UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

EVALUACIÓN DE *VISION TRANSFORMERS* Y *SEGMENT ANYTHING MODEL* PARA SEGMENTACIÓN SEMÁNTICA EN EL MONITOREO DE ARBOLADO URBANO

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

SERGIO IVÁN BAEZ MENA

PROFESOR GUÍA:
JOSÉ MANUEL SAAVEDRA RONDO

MIEMBROS DE LA COMISIÓN:
BENJAMÍN BUSTOS CÁRDENAS
MAURICIO CERDA VILLABLANCA

Este trabajo ha sido parcialmente financiado por FONDECYT 21I10360

SANTIAGO DE CHILE
2024

Resumen

La segmentación semántica es una de las principales tareas de visión cuyo objetivo es la clasificación de cada pixel en una imagen dentro de una categoría. Esta tarea está siendo aplicada en distintas áreas de la industria, teniendo un incipiente uso en el monitoreo de arbolado urbano que consiste en el censo y medición del estado de salud de los árboles pertenecientes a una ciudad.

En la presente memoria se persiguen dos objetivos. El primero consiste en la evaluación de distintos tipos de *vision transformers* sobre un nuevo dataset de arbolado urbano denominado TreeSegmentation, perteneciente al proyecto Arbocensus. En particular, los modelos evaluados pertenecen a las categorías: escala uniforme, multiescala y clasificación de máscaras. Por su parte, el segundo objetivo es la evaluación del primer *foundation model* para segmentación *Segment Anything Model* (SAM). Más específicamente, se busca transferir las características desde su *image encoder*.

Los experimentos realizados arrojaron que el mejor desempeño obtenido corresponde al modelo de clasificación de máscaras Mask2Former con 87.21 mIoU. Otro resultado relevante es la ventaja mostrada por los modelos multiescala por sobre los de escala uniforme evaluados, estando los primeros arriba por 5 mIoU en una de las categorías. Por último, el uso del *image encoder* de SAM mostró buenos resultados, alcanzando un desempeño de 86.60 mIoU con el modelo SAM-Mask. Sin embargo, para determinar la magnitud de los beneficios se requiren más experimentos.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Contexto

La segmentación semántica es una de las principales tareas de visión por computadora. Esta tiene como objetivo particionar las imágenes en regiones que delimiten las distintas categorías de objetos relevantes para un cierto contexto (Figura 1.1). Al igual que en las demás tareas de visión, la introducción de los modelos de aprendizaje profundo y, en particular, los avances de los últimos años ha permitido diversas aplicaciones en la industria [12, 22].

Figura 1.1: Representación de la tarea de segmentación semántica. Cada categoría de objeto está delimitada por una misma región (color). Fuente: [15]

Un área donde se ha comenzado a aplicar este tipo de modelos es en el monitoreo de arbolado urbano. Este consiste en el censo y medición del estado de salud de los árboles presentes en una ciudad. La realización de este tipo de monitoreo trae consigo varios beneficios tales como una mejor distribución de los recursos, la detección de árboles en mal estado y una mejor planificación urbana.

Sin embargo, hay distintas tareas implicadas en el monitoreo de arbolado urbano que son costosas. El censo requiere de una persona capacitada que vaya árbol tras árbol recopilando datos como la especie y localización. Por su parte, monitorear la condición de salud de un

árbol requiere mediciones físicas o apreciaciones visuales que permitan predecir, por ejemplo, el riesgo de caída.

Específicamente, en el área de segmentación se han propuestos distintos trabajos para avanzar en soluciones que disminuyan estos costos, ya sea propuestas relacionados directamente con la segmentación de las estructuras del árbol [5, 11] o con el desarrollo de nuevos datasets [25]. Sin embargo, estos no profundizan en la evaluación de modelos pertenecientes al estado del arte.

Actualmente, el estado del arte está constituido por los *vision transformers*. Estos son modelos que adaptan la arquitectura *Transformer* [20] para su aplicación en las distintas tareas de visión. Además, en la búsqueda de mejores adaptaciones se han desarrollado distintos tipos de modelos que incorporan progresivamente estrategias para mejorar el desempeño y el costo computacional.

Dentro de estos últimos avances está el primer *foundation model* para segmentación de imágenes, denominado *Segment Anything Model* (SAM) [12]. Este modelo permite segmentar cualquier nueva imagen u objeto presente en ella, al indicarse mediante una entrada como un punto o *bounding box*. El desarrollo de este modelo significa un gran avance para segmentación y permitirá un conjunto de nuevas aplicaciones.

1.2. Relevancia del problema

El presente trabajo de memoria se guía por dos temáticas centrales. La primera es la falta de evaluaciones en el área de arbolado urbano y la necesidad de conocer el desempeño de distintos tipos de modelos para su incorporación en herramientas de inteligencia artificial. Concretamente, este trabajo se concentra en estudiar distintos tipos de *vision transformers* sobre un nuevo dataset de segmentación denominado TreeSegmentation.

Como se menciona anteriormente, SAM significa un gran avance para la tarea de segmentación. Para el entrenamiento de este modelo fue necesario la creación de un nuevo dataset de gran escala denominado SA-1B [12], constituido por 11M de imágenes y 1B de máscaras. El segundo tema consiste en transferir las características aprendidas sobre SA-1B por el *image encoder* de SAM. Esto motivado por la gran escala del dataset, destacando principalmente su número de máscaras.

1.3. Objetivos

1.3.1. Objetivo general

El objetivo general del presente trabajo de memoria es evaluar el desempeño de distintos *vision transformers* y el impacto de la transferencia de características desde *Segment Anything Model* en el área de monitoreo urbano.

1.3.2. Objetivos específicos

El presente trabajo consta de los siguientes objetivos específicos:

1. Preparar el dataset TreeSegmentation.
2. Definir el modelo(s) convolucional(es) a utilizar como referencia.
3. Definir los *vision transformers* a ser evaluados.
4. Definir los modelo(s) que incorporarán el *image encoder* de *Segment Anything Model*.
5. Entrenar, evaluar y comparar los distintos modelos sobre el dataset TreeSegmentation.

Capítulo 2

Marco teórico y estado del arte

En este capítulo se introducen los conceptos y el estado del arte relevante para el presente trabajo de memoria. En la Sección 1.1 se define formalmente la tarea de segmentación semántica. Las Secciones 2.2 y 2.3 presentan la arquitectura convolucional y *transformers*, respectivamente. Por su parte, las Secciones 2.4 y 2.5 introducen el conjunto de modelos denominados *vision transformers* y sus distintos tipos. Finalmente, se destina la Sección 2.6 para explicar los *foundation models* y la Sección 2.7 para definir las métricas usadas.

2.1. Segmentación semántica

La segmentación semántica es un problema de visión por computadora cuyo objetivo es clasificar cada pixel de una imagen dentro de una categoría, sin hacer distinción entre instancias particulares. Esta tarea es llevada a cabo mediante dos estrategias: clasificación por pixel y clasificación de máscaras.

