GB
Núcleos NVIDIA CUDA	5,120
Ancho de banda de memoria	900 GB/s
Potencia máxima	250 W
Operaciones en simple precisión	14 TFLOPS
Operaciones en doble precisión	7 TFLOPS
Interfaz del sistema	PCI Express 3.0

2.6. Trabajos relacionados

En las imágenes de microscopía los objetos capturados se caracterizan por tener una baja resolución y variación de la señal a ruido. Para comprender las características y actividades biológicas ligadas a las imágenes de microscopía se aplican algoritmos de análisis y visión por computador [32], [59].

En consecuencia al reto que estas imágenes presentan, se ha realizado estudios aplicando algoritmos de aprendizaje profundo como la clasificación de imágenes, segmentación, seguimiento de objetos, entre otros [32], [59]. Dichos estudios han demostrado que los métodos de segmentación basados en aprendizaje profundo alcanzan mejores resultados de precisión que los métodos tradicionales. No obstante, dichos estudios se han aplicado mayormente a células, estructuras subcelulares y tejidos en las aplicaciones de conteo de células, análisis de morfometría, diagnósticos de enfermedades y el análisis de imágenes de tejidos [32]. Estas estructuras estudiadas son estáticas y de formas similares entre sí, a diferencia de los nematodos que son especies de aspecto vermiforme y dinámicos. Esta diferencia requiere de datos y etiquetas robustas y exactas para el aprendizaje de la red.

2.6.1. Segmentación de quistes causados por nematodos

En [60] se realizan estudios en imágenes de microscopía provenientes de muestras de suelo con el fin de cuantificar la infestación de los nematodos, a través de la segmentación de instancia y clasificación de los quistes de nematodos. Para abarcar dicho problema, proponen el uso de un algoritmo LMBI (*Local Maximum of Boundary Intensity*) para el proceso de segmentación de instancia, seguido de un clasificador para separar los quistes de partículas del suelo. El conjunto de datos utilizado en este caso contiene alrededor de 132 imágenes con un total de 1473 nematodos identificados del tipo *Heterodera schachtii* o nematodo de la remolacha.

La estrategia utilizada en este método [60] es generar, a partir de un proceso de crecimiento

de regiones como el ilustrado en la figura 2.10, los quistes candidatos a segmentar mediante el algoritmo LMBI, trazando la intensidad media del gradiente a lo largo del límite de la región mientras crece. Seguidamente, un clasificador SVM separa los quistes de otras regiones pertenecientes a partículas del suelo mediante colores, texturas y formas.

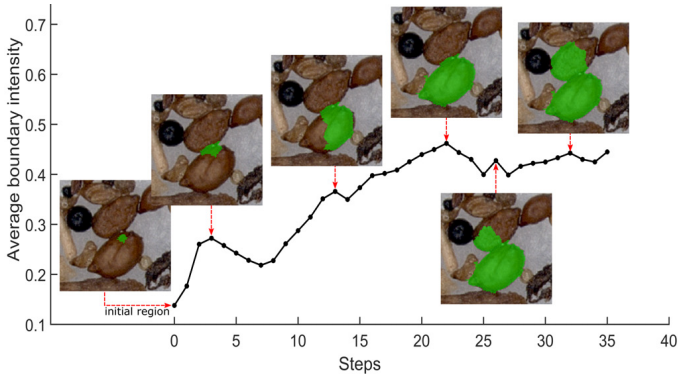


Figura 2.10: Proceso de crecimiento de regiones. Fuente: [60]

Clasificar nematodos resulta complicado debido a la semejanza de colores, texturas y formas de los componentes orgánicos con los quistes de nematodos. No obstante, el método aplicado en [60] logra obtener resultados altos de exactitud así como valores entre 70 % a 75 % del coeficiente *dice* [61].

2.6.2. Segmentación aplicada a nematodos

En [62] se propone un marco de trabajo para la detección objetos en forma de gusano en imágenes de microscopía basado en redes neuronales convolucionales (CNN). En esta detección se opta por anotaciones de líneas curvas en lugar de cuadros delimitadores, como se usan tradicionalmente en la detección de objetos.

Para evaluar el método, en [62] utilizan dos conjuntos de datos. El primer conjunto consiste de una recopilación de 3376 imágenes del nematodo dorado de la papa (*Globodera spp.*). El segundo, conjunto corresponde a un conjunto de datos público del nematodo *C. elegans* de alrededor de 200 imágenes (*Broad Bioimage Benchmark Collection:* BBBC010 [63]).

El método empleado se encarga de predecir curvas asemejando el esqueleto y puntos finales del cuerpo. Esto se logra con base a una arquitectura *U-Net* con dos ramas añadidas a la última capa para las características a predecir. Además, el modelo aplica la variante de pérdida focal para evitar que el desbalance de etiquetas de píxeles afecte el entrenamiento.

Al estimar los esqueletos de los nematodos así como los puntos finales del cuerpo, se pueden reconocer aquellos nematodos que se encuentren entrelazados. Finalmente el método [62] busca estimar el ancho del nematodo al obtener los bordes con círculos centrados en el esqueleto, finalmente con estos datos se reconstruye un mapa de segmentación de cada imagen. Realizando así una segmentación semántica a partir de una detección de objetos.

Con la aplicación de la pérdida focal, el modelo logra predecir el esqueleto y puntos finales

del nematodo de forma robusta, y logra además resultados de precisión entre el 80 % y 90 % [62].

Asimismo, en [64] se identifican nematodos utilizando redes neuronales convolucionales (CNN). Utilizando Keras [65] y tras una evaluación de 13 modelos disponibles, eligen utilizar el modelo *Xception*. En ese caso se busca con qué hiperparámetros, optimizadores y métodos de entrenamiento se logra una identificación óptima de los nematodos.

Como conjunto de datos en [64] extraen imágenes de nematodos entomopatógenos (EPN) separándolos en las etapas de vida de joven y adulto debido a la diferencia de características. Este conjunto de datos logra recopilar 422 imágenes en total. Debido a la mínima cantidad de datos, se opta por hacer un aumento de datos clásico por medio de la rotación y volteo vertical de las imágenes.

Usando el modelo *Xception*, se explora cual método de entrenamiento es el óptimo. Para ello se hacen experimentos con dos estrategias de entrenamiento: aprendizaje transferido e inicialización aleatoria. La estretegia de aprendizaje transferido se divide en dos métodos: estracción de carácterisitcas donde se continua un entrenamiento preexistente y ajuste fino (*fine-tunning*) donde se preentrena el modelo con una matriz de pesos preexistente. Con el método de entrenamiento de extracción de características se obtiene una menor precisión, sin embargo, es el método que tiene un menor nivel de clasificación errónea al presentarle imágenes donde no hay nematodos. Finalmente, este método de extracción de características logra obtener resultados de precisión en la validación de 88 % en nematodos jóvenes y 69 % en nematodos adultos [64].

Capítulo 3

Segmentación semántica en imágenes de microscopía óptica

3.1. Preparación de datos

El conjunto de datos de nematodos actual disponible del SIPLab cuenta con 7268 imágenes en formato png y tamaño 1024×768 píxeles. Estas imágenes provienen de capturas de 15 vídeos de muestras cotidianas de microscopía de nematodos. De este conjunto de datos, Taylor [19] obtiene 3856 imágenes anotadas usando hitos (*landmarks*) almacenados en un archivo en formato de texto json. Este archivo contiene la información de la región que cubre un solo nematodo en cada una de las imágenes del conjunto.

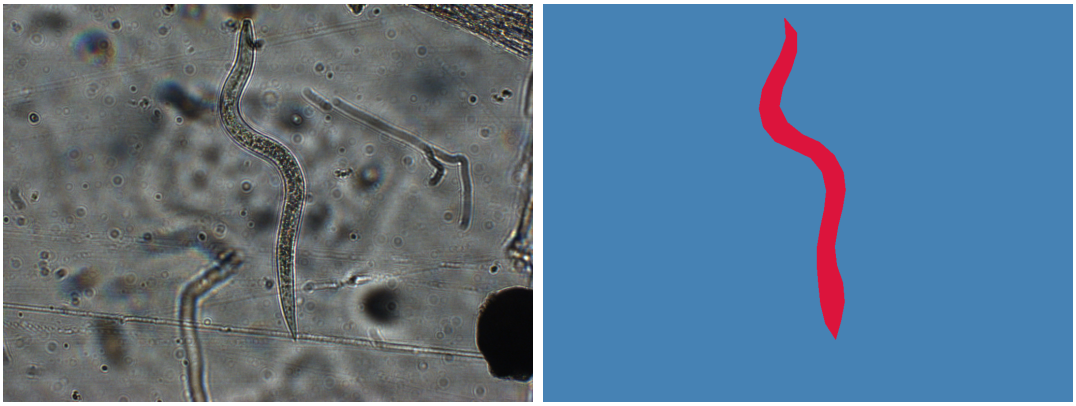
El conjunto de datos y las máscaras de segmentación se separan en tres subconjuntos: entrenamiento, validación y prueba. El subconjunto de entrenamiento es el responsable del aprendizaje de la red y comprende las tres quintas partes del conjunto. El subconjunto de validación es utilizado para obtener las métricas de evaluación y hacer la selección del mejor de los modelos; este comprende una quinta parte del conjunto. Por último, el subconjunto de prueba comprende datos no antes vistos por la red en el entrenamiento y así evaluar el error real del modelo seleccionado; este comprende una quinta parte del conjunto.

Los modelos de aprendizaje profundo se caracterizan por la gran cantidad de datos necesaria para su funcionamiento y buen rendimiento. Debido a esto y a la escasez de datos del conjunto disponible, se aplica un aumento de datos utilizando la herramienta de transformaciones brindada por el proyecto OpenMMLab. Se aplican al conjunto de datos las transformaciones clásicas como: rotación, volteado, escalamiento, recortado y distorsión fotométrica.

3.1.1. Generación de máscaras de segmentación

La máscara de segmentación o referencia (*groundtruth*) corresponde al resultado esperado de la red. Para la generación de la misma se utiliza la biblioteca *pillow*, con la cual es posible abrir, manipular y guardar imágenes. El archivo de extensión json obtenido de Taylor [19], contiene la sucesión de hitos (x,y) que delimita a cada nematodo con los que se generan las máscaras de segmentación para el conjunto de datos.

Utilizando la biblioteca *pillow* y las coordenadas del nematodo, se genera una imagen RGB donde el fondo es azul y el nematodo es rojo. Debido a que la referencia corresponde a una imagen binaria, se utiliza la biblioteca *pngquant* para indexar las imágenes, denominando los píxeles azules de la imagen con un valor de 0 y los píxeles rojos con un valor de 1, convirtiéndola así en una imagen binaria. En la figura 3.1 se presenta un ejemplo de la imagen original con su respectiva máscara.



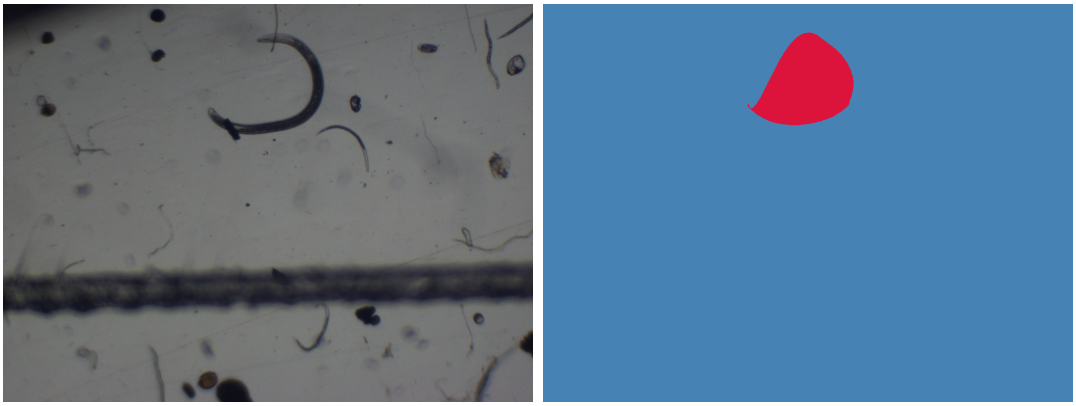
(a) Original **(b)** Máscara

Figura 3.1: Comparación de imagen original con su respectiva máscara.