La clasificación por pixel busca generar un tensor de salida $y \in \mathbb{R}^{C \times H \times W}$ con la misma resolución espacial que la imagen de entrada, donde C es el número de clases. De esta manera, cada pixel tiene asociada una distribución de probabilidad a través de las clases. Por su parte, la clasificación de máscaras tiene como objetivo generar N tuplas $y = \{(p_i, m_i)\}_{i=1}^{N}$, donde m_i es una máscara binaria de dimensión $H \times W$ y p_i una distribución de probabilidad sobre las C clases.

Independiente del tipo de estrategia empleada, la segmentación semántica y, en general, las tareas de visión presentan tres desafíos principalmente: (1) la presencia de objetos en múltiples escalas, (2) la obtención de campos receptivos de gran tamaño capaces de modelar contexto y (3) la pérdida de información de bajo nivel a través de las capas.

2.2. Redes convolucionales

Las redes convolucionales (o CNN por sus siglas en inglés) han desarrollado un conjunto de estrategias para superar los desafíos asociados a segmentación. Uno de los primeros trabajos relevantes es U-Net [18]. Este es un modelo con estructura *encoder-decoder* que utiliza conexiones residuales y *upsampling* para fusionar mapas de características de distintos niveles y, de esta forma, mitigar la pérdida de información de bajo nivel.

Una desventaja de las CNN es que para aumentar el tamaño de los campos receptivos deben disminuir progresivamente la resolución de la imagen. Esto dificulta la tarea de segmentación, ya que se debe realizar una predicción a nivel de pixel. Para resolver esto, distintos modelos incorporan *Atrous Convolution* [26], representado en la Figura 2.1. Este tipo de convolución permite aumentar el tamaño de los campos receptivos sin reducir la resolución, ni aumentar el número de parámetros. Los modelos de la familia DeepLab incorpora esta idea en las últimas capas de sus *backbones* [1, 2].

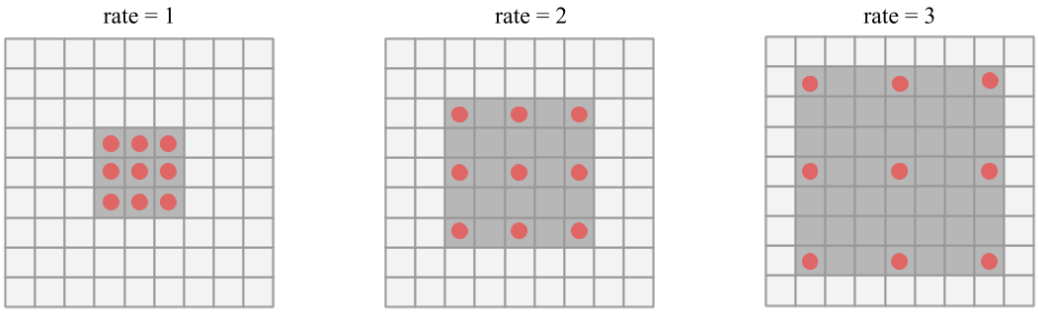

Figura 2.1: Representación de *atrous convolution* con un kernel 3×3 (puntos rojos) con distintos *rates*. El *rate* indica el espaciamiento entre los pesos del kernel. A medida que aumenta el *rate* se va incrementando el campo receptivo (zona sombreada), sin incrementar el número de parámetros.

El último modelo de esta familia es DeepLabv3+ [2]. Este emplea una estructura *encoder-decoder* y la componente *Atrous Spatial Pyramid Pooling* (ASPP) [1]. Esta última usa capas paralelas de *atrous convolution* con distintos *rates* con el objetivo de (1) incrementar el tamaño de los campos receptivos sin disminuir la resolución del mapa de características y (2) capturar información multi-escala. La Figura 2.2 presenta una esquema general de este modelo.

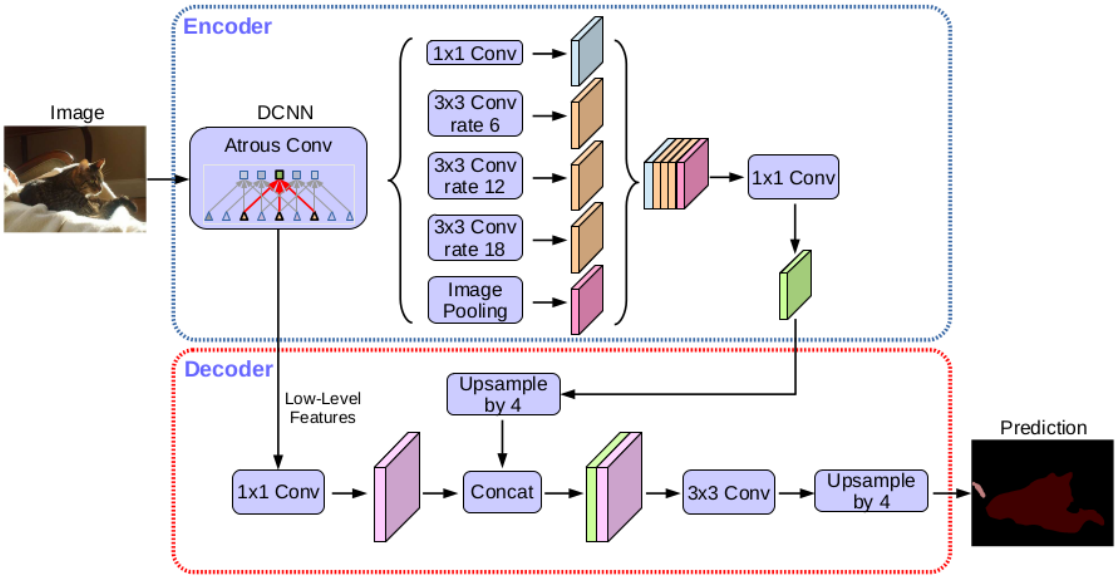

Figura 2.2: Esquema general de DeepLabv3+. Fuente: [2]

2.3. *Transformer*

La arquitectura *Transformer* [20] es un tipo de red –propuesta originalmente para el área de Procesamiento de Lenguage Natural (o NLP por sus siglas en inglés)– diseñada para mapear una secuencia de vectores de entrada $x = (x_1, ..., x_n)$ hacia una secuencia de salida $y = (y_1, ..., y_m)$ mediante la aplicación de sucesivos bloques que incorporan las operaciones *cross-attention* y *self-attention*.

2.3.1. *Cross-attention*

La operación de *cross-attention* estándar con una conexión residual está dada por la siguiente ecuación [3]:

$$\mathbf{X}_l = \text{softmax}(\mathbf{Q}\mathbf{K}^T)\mathbf{V} + \mathbf{X}_{l-1} \tag{2.1}$$

En esta ecuación $\mathbf{X}_l \in \mathbb{R}^{N \times C}$ representa una secuencia de vectores para una capa de índice l. En la operación de *cross-attention* la matriz $\mathbf{Q} \in \mathbb{R}^{N \times C}$ se obtiene mediante una transformación lineal de la secuencia $\mathbf{X}_{l-1}$, mientras que las matrices $\mathbf{K}, \mathbf{V} \in \mathbb{R}^{N' \times C}$ se obtienen linealmente de una secuencia diferente $\mathbf{Z}_{l-1} \in \mathbb{R}^{N' \times C}$.

En problemas de visión comúnmente la secuencia $\mathbf{X}_{l-1} \in \mathbb{R}^{N \times C}$ representa N vectores entrenables denominados *embeddings*, mientras que la secuencia $\mathbf{Z}_{l-1} \in \mathbb{R}^{N' \times C}$ a los *image embeddings*, o viceversa. Estos últimos son representaciones de las imágenes con menor dimensión espacial pero con mayor valor semántico. En este caso $N' = h \times w$, donde (h, w) es la resolución de los *image embeddings*.

2.3.2. *Self-attention*

La operación *self-attention* es un caso particular de la Ecuación 2.1 donde las matrices $\mathbf{Q}$, $\mathbf{K}$ y $\mathbf{V} \in \mathbb{R}^{N \times C}$ se obtienen de la misma secuencia de vectores $\mathbf{X}_{l-1} \in \mathbb{R}^{N \times C}$, cada una mediante una transformación lineal independiente. Por medio de esta, cada vector de la secuencia $\mathbf{X}_{l-1}$ incorpora la información de todos los demás para computar el vector de su misma posición en $\mathbf{X}_l$.

Debido a que cada vector de la secuencia $\mathbf{X}_{l-1}$ atiende a todos los demás, esta operación tiene un costo computacional de $O(N^2)$, es decir, cuadrático con respecto al largo de la secuencia. Esto es poco abordable para las tareas de visión por computadora donde $N = h \times w$, siendo $(h,\, w)$ la resolución del mapa de características sobre el que se computa.

2.3.3. *Multi-head attention*

Independientemente si se hace uso de *cross-attention* o *self-attention*, los distintos modelos no computan estas operaciones una sola vez, sino que las computan paralelamente mediante *heads*. La Ecuación 2.2 muestra el computo realizado por una *multi-head self-attention* (MHSA):

$$\begin{aligned} \text{Multi-Head}(\mathbf{x}) &= \text{concat}(\text{head}_1(\mathbf{x}),\ ...,\ \text{head}_\text{N}(\mathbf{x}))\,\mathbf{W}_O \\ \text{head}_i(\mathbf{x}) &= \text{self-attention}(\mathbf{x}\mathbf{W}_i^\text{Q},\ \mathbf{x}\mathbf{W}_i^\text{K},\ \mathbf{x}\mathbf{W}_i^\text{V}) \end{aligned} \tag{2.2}$$

Donde $\mathbf{x} \in \mathbb{R}^{N \times C}$, $\mathbf{W}_i^\text{Q} \in \mathbb{R}^{C \times C'}$, $\mathbf{W}_i^\text{K} \in \mathbb{R}^{C \times C'}$, $\mathbf{W}_i^\text{V} \in \mathbb{R}^{C \times C'}$.

2.4. *Vision transformers*

Se denominan *vision transformers* aquellos modelos que incorporan y ajustan la arquitectura *transformer* al área de visión por computadora. Un grupo importante de este tipo de modelos son los *vision transformers backbones*.

2.4.1. *Vision transformers backbones*

Los *vision transformers backbones* [6, 16, 21] son *backbones* de propósito general que basan su estructura en bloques MHSA o en alguna de sus variantes. Estos modelos permiten emplear estos bloques directamente sobre las imágenes para la extracción de características y proponen distintas estrategias para superar el costo computacional y la resolución uniforme asociada a *self-attention*.

Vision Transformer (ViT)

El modelo *Vision Transformer* (ViT) [7] es el trabajo pionero en el uso de la arquitectura *transformer* en el área de visión (Figura 2.3). Este modelo lleva a cabo la clasificación de imágenes empleando un *encoder* estándar de *transformer* [20] sobre una secuencia construida directamente a partir de la imagen, denominada *patch embeddings*. Esta componente constituye la base de los *vision transformers*.

La construcción de esta secuencia comienza con la división de la imagen en ventanas no superpuestas de tamaño $P \times P$. Luego, las ventanas son redimensionadas y concatenadas para obtener la secuencia de dimensión $(N \times P^2C)$, donde N es el número de ventanas y C el número de canales de la imagen. Finalmente, se aplica una proyección lineal para obtener la secuencia de *patch embeddings*.

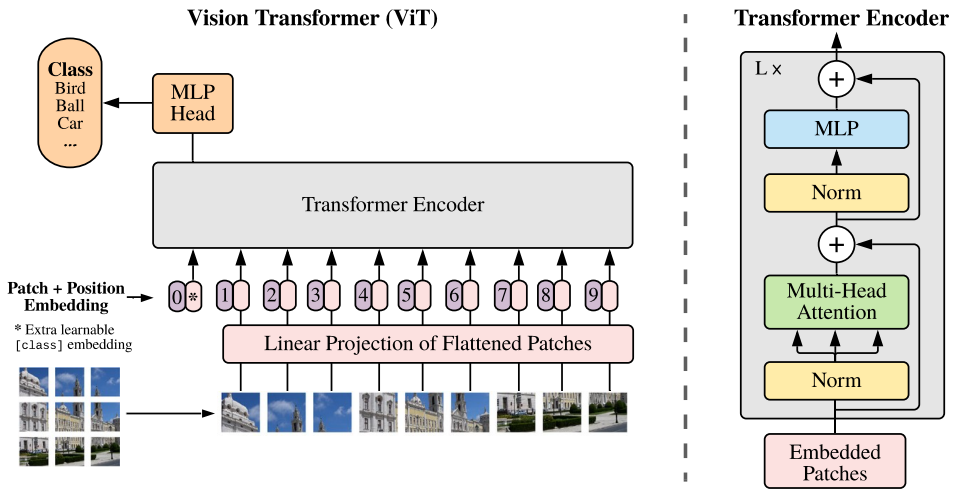

Figura 2.3: Esquema general de ViT. Fuente: [7]

Swin Transformer

Swin Transformer [16] es un *backbone* jerárquico. Esto significa que reduce la resolución espacial del mapa de características progresivamente a través de las capas. La principal característica de este modelo es que limita la computación de *self-attention* a ventanas no superpuestas, reduciendo el costo computacional asociado. Un esquema general de la versión de menor capacidad de Swin Transformer se muestra en la Figura 2.4.

La capacidad del modelo para producir mapas de diversa escala radica en las capas *patch merging*. Estas capas concatenan los *patch embeddings* pertenecientes a una vecindad y reducen la dimensionalidad de los *embeddings* concatenados a la mitad mediante una proyección lineal. La propuesta original propone el uso de vecindades de 2×2, lo que reduce el número de *patches* por un factor de 4.