Las etiquetas de referencia de una segmentación semántica requieren más trabajo manual para su elaboración, que las etiquetas para una red de detección de objetos, por ejemplo. Para una imagen de microscopía óptica de baja resolución, distinguir los bordes de los objetos se vuelve complejo [32], por lo que las etiquetas manuales suelen no ser óptimas. Por ejemplo, en la figura 3.1b se observa cómo la punta inferior del nematodo tiene un grosor mayor a la de la figura 3.1a. Por ende, se descartan 29 máscaras que presentan fallos en la delimitación de los nematodos como la de la figura 3.2.

3.2. Análisis de métodos de segmentación semántica

Para la realización de este proyecto se utilizan los modelos disponibles en la biblioteca *MMSegmentation* del proyecto *OpenMMLab* [35]. Esta biblioteca soporta alrededor de 34 modelos de segmentación, de los cuales se analizan aquellos con una antigüedad de no más de tres años. Los modelos que cumplen con este criterio se resumen en la tabla 2.1.



(a) Original **(b)** Máscara

Figura 3.2: Máscara con fallos en la delimitación del nematodo.

Para hacer una selección de los modelos a utilizar se contemplan los criterios de comparación, definidos en la tabla 2.1. Con el criterio de la columna del modelo (*backbone*), se identifica si este corresponde a una arquitectura basada en redes convolucionales o en *transformers*.

Los criterios como los parámetros (params) y los GFLOPS permiten reconocer la capacidad computacional necesaria para llevar a cabo el entrenamiento del modelo en una tarjeta GPU. Entre menor cantidad de parámetros y GFLOPS, menor capacidad computacional es necesaria.

Por último, el criterio de la métrica mIoU demuestra el rendimiento de la red respecto a un conjunto de datos. Los modelos de la biblioteca *MMSegmentation* utilizan los conjuntos datos disponibles en la misma como: *ADE20K* (20210 imágenes), *Cityscapes* (5000 imágenes), *Pascal Context* (10103 imágenes) y *CamVid* (701 imágenes) [28], [41].

3.2.1. Modelos seleccionados

Tomando en cuenta los criterios de comparación se seleccionan los tres mejores modelos para llevar a cabo una segmentación semántica con imágenes de microscopía óptica de nematodos. Para dar variedad a la comparación se toma 1 modelo diseñado con *transformers* y 2 modelos diseñados con redes convolucionales.

Se tienen disponibles tres tarjetas GPU correspondientes a los modelos K80, A100 y V100 en el clúster Kabre. La tarjeta A100 de 40 GB de memoria GPU disponible se utiliza mediante dos máquinas virtuales a las que se les destina 10 GB y 20 GB respectivamente. Las tarjetas K80, A100 de 10GB y A100 de 20 GB tienen una capacidad computacional que permite el entrenamiento de los modelos Segformer-B0, Segmenter Tiny, STDC, BiSeNetV2 y PointRend. La tarjeta V100 del clúster Kabre tiene además una capacidad computacional que permite el entrenamiento de los modelos Segformer del B1 al B4. Con el criterio del rendimiento de la red en la métrica mIoU, se seleccionan finalmente los

modelos Segformer, STDC y PointRend para la segmentación de nematodos.

3.2.2. Configuración de modelos

Los modelos de segmentación semántica disponibles en la biblioteca *MMSegmentation* están diseñados mediante *scripts* de configuración llamados *config files*. En estos archivos se especifican todos los componentes necesarios para la definición del modelo, datos y configuración del entrenamiento. La estructura de estos archivos incluye cuatro componentes: *model, dataset, schedule, default_runtime*.

El componente *model* comprende la arquitectura del modelo de segmentación. Este contiene parámetros como el tipo de segmentación, la definición de la columna y el decodificador a utilizar con sus respectivas capas y módulos complementarios, y las configuraciones de los procesos de entrenamiento e inferencia.

El *dataset* es el componente donde se establece el conjunto de datos a segmentar. Se deben especificar parámetros como la ruta de los datos, las muestras por GPU, el tamaño de la imagen y el *pipeline*. El conjunto de datos tiene un *pipeline* para el entrenamiento y para la inferencia, donde además de cargar las imágenes y máscaras se determinan las transformaciones de aumento de datos a aplicar.

La configuración de hiperparámetros como la optimización, cantidad de iteraciones, tasa de aprendizaje, las métricas de evaluación, los intervalos de evaluación y guardado del modelo se definen en el componente *schedule*. Por último, el componente de *default_runtime* especifica para el entrenamiento los *hooks*, el flujo de trabajo y el preentrenamiento en caso que se de. La biblioteca *MMSegmentation* aplica el uso de *hooks* que brindan principalmente el modo de visualización de los resultados de la evaluación, tales como *TensorBoard, Weights and Biases* (W&B) o simplemente en salida de texto.

3.3. Proceso de comparación de modelos

El proceso de comparación de modelos de segmentación semántica integra las etapas de: la preparación de los datos, donde se lleva a cabo la generación de las máscaras de segmentación y el aumento de los datos. Segundo, se lleva a cabo la implementación de los modelos STDC, PointRend y Segformer mediante los archivos de configuración con 4 estrategias de entrenamiento. Tercero, se lleva a cabo el entrenamiento de los modelos. Cuarto, se obtienen las métricas de evaluación de cada configuración de cada modelo. Por último, se realiza la comparación de las predcciones provenientes del set de validación y de prueba. Estas etapas se ilustran de manera detallada en la figura 3.3.

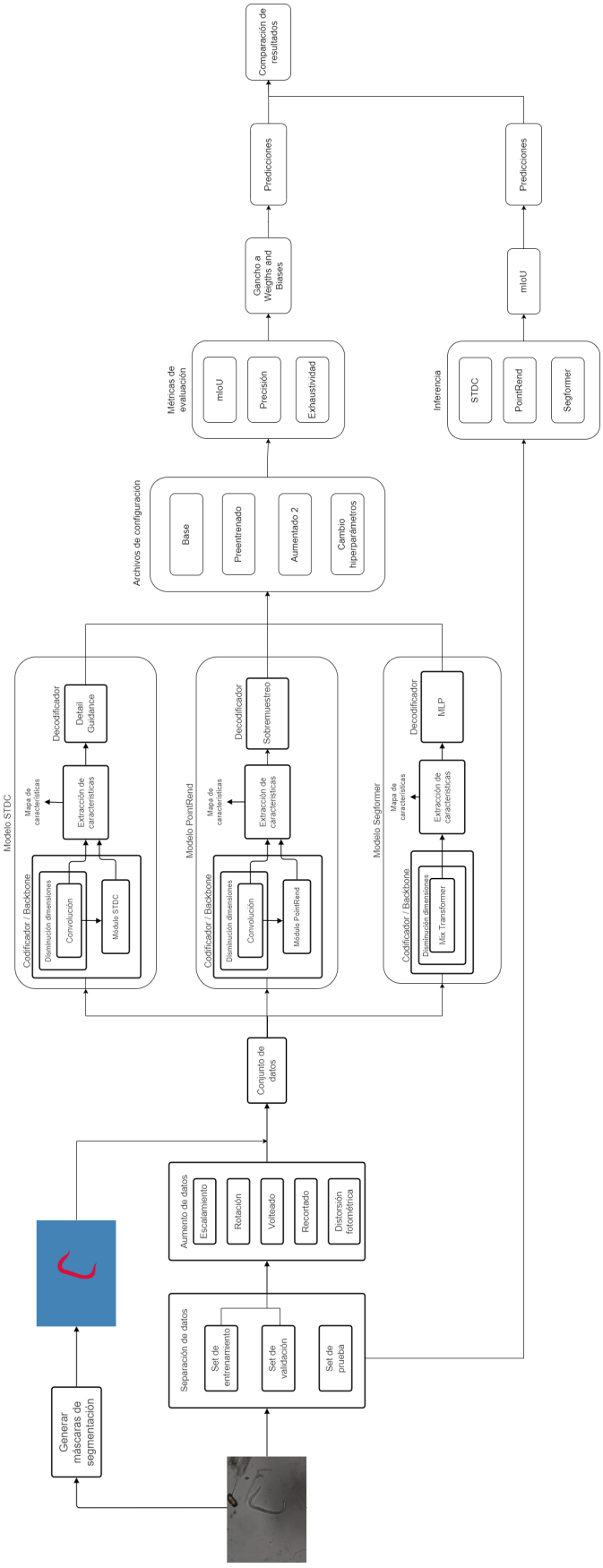


Figura 3.3: Diagrama de bloques de la solución.

Capítulo 4

Resultados y análisis

En este capítulo se presentan los resultados de las pruebas realizadas con los modelos de segmentación semántica seleccionados. El desempeño de los modelos en la tarea de segmentación semántica se mide a través de la precisión y exhaustividad del modelo, así como media de la intersección sobre unión (mIoU) entre la máscara de segmentación y la predicción del modelo. Además, se lleva a cabo una comparación para determinar los modelos óptimos en la segmentación de nematodos.

4.1. Segmentación con el modelo STDC

El modelo de segmentación STDC aporta dos configuraciones llamadas STDC1 y STDC2 que varían el paso y tiempos de repetición de las etapas del modelo. Sumado a estas configuraciones se llevan a cabo pruebas de entrenamiento base donde el modelo crea los pesos iniciales para el conjunto de datos de nematodos, así como pruebas con un modelo preentrenado de proyecto *MMSegmentation* con datos del conjunto *Cityscapes*. Se elige *Cityscapes* ya que es el único conjunto de datos disponible para el modelo STDC en *MMSegmentation*. Por último, se realiza una prueba cambiando las transformaciones del aumento de datos omitiendo el proceso de rotación de las imágenes y la distorsión fotométrica. A este caso se le da el nombre de aumentado 2. Los resultados de dichas pruebas se desglosan en la tabla 4.1.

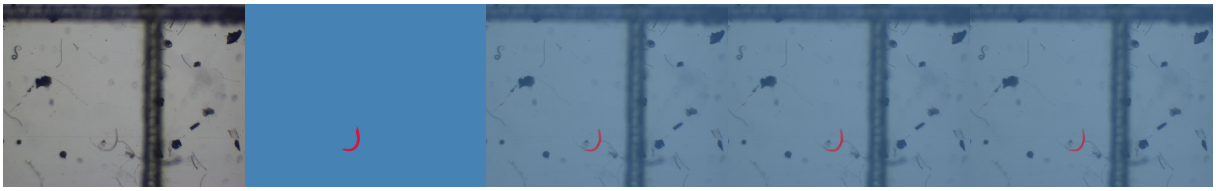
Tabla 4.1: Comparación de resultados de las configuraciones del modelo STDC.

Modelo	Preentrenado	Rotación	Params [M]	GFLOPs	mIoU val [%]	mIoU test [%]
STDC1	No	Sí	8.57	67.66	90.69	90.65
	Sí	Sí			91.33	91.27
STDC2	No	Sí	12.6	94.13	90.70	90.60
	Sí	Sí			91.37	91.23
	Sí	No			91.01	91.00

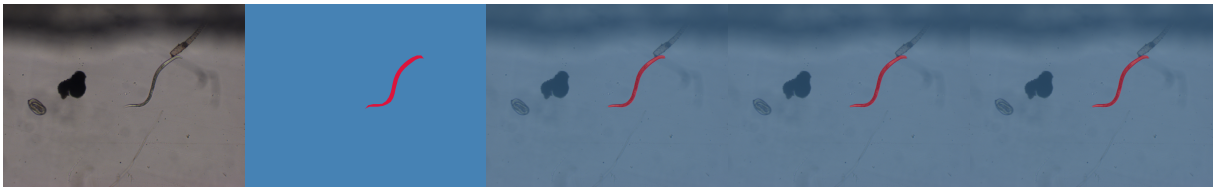
Como se observa en la tabla 4.1, la configuración STDC1 requiere de menos parámetros y GFLOPS, lo que lo hace un modelo más ligero y con menor capacidad computacional requerida; sin embargo, el entrenamiento requiere de alrededor de 20 horas, a diferencia de la configuración STDC2 que requiere de aproximadamente 22 horas de ejecución.