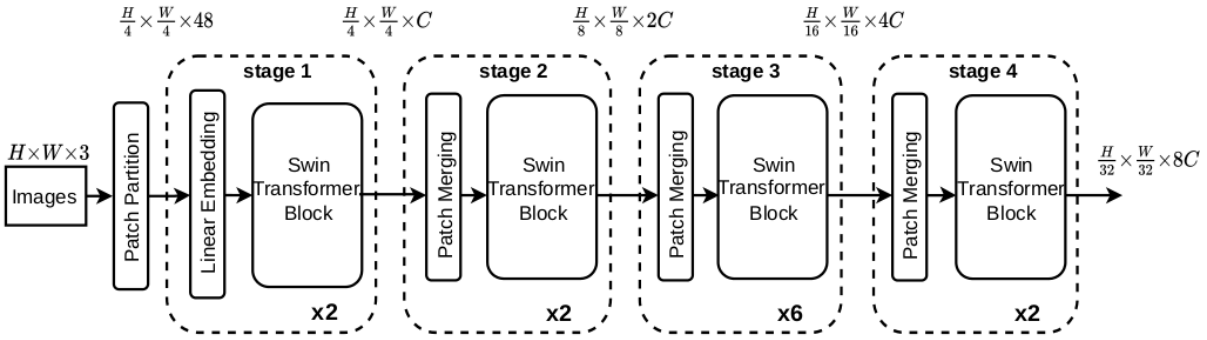

Figura 2.4: Esquema general de Swin-T. Esta es la versión de menor capacidad con $C = 96$. Fuente: [16]

2.5. *Vision transformers* para segmentación

2.5.1. *Vision transformers* de escala uniforme

Los primeros *vision transformers* que surgen para la tarea de segmentación son aquellos de escala uniforme [19, 27]. Estos modelos se caracterizan por tener un *backbone* basado en ViT [7] que mantiene la resolución del mapa de características a través de las capas. Estos presentan la ventaja de capturar las dependencias de largo alcance de manera eficaz pero con un mayor costo computacional asociado debido al uso de *self-attention* globalmente sobre los mapas de características.

Segment Transformer (SETR)

Un modelo representativo de esta categoría es SETR [27]. Este utiliza ViT como *encoder* para producir *image embeddings* de dimensión espacial $\frac{H}{16} \times \frac{W}{16}$. Posteriormente, estos son llevados a la dimensión original de la imagen $H \times W$ mediante un *decoder* que realiza una clasificación por pixel. En Zheng et al [27] se proponen tres *decoders*, siendo dos relevantes para el presente trabajo. En la Figura 2.5 se muestra un esquema del modelo y de ambos *decoders*.

El primero de estos se denomina *Progressive Upsampling* (PUP). Este consiste en un diseño simple de 4 capas convolucionales, cada una seguida de una interpolación que aumenta la dimensión espacial por un factor de 2. El segundo *decoder* es *Multi-Level feature Aggregation* (MLA). Este selecciona 4 mapas de características uniformemente distribuidos desde ViT. Luego, a cada mapa le aplica 3 convoluciones y una interpolación bilineal para aumentar su dimensión espacial por un factor de 4. Finalmente, estos son concatenados y llevados a la dimensión original de la imagen.

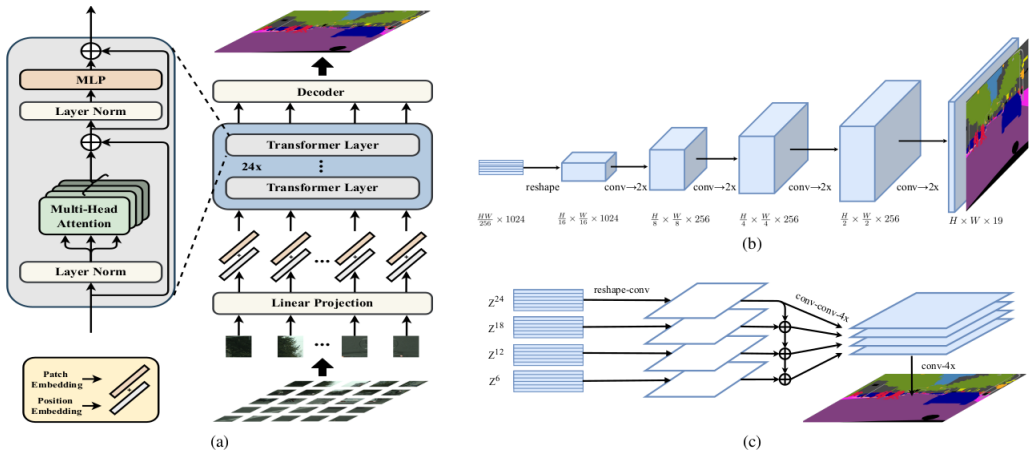

Figura 2.5: Esquema general de SETR. (a) Esquema del encoder de SETR. (b) Decoder *Progressive Upsampling* (PUP). (c) Decoder *Multi-Level feature Aggregation* (MLA). Fuente: [27]

2.5.2. *Vision transformers* multiescala

Una segunda categoría de modelos son los *vision transformer* multi-escala [8, 10, 23]. Estos se caracterizan por el uso de *backbones* con estructura jerárquica que reducen progresivamente la resolución de los mapas de características. Este tipo de modelos se ajusta de mejor manera a los problemas de visión, ya que genera características con distinto nivel de localidad en los mapas de las distintas capas.

SegFormer

Un modelo de este tipo que destaca por su sencillez y eficiencia es SegFormer [23], presentado en la Figura 2.6. En términos generales, el modelo divide la imagen de entrada en *patch embeddings* considerando ventanas de 4×4. Estos son usados como entrada para un *vision backbone* multiescala que construye mapas de resolución $\{\frac{1}{4}, \frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ con respecto a la original. Por último, estos distintos mapas son entregados al *decoder* que los redimensiona y fusiona para llevar a cabo una clasificación por pixel.

El componente más importante propuesto en Xie et al [23] es el *backbone Mix Transformer* (MiT). Este corresponde a una serie de distintos tamaños, desde MiT-B0 hasta MiT-B5, basados en ViT que incorporan distintas estrategias especialmente diseñadas para su uso en segmentación. Una de ellas consiste en el uso de una variante de *self-attention* denominada *spatial-reduction attention* [21] que disminuye el costo computacional.

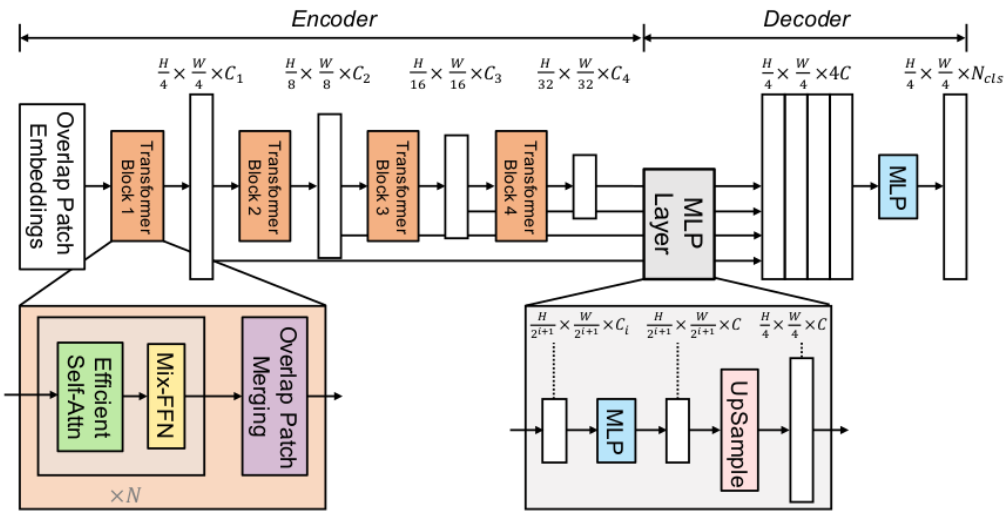

Figura 2.6: Esquema general de SegFormer. Se presenta el *encoder* multi-escala MiT y el *decoder* que concatena los mapas y lleva a cabo una predicción por pixel. Fuente: [23]

2.5.3. Clasificación de máscaras

Independientemente de los tipos mencionados, se han desarrollado modelos que optan por utilizar la estrategia de clasificación de máscaras para la tarea de segmentación (Sección 2.1). Estos trabajos son de gran interés debido a que presentan desempeños competitivos con respecto a los modelos que llevan a cabo una clasificación por pixel y, especialmente, por ser la base del *foundational model* para segmentación SAM [12].

MaskFormer

MaskFormer [4] es un modelo de clasificación de máscaras que –usando la definición dada en [4]– realiza la tarea de segmentación computando N pares $z = \{(p_i, m_i)\}_{i=1}^{N}$, donde m_i es una máscara binaria de dimensión $H \times W$ y p_i es una distribución de probabilidad sobre $K+1$ categorías, siendo K el número de clases asociados al dataset más un clase vacía $\emptyset$. Está última se le asigna a las máscaras que no presentan ningún objeto. El modelo se compone de tres módulos, siendo cada uno presentado en la Figura 2.7.

En el primer módulo, el modelo mediante un *backbone* genera un mapa de características $\mathcal{F} \in \mathbb{R}^{C_\mathcal{F} \times \frac{H}{32} \times \frac{W}{32}}$ a partir de la imagen de entrada, donde $C_\mathcal{F}$ depende del *backbone*. Luego, un *pixel decoder* aumenta gradualmente la resolución de $\mathcal{F}$ para generar *pixel embeddings* $\mathcal{E}_{pixel} \in \mathbb{R}^{C_\mathcal{E} \times H \times W}$ de igual dimensión espacial que la imagen. En el trabajo se propone un *pixel decoder* basado en la arquitectura FPN [14]. De esta forma, se integran los mapas de los distintos niveles del *backbone* para producir los *pixel embeddings*.

En un segundo módulo, un *transformer decoder* entrena N *embeddings* con la información del mapa de características $\mathcal{F}$. El uso de *cross-attention* permite que cada *embedding* incorpore información globalmente desde el mapa de características. Finalmente, un último módulo de segmentación genera por cada uno de los N *embeddings* una máscara binaria y la distribución de clases asociada.

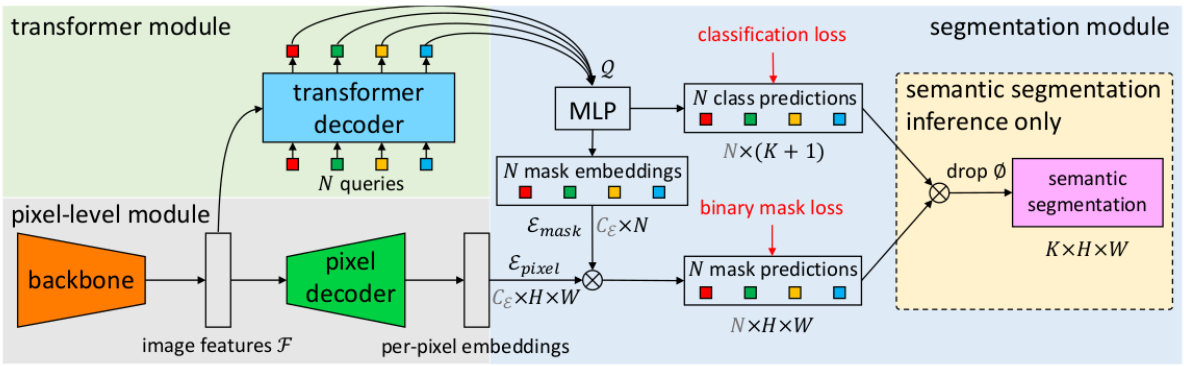

Figura 2.7: Esquema general de MaskFormer. Fuente: [4]

Mask2Former

El modelo Mask2Former [3] adopta la misma arquitectura general presentada en Cheng et al. [4] pero incorporando dos modificaciones importantes. La primera es el uso de *masked attention* [3] dentro del *transformer decoder*. Esta operación restringe las características del *image embedding* que pueden ser incorporadas por cada uno de los N *embeddings*. A diferencia de *cross-attention* donde se permite la atención sobre todo el mapa de características.

La segunda modificación corresponde a la utilización de mapas de distinta resolución en las capas del *transformer decoder*, tal como se observa en la Figura 2.8. A través de una FPN como *pixel decoder* se generan mapas con resolución $\{\frac{1}{8}, \frac{1}{16}, \frac{1}{32}\}$ de la imagen original. Estos son entregados sucesivamente a los 3 bloques que conforman el *decoder*. Este proceso a su vez se repite L veces, utilizando $L = 3$. Finalmente, para la predicción de las máscaras y clases se utiliza el mapa de resolución $\frac{H}{4} \times \frac{W}{4}$. Esta modificación tiene como objetivo mejorar el desempeño sobre objetos de menor escala.