Con respecto al porcentaje de mIoU, este ronda los valores entre 90 % al 91.5 % entre las máscaras y las predicciones, tanto en el proceso de validación como en el proceso de prueba con imágenes no antes vistas en el entrenamiento. Estos porcentajes indican que se segmenta alrededor de un 90 % de los nematodos definidos en las máscaras, obteniendo resultados donde el nematodo es completamente segmentado como se visualiza en la figura 4.1.

(a) Original **(b)** Máscara **(c)** STDC1 base **(d)** STDC2 base **(e)** STDC2 Aumentado 2



(f) Original **(g)** Máscara **(h)** STDC1 Preentrenado **(i)** STDC2 base **(j)** STDC2 preentrenado

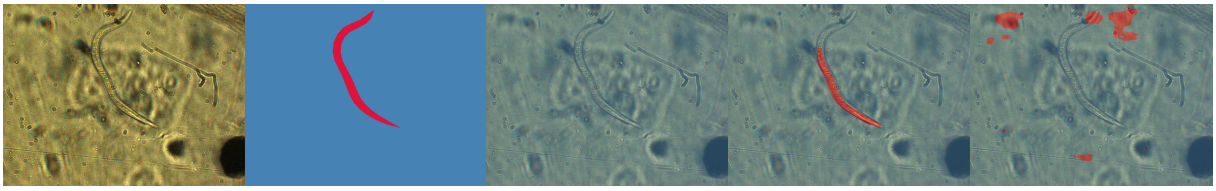


(k) Original **(l)** Máscara **(m)** STDC2 Base **(n)** STDC2 Preentrenado **(ñ)** STDC2 Aumentado 2

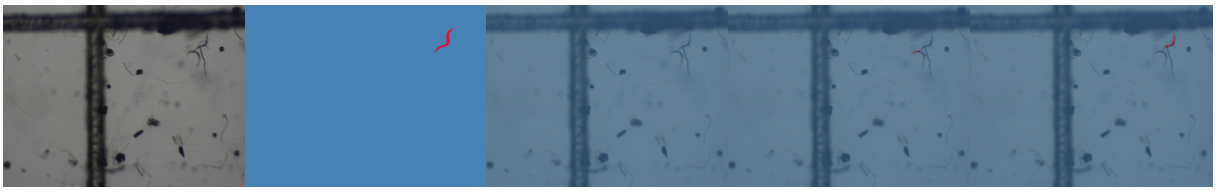
Figura 4.1: Predicciones correctas del modelo STDC.

El 10 % restante comprende resultados como los de la figura 4.2, donde el nematodo no se identifica o es segmentado parcialmente. Inclusive se presentan casos donde el nematodo segmentado por el modelo no fue segmentado en las máscaras como en la figura 4.3.

La pérdida de entropía cruzada utilizada por el modelo, ilustrada en la figura 4.4, indica la disminución del error del modelo durante el entrenamiento que se estabiliza alrededor de un 0.20 para todas las configuraciones realizadas. La pérdida tanto en el entrenamiento como en la validación siguen una tendencia muy similar, lo que indica el constante aprendizaje de la red sin llegar al sobreajuste.

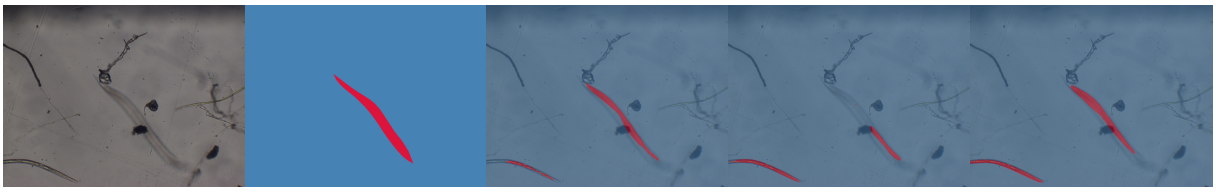


(a) Original **(b)** Máscara **(c)** STDC1 base **(d)** STDC2 Preentrenado **(e)** STDC2 Aumentado 2



(f) Original **(g)** Máscara **(h)** STDC1 Base **(i)** STDC2 base **(j)** STDC1 Preentrenado

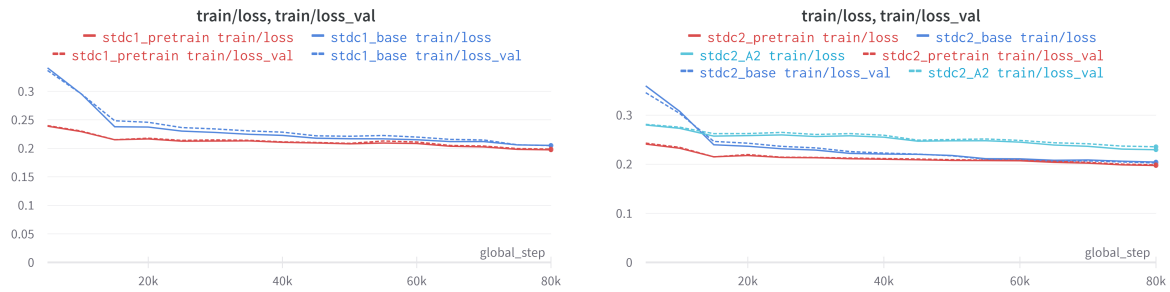
Figura 4.2: Predicciones incorrectas del modelo STDC.



(a) Original **(b)** Máscara **(c)** STDC1 Preentrenado **(d)** STDC2 Preentrenado **(e)** STDC2 Aumentado 2

Figura 4.3: Predicciones de múltiples nematodos del modelo STDC.

Utilizando la precisión y exhaustividad de cada configuración se obtiene el frente de Pareto que se observa en la figura 4.5, el cual destaca las configuraciones óptimas para el modelo STDC con el conjunto de datos de nematodos. De las cinco configuraciones realizadas, solamente dos configuraciones resultan óptimas en la segmentación de nematodos; estas corresponden a las configuraciones STDC1 preentrenado y STDC2 preentrenado.



(a) STDC 1 **(b)** STDC 2

Figura 4.4: Gráficos de pérdida del modelo STDC.

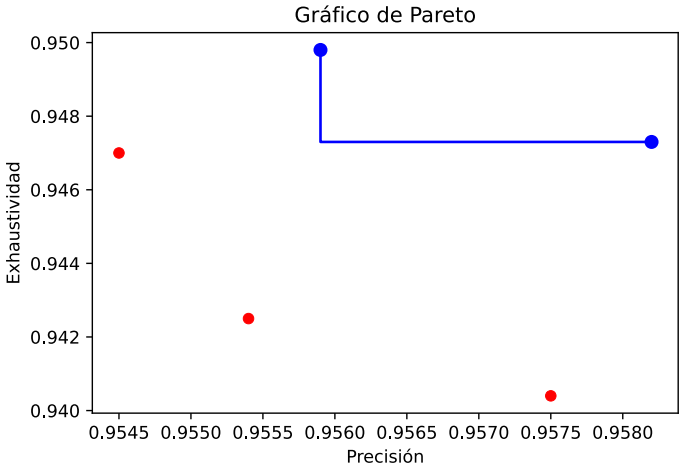
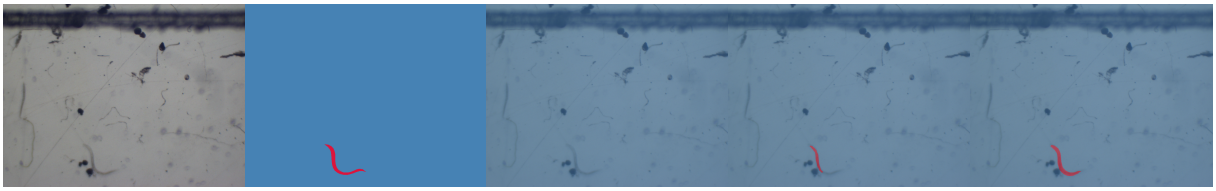


Figura 4.5: Frente de Pareto del modelo STDC.

A pesar de la ligera diferencia en los resultados, hay imágenes que los modelos óptimos si logran segmentar y los modelos dominados no segmentan parcial o completamente, como se observa en la figura 4.6.



(a) Original **(b)** Máscara **(c)** STDC1 base **(d)** STDC2 Base **(e)** STDC2 Preentrenado

Figura 4.6: Predicciones variables entre modelos STDC.

4.2. Segmentación con el modelo PointRend

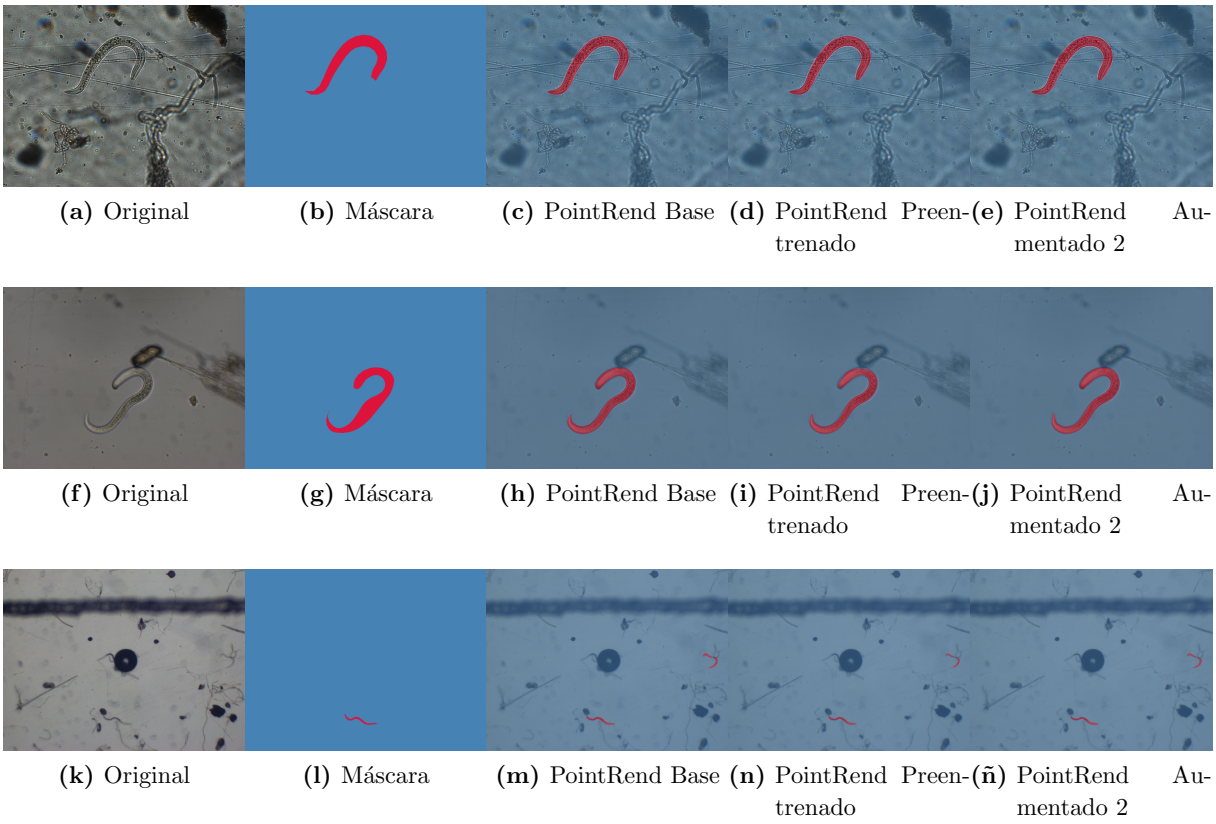
El rendimiento del modelo PointRend se evalúa con cuatro configuraciones. Se ejecuta una prueba de un entrenamiento base, así como una prueba con pesos preentrenados de un modelo del proyecto *MMSegmentation* con el conjunto de datos *Cityscapes*. Se elige *Cityscapes* ya que es el conjunto datos con el mayor rendimiento en IoU. Además, se analiza el comportamiento del modelo cuando los datos no poseen transformaciones de rotación y distorsión fotométrica, a este caso se le da el nombre de aumentado 2. Los resultados de estos experimentos se encuentran detallados en la tabla 4.2. Por último, también se realizan pruebas al alterar la cantidad de puntos a evaluar en el módulo PointRend.