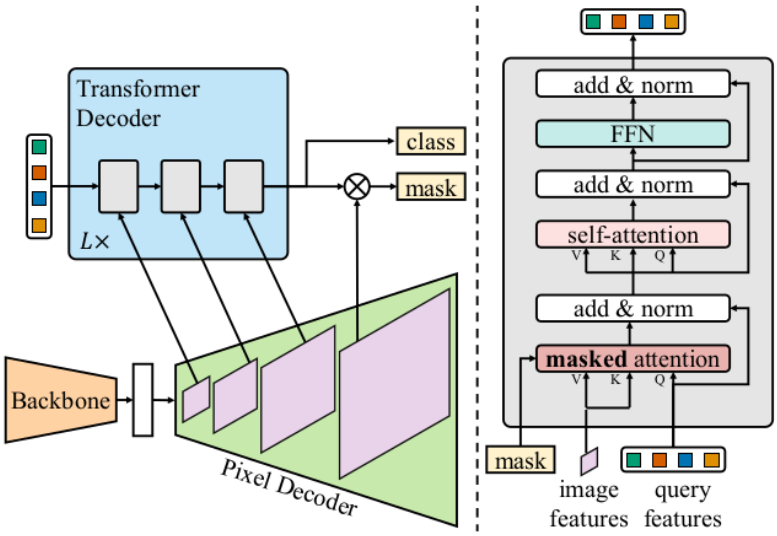

Figura 2.8: Esquema general de Mask2Former. Se puede observar la incorporación de los mapas con distinta resolución y el diseño de los bloques que incorporan *masked attention* en lugar de *cross-attention*. Fuente: [3]

2.6. *Foundation models*

Los denominados *foundation models* son modelos pre-entrenados sobre datasets de gran escala con la capacidad de ser adaptados para su uso en distintas tareas posteriores, frecuentemente con el uso de entradas. A diferencia de NLP, el desarrollo de este tipo de modelos en visión por computadora se ha visto retrasado por la falta de datasets etiquetados de gran escala. Algunos modelos como CLIP [17] son pre-entrenados con pares imagen-texto, pares con mayor disponibilidad en internet que máscara-imagen.

2.6.1. *Segment Anything Model*

El primer *foundation model* propuesto para segmentación se denomina *Segment Anything Model* (SAM) [12]. Este se compone de un *image encoder*, un *prompt encoder* y un *mask decoder*. El *image encoder* consiste en ViT pre-entrenado mediante la estrategia auto-supervisada MAE [9] con adaptaciones para su uso con imágenes de alta resolución [13]. Específicamente, se restringe el uso de *self-attention* a ventanas no superpuestas en todas las capas, excepto en 4 que están uniformemente distribuidas a través del *backbone*.

La adaptabilidad de SAM a distintas tareas se debe al uso de *prompts* que permiten indicar que objeto dentro de la imagen debe ser segmentado. Los tipos de *prompts* soportados son puntos, *bounding boxes*, máscaras y texto. Una vez procesados por el *prompt encoder*, estos en conjunto con los *image embeddings* son mapeados por el *mask decoder* a las máscaras de salida. Un esquema general de este proceso se muestra en la Figura 2.9.

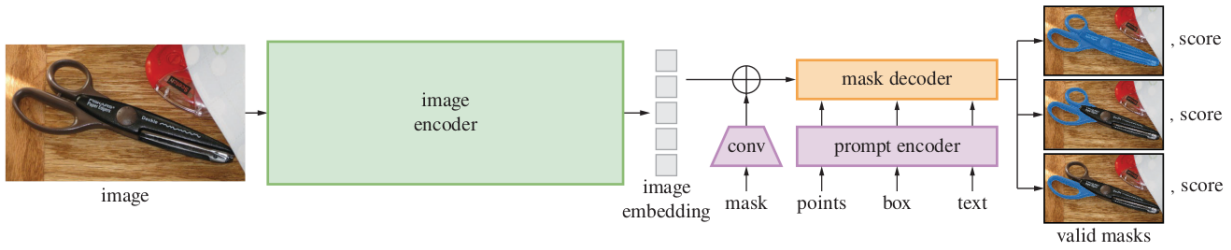

Figura 2.9: Esquema general de SAM. El *image encoder* genera *image embeddings* de dimensión 256 $\times \frac{H}{16} \times \frac{W}{16}$. Luego, un *prompt encoder* mapea un punto, *bounding box* o texto a un vector de dimensión 256 (*prompt embedding*). Finalmente, un *mask encoder* mapea los *image embeddings* y el *prompt embedding* a las máscaras de salida. Fuente: [12]

Como se menciona anteriormente, el otro componente relevante para el desarrollo de *foundation models* es la escala del dataset utilizado. SAM se entrena sobre el dataset SA-1B [12] que contiene 11M imágenes y un total de 1.1B máscaras, 400 veces más que cualquier otro dataset. Esta escala y diferencia con respecto a los demás datasets de segmentación hace interesante evaluar el impacto de transferir las características aprendidas desde SA-1B por parte del *image encoder*.

2.7. Métricas

Antes de introducir la métricas utilizadas, se deben definir dos conceptos relevantes. Para esto, es conveniente considerar tanto la predicción del modelo como la etiqueta asociada a una imagen como un tensor de dimensión espacial $H \times W$ donde cada posición tiene un valor entero que indica la clase de ese pixel en la imagen. Dada una determinada imagen y clase, los conceptos (representados en la Figura 2.10) se definen de la siguiente manera:

- **Área Intersección** $_{\text{clase}}$: número de pixeles categorizados como la clase respectiva tanto en la predicción del modelo como en la etiqueta.
- **Área Unión** $_{\text{clase}}$: número de pixeles categorizados como la clase respectiva en la predicción del modelo o en la etiqueta.

La primera de las métricas utilizadas se denomina *Intersection over Union* (IoU). Esta es calculada individualmente para cada clase. Dada una imagen, el valor de la métrica para una determinada clase se obtiene siguiendo la Ecuación 2.3. El valor final, calculado independientemente para cada clase, corresponde al promedio de los valores obtenidos para las imágenes pertenecientes al conjunto test o validación V (Ecuación 2.4).

$$\text{IoU}_{\text{clase}}^{\text{img}} = \frac{\text{Área Intersección}_{\text{clase}}}{\text{Área Unión}_{\text{clase}}} \tag{2.3}$$

$$\text{IoU}_{\text{clase}} = \frac{1}{|V|} \sum_{\text{img} \in V} \text{IoU}^{\text{img}}_{\text{clase}} \tag{2.4}$$

La segunda métrica corresponde a *Mean Intersection over Union* (mIoU). Esta métrica, de carácter más general, consiste en el promedio de $\text{IoU}_{\text{clase}}$ sobre el conjunto de todas las clases C, incluyendo la clase *background*. El cálculo de esta métrica se obtiene a través de la Ecuación 2.5.

$$\text{mIoU} = \frac{1}{|C|} \sum_{\text{clase} \in C} \text{IoU}_{\text{clase}} \tag{2.5}$$

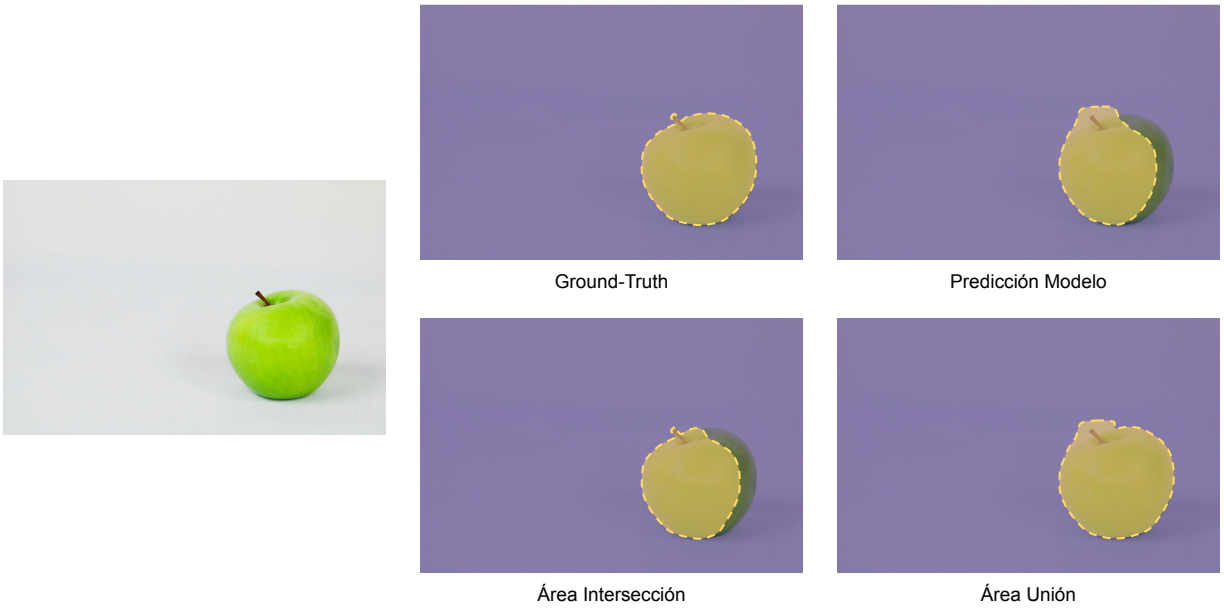

Figura 2.10: Representación gráfica de los conceptos *Área Intersección* y *Área Unión* para la clase *manzana*. En amarillo se muestran los pixeles categorizados como la clase *manzana* y en morado como *background*.

Capítulo 3

Desarrollo

En este capítulo se presenta el desarrollo previo a la realización de los experimentos. En la Sección 3.1 se explica el proceso de etiquetamiento y preparación del dataset principal TreeSegmentation. La Sección 3.2 presenta la preparación del dataset *Urban Street Tree*. Finalmente, la Sección 3.3 presenta el diseño y la selección de los modelos evaluados durante la experimentación.

3.1. TreeSegmentation

3.1.1. Especificaciones del dataset

TreeSegmentation es un dataset de segmentación constituido por imágenes de arbolado urbano perteneciente a distintas comunas de la ciudad de Santiago. Este pertenece al proyecto Arbocensus (Fondef-Idea) que busca la incorporación de inteligencia artificial en el censo y evaluación de la condición del arbolado. Una muestra de TreeSegmentation se puede observar en la Figura 3.1.

Figura 3.1: Muestra de imágenes del dataset TreeSegmentation.

Las principales especificaciones de la versión de TreeSegmentation utilizada en el presente trabajo de memoria son las siguientes:

- Las imágenes fueron capturadas desde la perspectiva de la calle mediante teléfonos móviles en distintos horarios y estaciones del año. Esto permite la presencia de árboles con y sin follaje, y bajo distintas condiciones de luz.
- Contiene 835 imágenes en total. Estas se particionan aleatoriamente en 710 y 125 imágenes para entrenamiento y validación, respectivamente.
- Presenta 2 categorías: *cortex* y *crown*. Las categorías poseen 880 y 554 número de instancias considerando ambas particiones, respectivamente. Esto se traduce en 798 imágenes con un único polígono de categoría *cortex*, es decir, con un único árbol etiquetado.

3.1.2. Proceso de etiquetamiento

Las etiquetas del dataset TreeSegmentation, descrito en la sección 3.1.1, se obtuvieron a partir de un proceso de re-etiquetamiento de una versión anterior del mismo. El primer paso dentro de este proceso fue la redefinición de las categorías:

1. La categoría *crown* se redefine como la parte del árbol que presenta follaje. La versión anterior del dataset definía *crown* como la parte superior del árbol con independencia de la presencia de follaje.
2. La categoría *cortex* se redefine como toda la estructura leñosa del árbol. La versión anterior del dataset definía *cortex* como el tronco y las ramas principales.

Una vez redefinidas las categorías, se realiza el proceso de re-etiquetamiento haciendo uso del software *supervisely*[1]. El protocolo utilizado se encuentra en el Anexo A.1. En términos generales, en esta etapa se llevan a cabo los siguientes pasos sobre las etiquetas de la versión anterior:

1. Se eliminan los polígonos de categoría *crown* si hay ausencia o escaso follaje, por ejemplo, en época otoñal.
2. Se ajustan los polígonos de categoría *cortex* lo más posible a la estructura leñosa del árbol y se incluye cualquier estructura (ramas o tronco) que no haya sido considerada en la versión anterior.
3. Se ajustan los polígonos de categoría *crown* lo más posible al follaje del árbol.
4. Se elimina cualquier elemento perteneciente al alumbrado público, letreros u otro similar de los polígonos.

[1] https://supervisely.com/

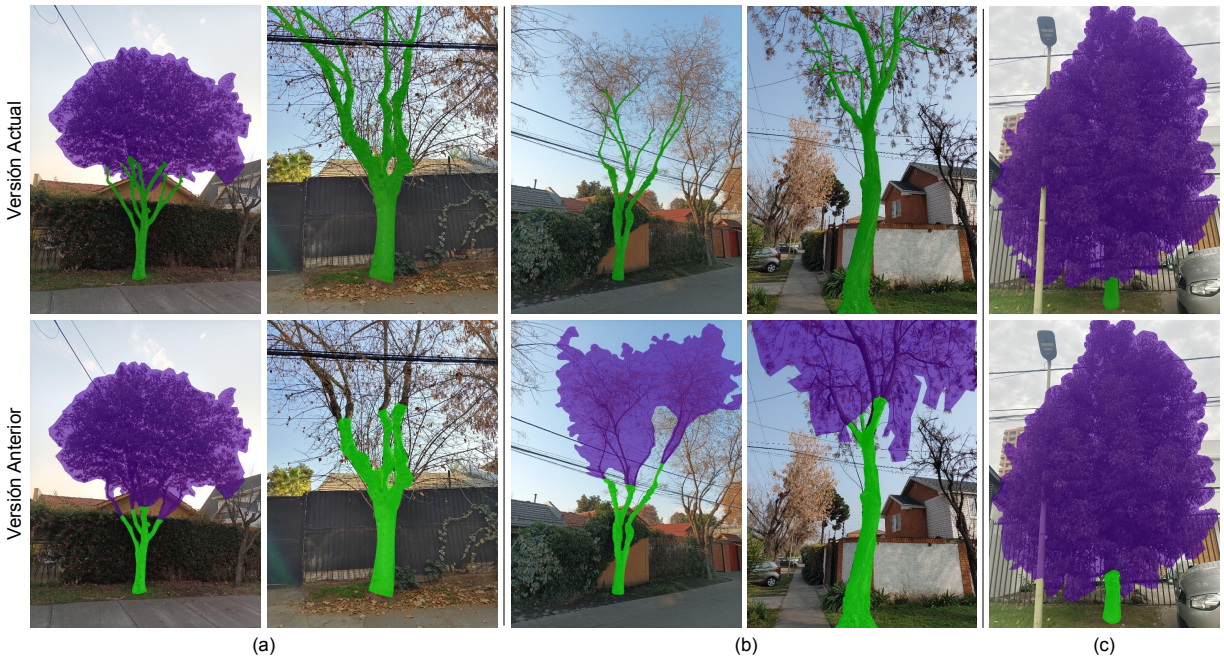

Figura 3.2: Ejemplos de imágenes re-etiquetadas de TreeSegmentation. (a) muestra el reajuste de polígonos de la clase *cortex* (verde), (b) la eliminación de polígonos de la clase *crown* (morado) y (c) la eliminación de un elemento del alumbrado público.