Tabla 4.2: Comparación de resultados de las configuraciones del modelo PointRend.

Preentrenado	Rotación	Puntos	Params [M]	GFLOPs	mIoU val [%]	mIoU test [%]
No	Sí	2048	47.69	522.6	91.27	91.06
Sí	Sí				91.05	90.80
Sí	No				78.67	81.26
No	Sí	3136	47.69	521.83	91.08	91.1

En comparación al modelo STDC, el PointRend necesita de una mayor capacidad computacional al considerar la cantidad de parámetros y GFLOPS; sin embargo, el proceso de entrenamiento del modelo ronda las 10 horas y 40 minutos de ejecución.

Como se detalla en la tabla 4.2 la métrica de mIoU se mantiene en un porcentaje alrededor del 91 %, lo que indica que PointRend es capaz de reconocer nematodos de diferentes tamaños y figuras, inclusive mejor que en las máscaras dadas, como se demuestra en la figura 4.7. Una excepción se da en el caso de la configuración aumentado 2, cuyo porcentaje se mantiene inferior al 82 %, lo que significa, en comparación a las demás configuraciones, errores en la segmentación de objetos del fondo con texturas similares y la no identificación de nematodos en la imagen, como se ilustra en la figura 4.8.



(a) Original **(b)** Máscara **(c)** PointRend Base **(d)** PointRend Preentrenado **(e)** PointRend Aumentado 2

(f) Original **(g)** Máscara **(h)** PointRend Base **(i)** PointRend Preentrenado **(j)** PointRend Aumentado 2

(k) Original **(l)** Máscara **(m)** PointRend Base **(n)** PointRend Preentrenado **(ñ)** PointRend Aumentado 2

Figura 4.7: Predicciones correctas del modelo PointRend.

(a) Original **(b)** Máscara **(c)** PointRend Aumentado 2

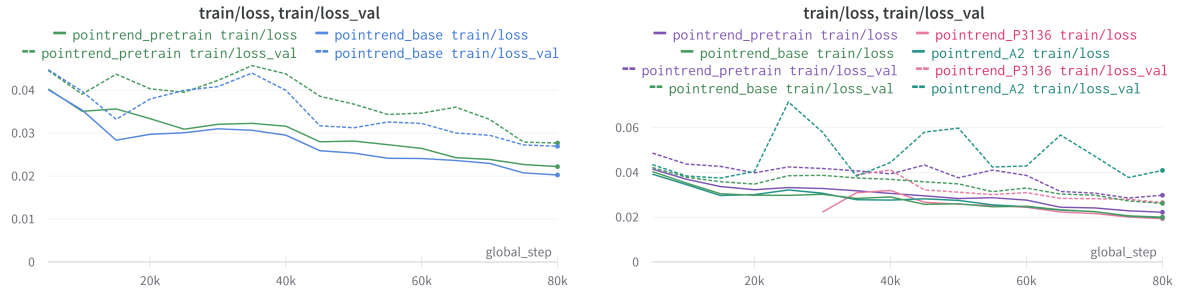
(d) Original **(e)** Máscara **(f)** PointRend Aumentado 2

(g) Original **(h)** Máscara **(i)** PointRend Aumentado 2

Figura 4.8: Predicciones incorrectas del modelo PointRend.

El modelo utiliza una pérdida de entropía cruzada. En la figura 4.9 se visualizan dos casos de entrenamiento base, dos casos de preentrenamientos y un caso sin transformaciones de rotación y distorsión. La pérdida del modelo tiene una tendencia a disminuir en el entrenamiento llegando a valores aproximados de 0.020 para todas las configuraciones realizadas. La pérdida en la validación tiende a conservar valores mayores a los obtenidos en el entrenamiento de aproximadamente 0.027. El caso del aumentado 2 difiere con el entrenamiento ya que se tienen picos de sobreentrenamiento de la red a lo largo de las 80000 iteraciones.

Con los resultados de precisión y exhaustividad se construye un frente de Pareto para determinar los modelos óptimos en la segmentación de nematodos. Los modelos que conforman el frente de Pareto corresponden a aquellos entrenados de base, al modelo entrenado con 3136 puntos y uno de los modelos preentrenados. El caso del modelo aumentado 2 es el que se encuentra más a la izquierda del frente de Pareto en la figura 4.10, considerado como un modelo dominado que siempre va a ser superado por el rendimiento de uno o varios de los modelos óptimos presentes en el frente. El caso de uno de los modelos preentrenados que se encuentra dominado por el frente es un caso inusual en precisión



(a) **(b)**

Figura 4.9: Gráficos de pérdida del modelo PointRend.

debido a la estabilidad del modelo.

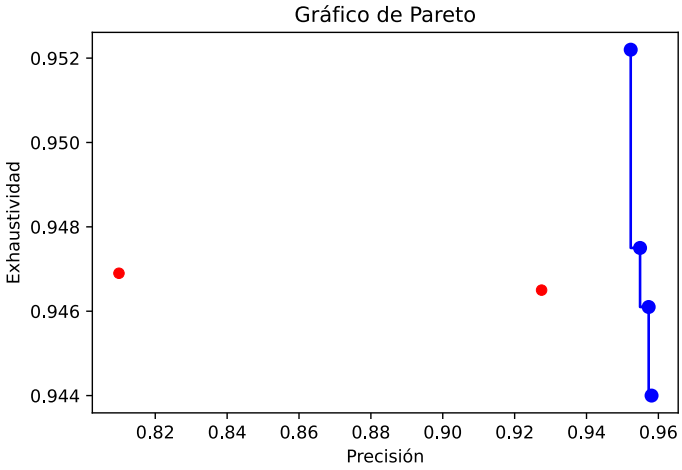


Figura 4.10: Frente de Pareto del modelo Pointrend.

4.2.1. Cantidad de puntos

Uno de los hiperparámetros del modelo es la cantidad de puntos que tendrá el set en el modulo PointRend. Variar este parámetro tiene una influencia poco significativa en los resultados [39]. Además, entre mayor sea la cantidad de puntos, mayor será la cantidad de GFLOPS necesaria para el entrenamiento de la red, como se visualiza en la figura 4.11.

Aunque es poco significativa la diferencia de resultados, en la figura 4.12, se aprecia que con un mayor número de puntos, la pérdida tiende a una mayor disminución del error, por lo que entre más puntos se utilicen, el modelo se estabilizará en una menor pérdida lo que disminuye el error en las predicciones. Estas pérdidas en el entrenamiento van desde el 0.026 en el caso de 49 puntos, al 0.021 en el caso de 3136 puntos. La pérdida en la validación continúa la tendencia del entrenamiento, al tratarse de una prueba base.

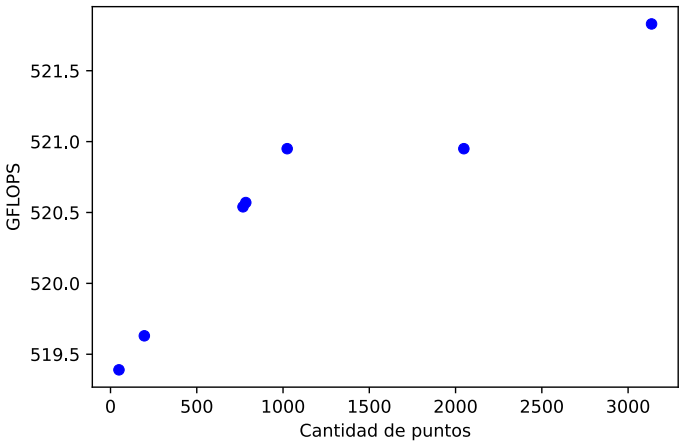


Figura 4.11: GFLOPS en función de la cantidad de puntos.

(a) **(b)**

Figura 4.12: Pérdida en la configuración de hiperparámetros del modelo pointrend.

4.3. Segmentación con el modelo Segformer

Los modelos que utilizan *transformers* necesitan de una mayor capacidad computacional para su entrenamiento. Por ello se realiza una comparación de los dos modelos ligeros con *transformers* disponibles para entrenar en la tarjeta A100. Como se visualiza en la tabla 4.3 el modelo SegformerB0 es más liviano en cuanto a parámetros y es entrenable solamente con 10GB de memoria GPU, a diferencia del modelo *Segmenter Tiny* que ocupa de 20 GB de memoria GPU. Además, el modelo SegformerB0 tiene un mejor rendimiento con respecto a las métricas mIoU, precisión y exhaustividad. Debido a estos resultados, el modelo SegformerB0 es elegido como uno de los 3 modelos a comparar.

Tabla 4.3: Modelo Segformer B0 vs Segmenter Tiny.

Modelo	Params [M]	GFLOPS	mIoU [%]	Precisión	Exhaustividad	Tarjeta
Segformer B0	3.72	51.07	91.13	0.9671	0.9368	A100 10GB
Segmenter Tiny	6.68	35.75	85.17	0.9097	0.9169	A100 20GB

Al igual que los modelos anteriores, se realizan pruebas de entrenamiento base, entrenamiento con preentrenamiento con el conjunto *Cityscapes* y pruebas con la falta de transformaciones de rotación y distorsión fotométrica. Para este modelo se presentan seis configuraciones de hiperparámetros relacionados a la cantidad de capas del modelo. Se realizan pruebas del modelo más ligero B0 diseñado para mayor rapidez de entrenamiento y el más grande soportado por las tarjetas disponibles, correspondiente al B4 diseñado para un mayor rendimiento. Además, este modelo está previamente programado para entrenar con 160000 iteraciones, por lo que se realizan pruebas con esta cantidad y con 80000 iteraciones para que sea comparable con los otros modelos.

Tabla 4.4: Comparación de resultados de las configuraciones del modelo Segformer.

Modelo	Iteraciones	Rotación	Preentrenado	Params [M]	GFLOPs	mIoU val [%]	mIoU test [%]
B0	80 k	Si	No	3.72	51.07	91.13	90.53
		Si	Si			91.26	91
		No	Si			91.27	90.88
	160 k	Si	No			90.76	90.97
		Si	Si			91.29	91.02
		No	Si			91.11	91.21
B4	160 k	Si	No	61.37	325.03	91.66	91.42
		Si	Si			91.64	91.4

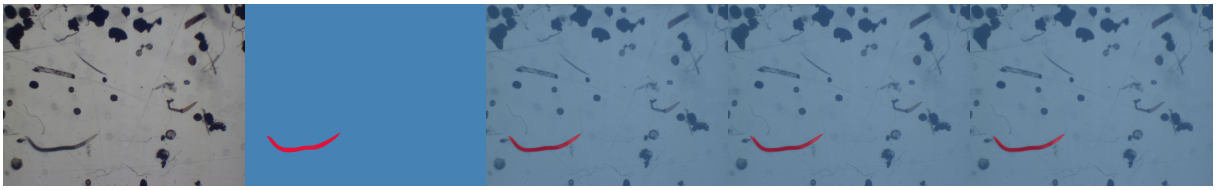
Basado en los resultados obtenidos de un modelo ligero como el B0 y un modelo más grande como el B4, estos cumplen con los propósitos de diseño, ya que en términos de rapidez el modelo B0 sobresale al tardar alrededor de 8 horas, al contrario al modelo B4 que dura aproximadamente 1 día. En términos de rendimiento, el modelo B4 sobresale con un máximo de 91.64 % de mIoU cuando el modelo B0 obtiene un máximo de 91.29 % de mIoU. A pesar de que el modelo B4 tiene un mejor rendimiento, este solamente tiene una diferencia positiva de 0.37 %. Tomando en cuenta el conjunto de datos, tiempo disponible y la capacidad computacional de las tarjetas GPU disponibles, es idóneo trabajar con el modelo B0.

El mayor rendimiento en mIoU obtenido por el modelo Segformer B0 es de 91.29 % al utilizar 160000 iteraciones y preentrenar el modelo con lo disponible en la biblioteca *MMSegmentation*. En general todas las configuraciones rondan los valores del 91 % de IoU, lo que permite obtener resultados donde el nematodo es segmentado de la manera esperada en las máscaras como se observa en la figura 4.13. A pesar de ello, aún existe un porcentaje de error al segmentar las imágenes de nematodos, esto se ve en imágenes donde los nematodos son de menor tamaño o hay nematodos cruzados, como se observa en la figura 4.14.

El modelo Segformer utiliza una pérdida de entropía cruzada cuya tendencia se ilustra en la figura 4.15. Estas gráficas indican la disminución del error del modelo que se estabiliza alrededor de un 0.0065 para las configuraciones realizadas. La pérdida tanto en el entrenamiento como en la validación siguen una tendencia muy similar a lo largo del entrenamiento con una diferencia aproximada del 0.0025 entre las curvas.

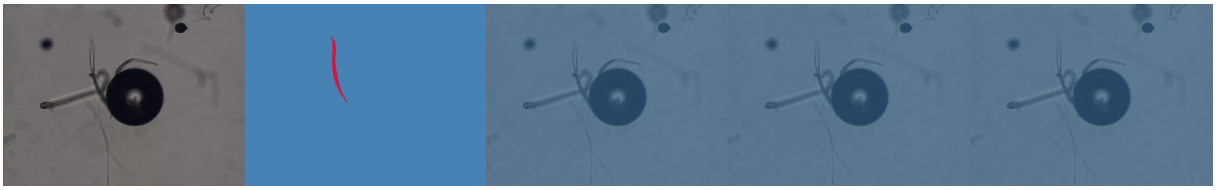
(a) Original **(b)** Máscara **(c)** SegformerB0 Pre-entrenado 80k **(d)** SegformerB0 Base 160k **(e)** SegformerB4 Pre-entrenado

(f) Original **(g)** Máscara **(h)** SegformerB0 Base 80k **(i)** SegformerB0 Pre-entrenado 80k **(j)** SegformerB0 Aumentado 2 80k

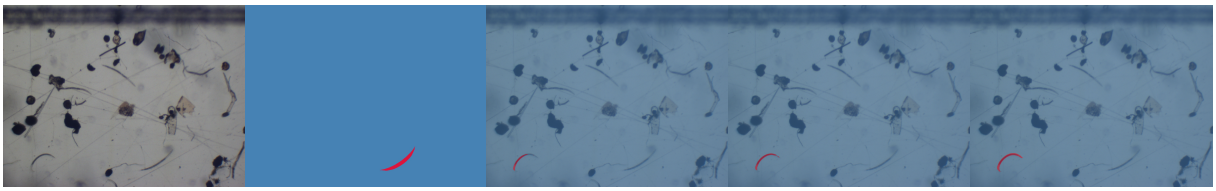


(k) Original **(l)** Máscara **(m)** SegformerB0 Base 160k **(n)** SegformerB0 Pre-entrenado 160k **(ñ)** SegformerB4 Base

Figura 4.13: Predicciones correctas del modelo Segformer.



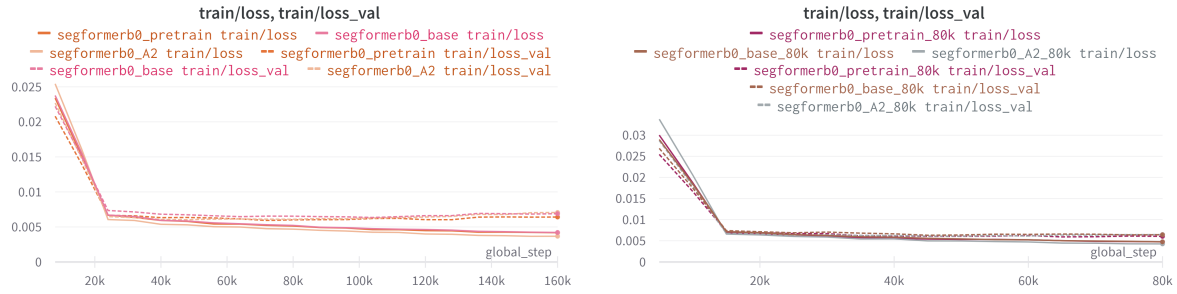
(a) Original **(b)** Máscara **(c)** SegformerB0 Pre-entrenado 16k **(d)** SegformerB0 Pre-entrenado 80k **(e)** SegformerB0 Aumentado 2 80k



(f) Original **(g)** Máscara **(h)** SegformerB0 Base 160k **(i)** SegformerB0 Pre-entrenado 160k **(j)** SegformerB4 Pre-entrenado

Figura 4.14: Predicciones incorrectas del modelo Segformer.

Mediante los resultados de precisión y exhaustividad se obtiene el frente de Pareto que determina los modelos óptimos entre las ocho configuraciones realizadas. El modelo SegformerB4 fue diseñado para un mejor rendimiento, por lo que este modelo junto al SegformerB0 preentrenado con 160000 iteraciones forman parte del frente de Pareto, como se observa en la figura 4.16a, siendo entonces los modelos óptimos a utilizar con el conjunto

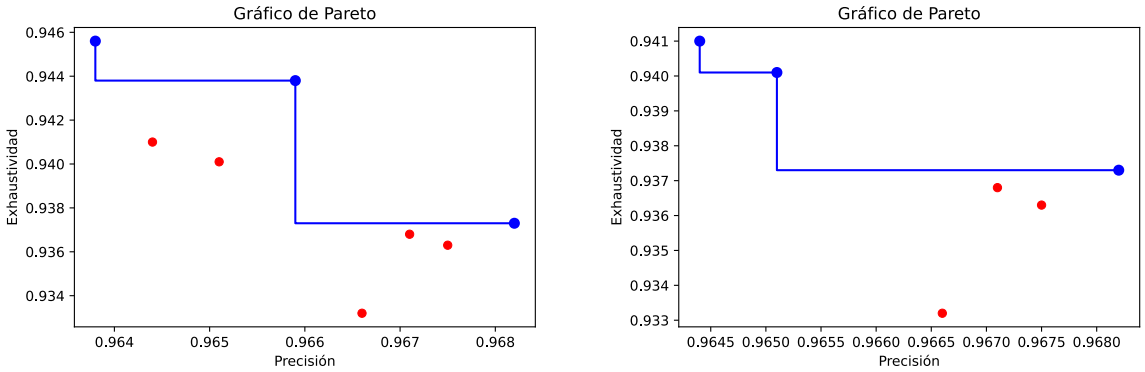


(a) 160000 iteraciones **(b)** 80000 iteraciones

Figura 4.15: Gráficos de pérdida del modelo Segformer.

de datos de nematodos.

Sin embargo, si evitamos el uso del modelo SegformerB4 debido a la capacidad computacional que necesita para su entrenamiento. Los modelos dominantes corresponden a los modelos SegformerB0 preentrenados, así como el modelo SegformerB0 con aumentado 2 con 160000 iteraciones como se observa en la figura 4.16b.



(a) Con el modelo SegformerB4 **(b)** Sin el modelo SegformerB4

Figura 4.16: Frente de Pareto del modelo Segformer.

4.4. Estabilidad de los modelos de segmentación

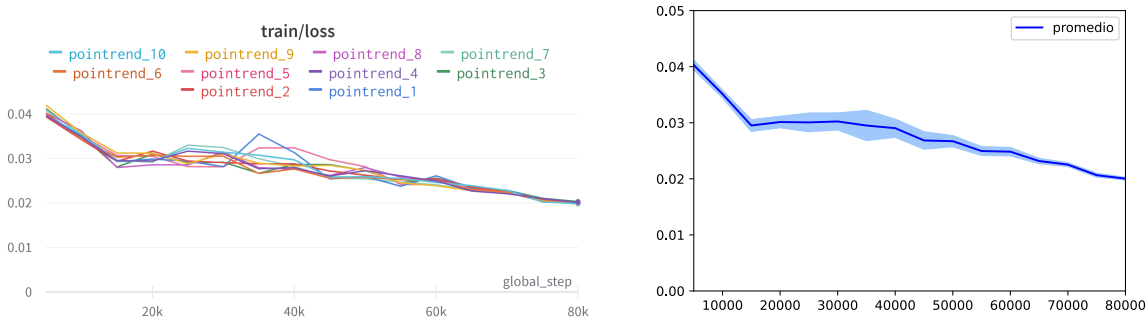
Debido al caso experimentado en el modelo PointRend, donde la repetición del modelo preentrenado provocó una variación visualmente significativa de los resultados de IoU y precisión, se lleva a cabo un experimento para revisar la estabilidad de los entrenamientos del modelo. Este experimento se aplica a los modelos PointRend debido al caso previamente presentado, y al modelo SegformerB0 por su brevedad en el tiempo ejecución comparado al modelo STDC.

Para establecer la estabilidad de los modelos, se llevan a cabo 10 entrenamientos del modelo sin hacer ningún cambio en los hiperparámetros. De los datos de pérdida de

los 10 entrenamientos se obtiene el promedio y desviación estándar. De estos datos se genera una gráfica donde se indica el promedio de pérdida y la dispersión que tienen los entrenamientos con respecto al promedio.

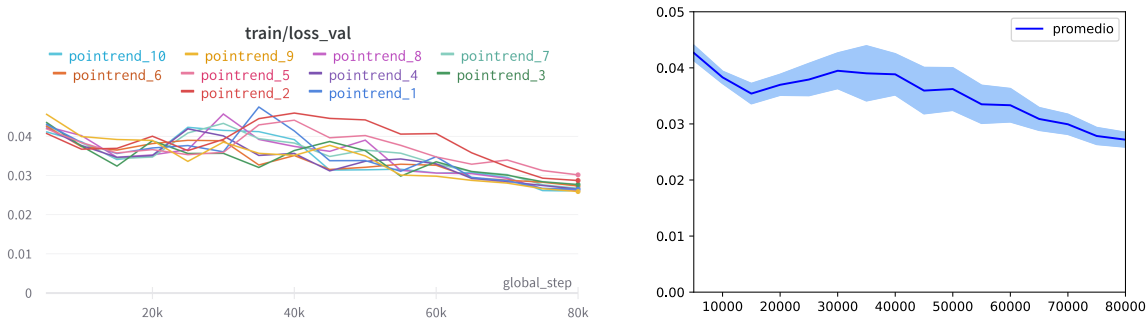
4.4.1. Estabilidad del modelo PointRend

Se lleva a cabo una prueba de estabilidad con el modelo base del PointRend. Como se observa en la figura 4.17, las 10 pruebas realizadas se desvían ligeramente del promedio, obteniendo una desviación estándar máxima del 0.0026178. Sin embargo, el proceso de validación tiene una mayor variación a comparación en el entrenamiento como se observa en la figura 4.18, con una desviación estándar máxima del 0.0048662, un poco menos del doble que en el entrenamiento.



(a) Original **(b)** Promedio y desviación

Figura 4.17: Estabilidad del entrenamiento en PointRend.

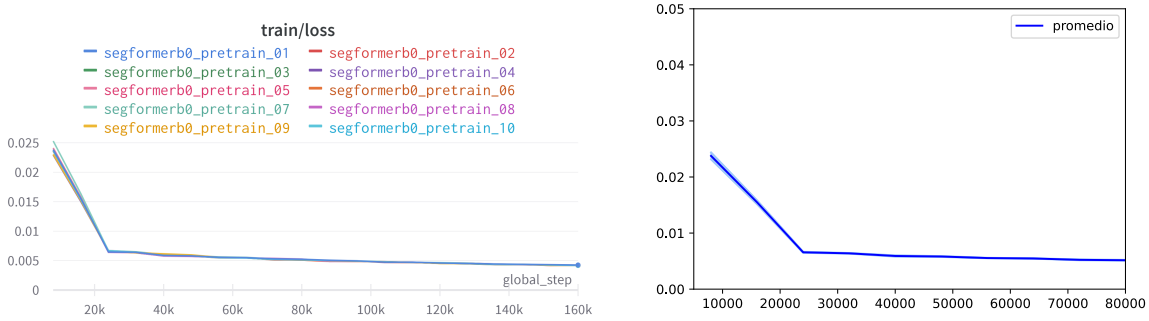


(a) Original **(b)** Promedio y desviación

Figura 4.18: Estabilidad de la validación en PointRend.

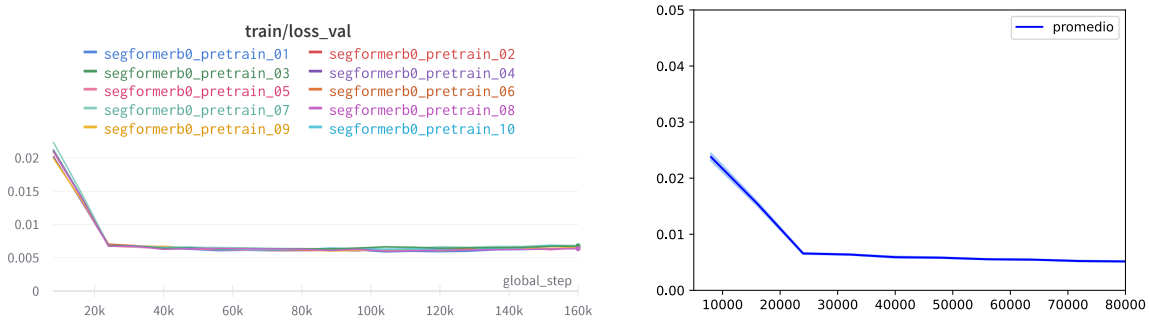
4.4.2. Estabilidad del modelo Segformer

Se lleva a cabo una prueba de estabilidad con el modelo preentrenado del SegformerB0. Como se observa en la figura 4.19, las 10 pruebas realizadas se mantienen cercanas al promedio de manera que visualmente no se denota la variación como en el modelo PointRend, obteniendo una desviación estándar máxima del 0.0006155. De igual manera, los resultados del proceso de validación se mantiene cercano al promedio como se observa en la figura 4.20, con una desviación estándar máxima del 0.0006155, igual al entrenamiento.



(a) Original **(b)** Promedio y desviación

Figura 4.19: Estabilidad del entrenamiento en Segformer.



(a) Original **(b)** Promedio y desviación

Figura 4.20: Estabilidad de la validación en Segformer.

En cuanto a pérdida y desviación estándar, el modelo SegformerB0 resulta más estable que el modelo PointRend. El modelo PointRend puede requerir de una serie de pruebas para asegurar un modelo con una menor pérdida, lo que no es necesario con el modelo SegformerB0.

En cuanto a la métrica de validación mIoU, se puede observar la variación mediante los diagramas de cajas de la figura 4.21, donde se observa que el modelo PointRend tiene valores que varían entre el 90 % y el 91 % y posee valores ligeramente alejados del promedio.

Al contrario, el modelo SegformerB0 se mantiene más estable con valores alrededor del 91 % que mantienen una distribución equitativa alrededor del promedio.

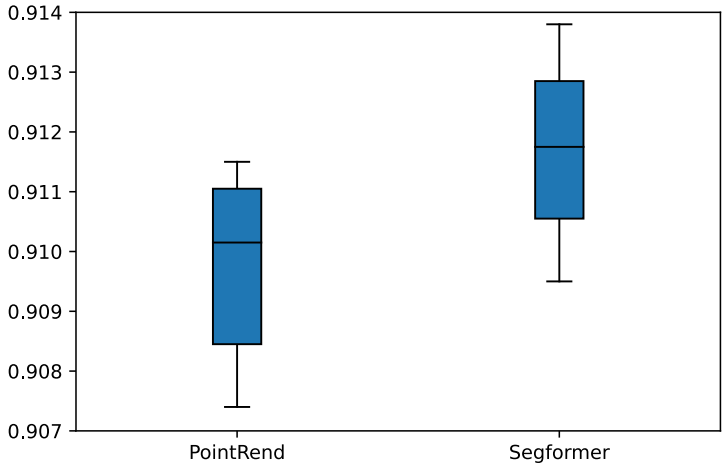


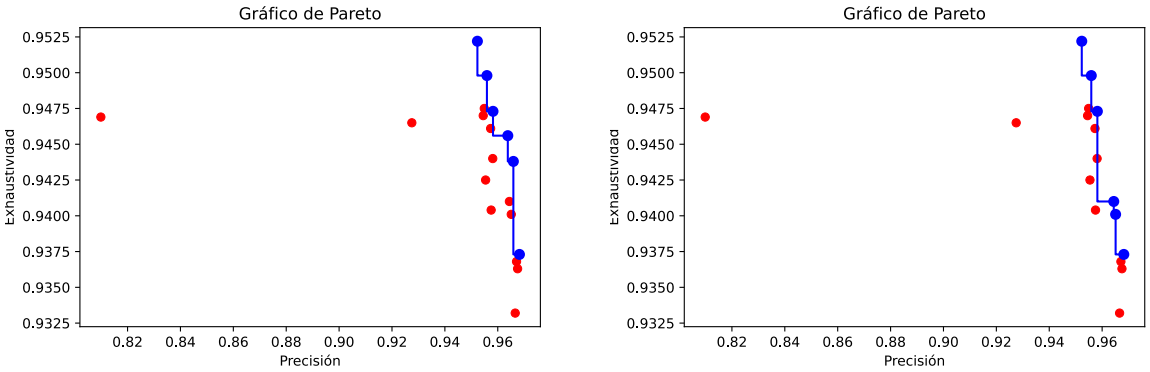
Figura 4.21: Estabilidad de la métrica mIoU.

4.5. Análisis comparativo de los modelos de segmentación semántica empleados

Se lleva a cabo una comparación de todos los modelos entrenados mediante la realización de un frente de Pareto. Para ello se toma en cuenta la precisión y exhaustividad de cada modelo. Los modelos que se encuentran en el frente se consideran como modelos óptimos y aquellos que no lo están, son modelos que se encuentran dominados por aquellos en el frente. Esto quiere decir que siempre va a haber al menos un modelo que obtenga un mejor rendimiento. Al comparar todas las configuraciones realizadas se obtienen 6 modelos óptimos en la segmentación de nematodos, como se observa en la figura 4.22. Los modelos óptimos corresponden a:

- Modelo STDC1 preentrenado
- Modelo STDC2 preentrenado
- Modelo PointRend base
- Modelo SegformerB0 preentrenado con 160000 iteraciones
- Modelo SegformerB4 base
- Modelo SegforemrB4 preentrenado

Como se observa en la figura 4.22 la variación en el rendimiento de los modelos es mínima en el parámetro de precisión, a excepción de dos modelos. Esta mínima variación en los modelos resulta en la similitud de resultados del rendimiento.



(a) Con el modelos SegformerB4 **(b)** Sin el modelos SegformerB4

Figura 4.22: Frente de Pareto de los modelos evaluados.

Por otro lado, el modelo SegformerB4, como se menciona anteriormente, toma una capacidad computacional mayor a los demás modelos. El SegformerB4 requiere de al menos una GPU V100 con 32 GB de memoria, a diferencia del resto de modelos que tienen una capacidad computacional para entrenar en una máquina virtual que utiliza una A100 con 10GB de memoria. El modelo SegformerB4 se descarta como modelo a utilizar con el conjunto de datos de nematodos debido a la capacidad computacional requerida, al tiempo de entrenamiento y a la mínima variación de la métrica mIoU, cuya mejora comparada al resto de modelos es de un 0.29 %, como se observa en la tabla 4.5. Esto lleva además a un nuevo frente de Pareto, donde los modelos SegformerB0 preentrenado con 80000 iteraciones y Segformer aumentado 2 con 160000 iteraciones toman el lugar del modelo Segformer B4, como se observa en la figura 4.22b

Tabla 4.5: Resultados de modelos óptimos.

Modelo	Configuración	Params [M]	GFLOPS	mIoU [%]	mPrecision	mRecall
STDC1	Preentrenado	8.57	67.66	91.33	0.9582	0.9473
STDC2	Preentrenado	12.6	94.13	91.37	0.9559	0.9498
PointRend	Base	47.69	522.6	91.27	0.9523	0.9522
SegformerB0	Preentrenado 160000 iteraciones	3.72	51.07	91.29	0.9644	0.9410
SegformerB0	Preentrenado 80000 iteraciones	3.72	51.07	91.26	0.9373	0.9682
SegformerB0	Aumentado 2 160000 iteraciones	3.72	51.07	91.11	0.9675	0.9363
SegformerB4	Base	61.37	325.03	91.66	0.9659	0.9438
SegformerB4	Preentrenado	61.37	325.03	91.64	0.9638	0.9456

Obtenidos los modelos óptimos, se realiza un análisis de la métrica de intersección sobre unión (IoU) utilizada mayormente en tareas de segmentación. En la tabla 4.5 se listan los valores de mIoU obtenidos en la validación y prueba de cada modelo. El modelo que sobresale en esta métrica corresponde al STDC2 preentrenado con un 91.37 %, el cual se considera como el modelo idóneo a utilizar en la segmentación semántica de nematodos. Esto sin embargo, es dependiente a la estabilidad de los modelos. Como los valores de mIoU varían muy poco entre ellos, existe la posibilidad que una prueba adicional de otro de los modelos óptimos mejore ligeramente este porcentaje. Este es el caso de uno de los casos de prueba para la estabilidad del modelo SegformerB0 preentrenado con 160000 iteraciones, el cual alcanzó un mIoU de 91.38 %.

Los valores de mIoU que rondan el 91 % de éxito en la predicción de nematodos respecto a la máscara dada, se reflejan en los resultados obtenidos de las predicciones de los modelos óptimos. Estos modelos cumplen con la tendencia de segmentar aquellas imágenes donde el nematodo es de gran tamaño, es decir que abarca la mayor parte de la imagen y solo hay un nematodo, como por ejemplo la figura 4.23.

(a) Original **(b)** Máscara **(c)** Predicción

Figura 4.23: Predicción correcta de un solo nematodo de gran tamaño.

El caso de las imágenes donde el nematodo en pequeño puede o no ser reconocido, así como que la segmentación no cumpla con la segmentación correcta de los bordes de los nematodos como se observa en la figura 4.24.

Por último, se encuentra el caso donde los nematodos son pequeños y se puede encontrar más de un nematodo en la imagen pero no en la máscara. En estos casos es donde la tarea de segmentación tiende a fallar, ya sea porque el nematodo no es segmentado correctamente, se toma en cuenta un nematodo distinto al especificado en la máscara o se segmentan partes de diferentes nematodos. Ejemplos de estos casos se ilustran en la figura 4.25.

(a) Original **(b)** Máscara **(c)** Predicción

(d) Original **(e)** Máscara **(f)** Predicción

Figura 4.24: Predicción en nematodos de tamaño pequeño.

(a) Original **(b)** Máscara **(c)** Predicción

(d) Original **(e)** Máscara **(f)** Predicción

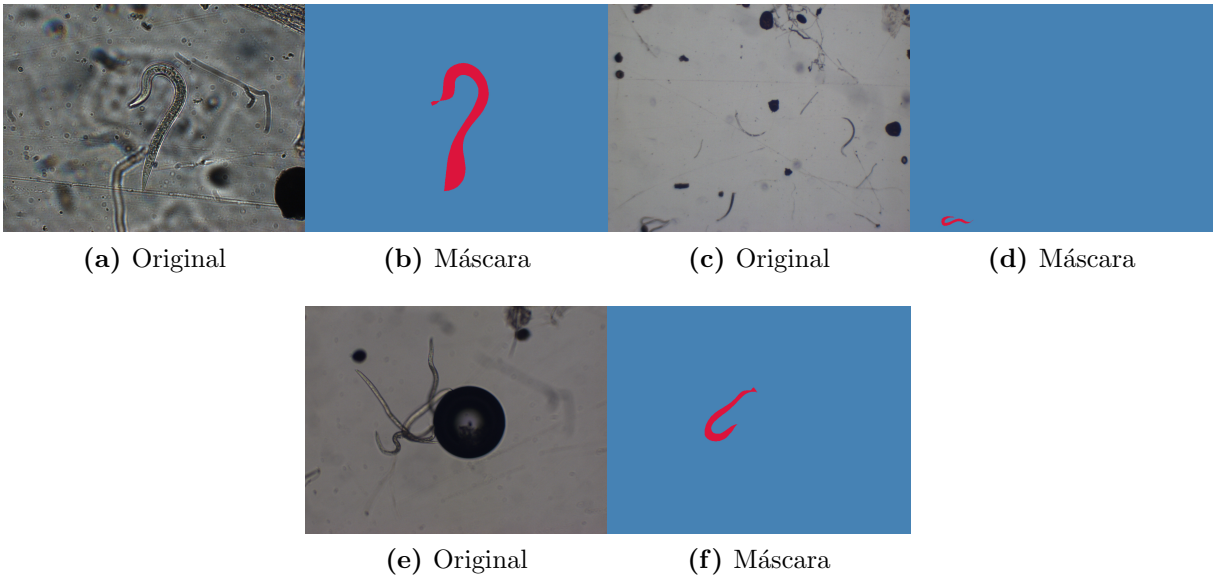
(g) Original **(h)** Máscara **(i)** Predicción

Figura 4.25: Casos de segmentación fallida. La primera fila es un ejemplo donde el nematodo no es segmentado correctamente. En la segunda fila se ilustra un ejemplo donde se toma en cuenta un nematodo distinto al especificado en la máscara. La tercera fila es un ejemplo donde se segmentan partes del nematodo especificado en la máscara y otro nematodo distinto.

4.5.1. Análisis de máscaras de segmentación

En todos los modelos entrenados se obtiene alrededor de un 91 % de éxito en la media de la intersección de la predicción con la máscara (mIoU). Con el conjunto de datos disponible los resultados se encuentran sesgados a este porcentaje. Para buscar mejorar estos resultados con los modelos entrenados se debe trabajar en el conjunto de datos de manera que se expanda la cantidad de datos o se perfeccionen las máscaras de segmentación.

Como se observa en la figura 4.26, las máscaras poseen algunas imperfecciones que pueden estar afectando el resultado de la métrica IoU. Como por ejemplo, el ancho del nematodo en las máscaras puede ser más grueso que en la imagen original, lo que implica que se están seleccionando píxeles correspondientes al fondo como nematodo. Otro de los casos comunes es la selección de un solo nematodo en la máscara, cuando en la imagen hay dos o más. Esto lleva a la confusión de la red, ya que píxeles con características de nematodos están siendo clasificados como fondo.



(a) Original **(b)** Máscara **(c)** Original **(d)** Máscara

(e) Original **(f)** Máscara

Figura 4.26: Máscaras de segmentación erróneas.

Capítulo 5

Conclusiones

Este trabajo propone la comparación de los modelos de segmentación semántica disponibles en el proyecto *MMSegmentation* de *OpenMMLab*, propuestos en los últimos tres años en la tarea concreta de segmentar nematodos. La comparación consta de los criterios de capacidad computacional, rendimiento y la columna del modelo utilizada. En el conjunto de modelos evaluados, se encontraron tres modelos que cumplen con las necesidades y limitaciones del proyecto.

Los modelos de segmentación semántica STDC, PointRend y Segformer, demuestran resultados óptimos y competentes en la tarea de segmentación semántica en imágenes de microscopía óptica de nematodos. El rendimiento de los modelos alcanza alrededor de un 91 % de éxito en las predicciones, basado en la media de la intersección de la misma con las máscaras de segmentación (mIoU).

Se utilizan los hitos disponibles para la generación de máscaras de segmentación semántica para un solo nematodo en la imagen. De los hitos disponibles se generan eficazmente las máscaras del 52.65 % de los datos, formando un conjunto de datos de 3827 imágenes. Debido a la reducida cantidad de imágenes se tuvo que aplicar rotación, volteado, escalamiento, recortado y distorsión fotométrica como estrategias de aumento de datos.

Se llevan a cabo una configuración llamada aumentado 2 para cada modelo. Esta configuración pretende corroborar la utilidad de las transformaciones de rotación y distorsión fotométrica. Se obtiene que en los modelos con columna basada en redes convolucionales, el uso de estas transformaciones disminuyen ligeramente el rendimiento de la red. Por el contrario, el modelo con columna basada en transformadores mejora el rendimiento sin el uso de las transformaciones.

Si bien se logran resultados de mIoU de de alrededor de un 91 %, las máscaras de segmentación se pueden mejorar en la clasificación de los bordes así como clasificar más de un nematodo por imagen. Estos errores en las máscaras afectan el valor de IoU, por ejemplo, en las predicciones donde se clasifican más de un nematodo o cuando se clasifica un nematodo diferente al de la máscara. La corrección de las etiquetas de entrenamiento pueden llevar a un porcentaje de mIoU mayor al 91 % obtenido.

Como trabajo futuro, primeramente se debe mejorar el conjunto de máscaras de segmentación. Se busca que las máscaras de segmentación demarquen de manera correcta los bordes de los nematodos para evitar confusiones con el fondo al predecir. Además, se deben generar máscaras que identifiquen a todos los nematodos en una imagen para que los modelos no se confundan y de igual manera identifiquen a todos los nematodos de una imagen.

El siguiente paso, tras mejorar las máscaras, corresponde a llevar a cabo una segmentación de instancia. Para ello, el proyecto *MMSegmentation* provee modelos capaces de realizar esta tarea de segmentación como el modelo K-net. Esta segmentación por sus características brinda los resultados para llevar a cabo la herramienta de conteo e identificación de nematodos.

Bibliografía

[1] K. O'neal Coto, *La agricultura costarricense se reinventa frente a la pandemia del COVID-19*, 2020. dirección: https://www.ucr.ac.cr/noticias/2020/05/16/la-agricultura-costarricense-se-reinventa-frente-a-la-pandemia-del-covid-19.html.

[2] IICA Costa Rica, «La agricultura de Costa Rica: Situación al 2010, su evolución y prospectiva,» *ICAA: Política de Estado para el Sector Agroalimentario y el Desarrollo Rural costarricense*, 2010.

[3] E. Mora Monge, J. C. Jiménez Flores, J. R. Wong Ruíz, L. Jiménez Carvajal y S. Mora Ramírez, «Informe de Gestión del Sector Agropecuario, Pesquero y Rural Mayo 2020 – Abril 2021,» 2021.

[4] O. Sotomayor, E. Ramírez y H. Martínez, «Digitalización y cambio tecnológico en las mipymes agrícolas y agroindustriales en América Latina,» Naciones Unidas, inf. téc., 2021.

[5] M. E. Chaves Gómez, «Densidad y diversidad de nematodos fitoparásitos y de suelo en sistemas orgánicos y convencionales de café en asocio con banano en el Valle Central y Occidental de Costa Rica,» Tesis de mtría., CATIE, Turrialba, 2014.

[6] Ó. A. Guzmán Piedrahita, J. Castaño Zapata y B. Villegas Estrada, «Principales nematodos fitoparásitos y síntomas ocasionados en cultivos de importancia económica,» *Estimación histopatológica del grado de infección inducido por Stagonospora nodorum (berk.) castellani & germano en plántulas de trigo (Triticum aestivum L.)*, pág. 38, 2012.

[7] H. Lopez-Nicora, L. Soilán-Duarte, G. Caballero-Mairesse, C. Grabowski-Ocampos y G. Enciso-Maldonado, «Manual De Nematología Agrícola,»

[8] R. Piedra Naranjo, «Guía De Muestreo De Nematodos Fitoparásitos En Cultivos Agrícolas,» Instituto Nacional de Innovación y Transferencia en Tecnología Agropecuaria (INTA- COSTA RICA), inf. téc., 2015.

[9] S. Sánchez-Moreno y M. Talavera, «Los nematodos como indicadores ambientales en agroecosistemas,» *Ecosistemas*, vol. 22, n.º 1, págs. 50-55, 2013.

[10] A. Barrera, C. Guindel, F. García y D. Martín, «Análisis, evaluación e implementación de algoritmos de segmentación semántica para su aplicación en vehículos inteligentes,» *Actas de las XXXIX Jornadas de Automática, Badajoz, 5-7 de Septiembre*, 2018.

[11] G. D. Corzo Ussa, «Segmentación semántica para imágenes de paisajes tropicales,» 2017.

[12] P. Alvarado-Moya, «Programa interinstitucional de investigación en biodiversidad y ecología de organismos de suelo, con énfasis en sistemas de producción limpia y control biológico,» Instituto Tecnológico de Costa Rica, inf. téc., 2009.

[13] J. E. Marín Hernández, «Ubicación de un nematodo en imágenes digitales utilizando modelos activos de forma,» Lic. tesis, Instituto Tecnológico de Costa Rica, Cartago, 2009.

[14] C. M. Grüner Monzón, «Active Dictionary Models: A framework for non-linear shape modeling,» Tesis de mtría., Instituto Tecnológico de Costa Rica, Cartago, 2015.

[15] J. Arroyo Hernández, «Modelo evolutivo de forma para el ajuste de organismos vermiformes en imágenes digitales,» Tesis doct., Instituto Tecnológico de Costa Rica, Cartago, 2021.

[16] O. J. Ramírez Morales, «Estrategia para la síntesis de imágenes de microscopía óptica digital utilizando técnicas de aprendizaje profundo,» Lic. tesis, Instituto Tecnológico de Costa Rica, Cartago, 2021.

[17] M. Zolla Salazar, «Aumento de datos utilizando redes generativas para mejorar el reconocimiento y segmentación de imágenes de microscopía óptica,» Lic. tesis, Instituto Tecnológico de Costa Rica, Cartago, 2022.

[18] O. E. Gómez Sólorzano, «Segmentación de nematodos en imágenes digitales usando redes neuronales,» Tesis de mtría., Instituto Tecnológico de Costa Rica, Cartago, 2009.

[19] M. Á. Taylor López, «Estrategia de ajuste de formas explícitas a imágenes con estructuras vermiformes.,» Lic. tesis, Instituto Tecnológico de Costa Rica, Cartago, 2017.

[20] J. Jiménez Chavarría, «SegNema: Nematode segmentation strategy in digital microscopy images using deep learning and shape models,» Tesis de mtría., Instituto Tecnológico de Costa Rica, Cartago, 2019.

[21] R. M. Villanueva, «Detección de objetos por computador,» 2017.

[22] D. F. Lopez Lozano y col., «Metodología de segmentación de instancias multiplano para la detección de Tromboembolismo Pulmonar,» 2021.

[23] T.-Y. Lin, M. Maire, S. Belongie y col., *Microsoft COCO: Common Objects in Context*, 2014. DOI: 10.48550/ARXIV.1405.0312.

[24] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff y H. Adam, «Encoder-decoder with atrous separable convolution for semantic image segmentation,» en *Proceedings of the European conference on computer vision (ECCV)*, 2018, págs. 801-818.

[25] L. Yan, J. Huang, H. Xie, P. Wei y Z. Gao, «Efficient depth fusion transformer for aerial image semantic segmentation,» *Remote Sensing*, vol. 14, n.º 5, pág. 1294, 2022.

[26] O. A. Soto-Orozco, A. D. Corral-Sáenz, C. E. Rojo-González y J. A. Ramírez-Quintana, «Análisis del desempeño de redes neuronales profundas para segmentación semántica en hardware limitado,» *ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica*, vol. 8, n.º 2, 2019.

[27] D. Marín Soto, «Segmentación de células mediante técnicas de Procesamiento Digital de Imágenes para el rastreo de células cancerosas,» Lic. tesis, Instituto Tecnológico de Costa Rica, Cartago, 2018.

[28] S. Zheng, J. Lu, H. Zhao y col., «Rethinking Semantic Segmentation from a Sequence-to-Sequence Perspective with Transformers,» *arXiv preprint arXiv:2012.15840*, 2020.

[29] L. Huang, Y. Yuan, J. Guo, C. Zhang, X. Chen y J. Wang, «Interlaced Sparse Self-Attention for Semantic Segmentation,» *arXiv preprint arXiv:1907.12273*, 2019.

[30] C. Yu, C. Gao, J. Wang, G. Yu, C. Shen y N. Sang, «Bisenet v2: Bilateral network with guided aggregation for real-time semantic segmentation,» *International Journal of Computer Vision*, págs. 1-18, 2021.

[31] R. Ranftl, A. Bochkovskiy y V. Koltun, «Vision Transformers for Dense Prediction,» *ArXiv preprint*, 2021.

[32] Z. Liu, L. Jin, J. Chen y col., «A survey on applications of deep learning in microscopy image analysis,» *Computers in Biology and Medicine*, vol. 134, pág. 104 523, 2021, ISSN: 0010-4825. DOI: https://doi.org/10.1016/j.compbiomed.2021.104523.

[33] P. I. Orellana Rueda, «Segmentación Semántica y reconocimiento de lugares usando características CNN preentrenadas,» Tesis de mtría., Universidad de Chile, Santiago de Chile, 2019.

[34] O. Ronneberger, P. Fischer y T. Brox, *U-Net: Convolutional Networks for Biomedical Image Segmentation*, 2015. DOI: 10.48550/ARXIV.1505.04597.

[35] M. Contributors. «MMSegmentation: OpenMMLab Semantic Segmentation Toolbox and Benchmark.» (2020), dirección: https://github.com/open-mmlab/mmsegmentation (visitado 14-11-2022).

[36] A. Liu y Z. Wang, *CV 3315 Is All You Need : Semantic Segmentation Competition*, 2022. DOI: 10.48550/ARXIV.2206.12571.

[37] Y. Yuan, X. Chen y J. Wang, «Object-Contextual Representations for Semantic Segmentation,» 2020.

[38] M. Yin, Z. Yao, Y. Cao y col., *Disentangled Non-Local Neural Networks*, 2020.

[39] A. Kirillov, Y. Wu, K. He y R. Girshick, «Pointrend: Image segmentation as rendering,» en *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, págs. 9799-9808.

[40] T. Wu, S. Tang, R. Zhang, J. Cao e Y. Zhang, «Cgnet: A light-weight context guided network for semantic segmentation,» *IEEE Transactions on Image Processing*, vol. 30, págs. 1169-1179, 2020.

[41] M. Fan, S. Lai, J. Huang y col., «Rethinking BiSeNet For Real-time Semantic Segmentation,» en *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, págs. 9716-9725.

[42] R. Strudel, R. Garcia, I. Laptev y C. Schmid, «Segmenter: Transformer for semantic segmentation,» en *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, págs. 7262-7272.

[43] W. Zhang, J. Pang, K. Chen y C. C. Loy, «K-Net: Towards Unified Image Segmentation,» en *NeurIPS*, 2021.

[44] E. Xie, W. Wang, Z. Yu, A. Anandkumar, J. M. Alvarez y P. Luo, «SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers,» *arXiv preprint arXiv:2105.15203*, 2021.

[45] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid y S. Savarese, *Generalized Intersection over Union: A Metric and A Loss for Bounding Box Regression*, 2019. DOI: 10.48550/ARXIV.1902.09630.

[46] D. J. Gómez Ortega y col., «Diseño, implementación y evaluación de una estrategia de detección de eventos acústicos,» 2021.

[47] R. A. Fuentes Zúñiga, «Prototipo de clasificador multiclase para relatos médicos,» 2021.

[48] V. Nanda, S. V. Belure y O. M. Shir, «Searching for the Pareto frontier in multi-objective protein design,» *Biophysical reviews*, vol. 9, n.º 4, págs. 339-344, 2017.

[49] S. B. Alemán Viteri, «Análisis de sentimientos para Twitter con Vader y TextBlob,» *Revista Odigos*, vol. 2, n.º 3, págs. 9-25, 2021.

[50] C. Yu, J. Wang, C. Peng, C. Gao, G. Yu y N. Sang, «Bisenet: Bilateral segmentation network for real-time semantic segmentation,» en *Proceedings of the European conference on computer vision (ECCV)*, 2018, págs. 325-341.

[51] W. Wang, E. Xie, X. Li y col., «Pyramid vision transformer: A versatile backbone for dense prediction without convolutions.,» *arXiv*, págs. 2, 3, 4, 2021.

[52] C. J. Reimúndez, «Estudio de rendimiento en GPU,» Tesis de mtría., Universidad Complutense de Madrid, Madrid, España, 2010.

[53] «NVIDIA A100 Tensor Core GPU Architecture,» Nvidia, Technical Report, 2020.

[54] «Advancing New Discoveries with a GPU-Powered Supercomputer,» Dresden University of Technology, Technical Report, 2015.

[55] «Nvidia Tesla K80 by PNY,» PNY Technologies Europe, Datasheet, 2014.

[56] «Tesla K80 GPU Accelerator Board Specification,» Nvidia, Datasheet, ver. 05, 2015.

[57] «NVIDIA A100 40GB PCIe GPU Accelerator,» Nvidia, Datasheet, ver. 03, 2020.

[58] «NVIDIA V100 TENSOR CORE GPU,» Nvidia, Datasheet, 2020.

[59] J. Zhang, C. Li, M. M. Rahaman y col., «A comprehensive review of image analysis methods for microorganism counting: from classical image processing to deep learning approaches,» 2021. DOI: 10.1007/s10462-021-10082-4.

[60] L. Chen, M. Strauch, M. Daub, M. Jansen, H.-G. Luigs y D. Merhof, «Instance Segmentation of Nematode Cysts in Microscopic Images of Soil Samples,» en *International Engineering in Medicine and Biology Conference (EMBC)*, 2019.

[61] R. Shamir, Y. Duchin, J. Kim, G. Sapiro y N. Harel, «Continuous Dice Coefficient: a Method for Evaluating Probabilistic Segmentations,» abr. de 2018. DOI: 10.1101/306977.

[62] L. Chen, M. Strauch, M. Daub y col., «A CNN Framework Based on Line Annotations for Detecting Nematodes in Microscopic Images,» en *IEEE International Symposium on Biomedical Imaging (ISBI)*, 2020.

[63] M. TI, C. AL, L.-F. J y col., «High-throughput screen for novel antimicrobials using a whole animal infection model,» vol. 4, n.º 7, págs. 527-533, 2009. DOI: 10.1021/cb900084v.

[64] J. Uhlemann, O. Cawley y T. Kakouli-Duarte, «Nematode Identification using Artificial Neural Networks,» en *DeLTA*, 2020.

[65] F. Chollet y col., *Keras*, https://keras.io, 2015.

Apéndice A

Modelos investigados

En la tabla A.1, se desglosan los criterios de los 11 modelos investigados.

Tabla A.1: Modelos de segmentación semántica investigados. Los datos marcados con * son obtenidos de un *script* experimental de la herramienta. Fuente: [28]-[31], [37]-[42], [44]

Modelo	Capacidad computacional	Backbone	Decoder	Datasets usados	mIoU	Params (M)	GFlops
Segformer	8 Tesla V100	Mix Transfomer MiT-B0	Lightweight All-MLP	ADE20K	37.4	3.8	8.4
				Cityscapes	76.2		125.5
				COCO-Stuff	35.6		8.4
Segformer	8 Tesla V100	Mix Transfomer MiT-B2	Lightweight All-MLP	ADE20K	46.5	27.5	62.4
				Cityscapes	81		717.1
				COCO-Stuff	44.6		62.4
Segformer	8 Tesla V100	Mix Transfomer MiT-B3	Lightweight All-MLP	ADE20K	49.4	47.3	79
				Cityscapes	81.7		962.9
				COCO-Stuff	45.5		79
Segformer	8 Tesla V100	Mix Transfomer MiT-B5	Lightweight All-MLP	ADE20K	51	84.7	183.3
				Cityscapes	82.4		1460.4
				COCO-Stuff	46.7		111.6
Segmenter	V100 GPU	MiT with patch tokens: Tiny	Mask Transformer (DETR)	ADE20K	39.03	6	-
Segmenter	V100 GPU	MiT with patch tokens: Small	Mask Transformer (DETR)	ADE20K	45.37	22	-
Segmenter	V100 GPU	MiT with patch tokens: Base	Mask Transformer (DETR)	ADE20K	48.48	86	-
Segmenter	V100 GPU	MiT with patch tokens: Large	Mask Transformer (DETR)	ADE20K	51.3	307	-
DPT	Nvidia RTX 2080 GPU	ViT bag-of-words	Convolutional Decoder Three-stage Reassemble	ADE20K	49.02	343	-
				Pascal Context	60.46		
SETR	-	ViT with image sequentialization	Naive	ADE20K	48.06	305.7	-
				Pascal Context	52.89		
				Cityscapes	-		
SETR	-	ViT with image sequentialization	Progressive (PUP)	ADE20K	48.58	318.3	-
				Pascal Context	54.4		
				Cityscapes	79.34		
SETR	-	ViT with image sequentialization	MLA	ADE20K	48.64	310.6	-
				Pascal Context	54.87		
				Cityscapes	-		
STDC1	NVIDIA GTX 1080Ti GPU	U-net architecture with STDC module	Detail Guidance for low-level layers	Cityscapes	75.3	8.44	813 [M]
				CamVid	73		
STDC2	NVIDIA GTX 1080Ti GPU	U-net architecture with STDC module	Detail Guidance for low-level layers	Cityscapes	76.8	12.47	1446 [M]
				CamVid	73.9		
BiSeNetV2	NVIDIA GeForce GTX 1080Ti card	BiSeNetV2	-	Cityscapes	73.4	14.8*	21.15
				CamVid	72.4		
				COCO-Stuff	87.9		
ISANet	4xP100 GPUs	ResNet-101 Self-Attention	-	Cityscapes	79.49	-	386
				ADE20K	45.04		
				Pascal Context	54.1		
				COCO-Stuff	39.2		
CGNet	2× V100 GPU	M3N21 with a Context Guided block	-	Cityscapes	64.8	496.32 [k]*	6
				CamVid	65.6		
CGNet	2× V100 GPU	ResNet-101 with a Context Guided block	-	Cityscapes	68.27	496.32 [k]*	27.46 *
PointRend + DeeplabV3	-	ResNet-101 with a PointRend module	-	Cityscapes	78.4	47.71*	521.69*
PointRend + SemanticFPN	-	ResNet-101 with a PointRend module	-	Cityscapes	78.5	47.71*	521.69*
PointRend	-	ResNet-101 with a PointRend module	-	Cityscapes	79.97	47.71*	521.69*
DNLNet	4-8 GPUs	ResNet-101 with DNL block	-	Cityscapes	82	71.485	765.16
				ADE20K	45.97		
				Pascal Context	56.23		
OCRNet	P40 GPU	HRNet-W48 with OCR module	Cross-attention module	Cityscapes	82.4	10.5	340
				ADE20K	45.66		
				Pascal Context	56.2		
				COCO-Stuff	40.5		
OCRNet	P40 GPU	ResNet-101 with OCR module	Cross-attention module	Cityscapes	81.8	-	-
				ADE20K	45.28		
				Pascal Context	54.8		
				COCO-Stuff	39.5		
OCRNet	P40 GPU	HRNet-W18 small with OCR module	Cross-attention module	Cityscapes	74.3	6.38*	353.47*
				ADE20K	35.06		