3.1.3. Preparación del dataset

El proceso de re-etiquetamiento entrega como resultado anotaciones en formato *json* asociadas a cada imagen. Sin embargo, para llevar a cabo el entrenamiento de los modelos se requiere como etiqueta una imagen de dimensión $H \times W$ donde cada pixel tenga un valor entero dentro del rango [0, num_clases - 1]. La generación de estas etiquetas a partir de los archivos *json* fue parte del desarrollo.

```
{
    "description"        : str,
    "tags"               : [tags],
    "size"               : {"height": int, "width": int},
    "objects": [{
          "id"           : int,
          "classId"      : int,
          "description"  : str,
          "geometryType" : "polygon" o "rectangule",
          "createdAt"    : datetime,
          "updatedAt"    : datetime,
          "tags"         : [tags],
          "classTitle"   : "Cortex" o "Crown",
          "points"       : {"exterior": [polygon], "interior": [polygon]},
    }, ...]
}
```

Código 3.1: Ejemplo de anotación en formato *json* asociada a cada imagen del dataset TreeSegmentation.

3.2. Urban Street Tree

3.2.1. Especificaciones del dataset

Urban Street Tree [25] es un dataset público que contiene imágenes de árboles y sus partes (hojas, tronco y ramas) en escenarios urbanos. Las imágenes son capturadas en 10 ciudades distintas mediante el uso de teléfonos móviles. Además, estas son tomadas desde distintas perspectivas, condiciones de luz y estaciones del año. Urban Street Tree se divide en imágenes para clasificación y segmentación.

En particular, en la presente memoria únicamente se hace uso de los sub-datasets *tree* y *branch* pertenecientes al conjunto de segmentación (Figura 3.3). Las especificaciones de estos dos sub-datasets son las siguientes:

- El sub-dataset *branch* tiene 1485 imágenes y 13 clases de árbol. Las imágenes pertenecientes a este sub-dataset contienen mayoritariamente árboles sin follaje. En este se etiqueta toda la estructura leñosa.
- El sub-dataset *tree* tiene 3949 imágenes y 22 clases de árbol. Las imágenes pertenecientes a este sub-dataset sólo contienen árboles con follaje. En este se etiqueta el árbol completo, sin hacer distinción entre copa y tronco.

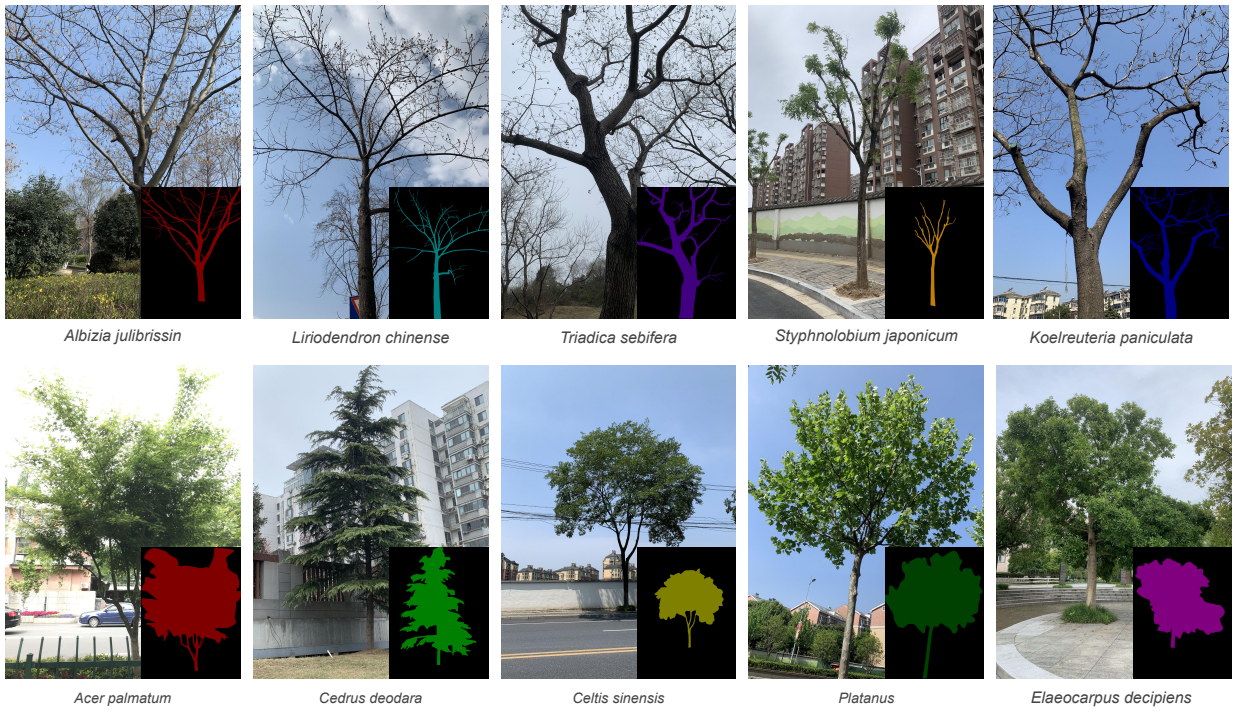

Figura 3.3: Muestra de sub-datasets. En la fila superior se presenta una muestra del sub-dataset *branch*, mientras que en la inferior del sub-dataset *tree*. Cada imagen tiene indicada la especie del árbol y su etiqueta que hace referencia a la especie mediante el valor del pixel (color).

3.2.2. Preparación del dataset

Debido a que la distinción entre especies de árboles no está dentro de los objetivos, las etiquetas de los sub-datasets *branch* y *tree* se convierten para ser agnósticas a la clase. Con este objetivo, las etiquetas son convertidas en máscaras binarias. Estos datasets indiferentes a la clase serán denominados como UST-Brach y UST-Tree.

3.3. Diseño de modelos

3.3.1. Modelos seleccionados

El modelo de referencia convolucional seleccionado es DeepLabv3+ [2]. Con respecto a los *vision transformers*, el presente trabajo pretende evaluar diferentes tipos de modelos: escala uniforme, multiescala y clasificación de máscaras. Esto con el objetivo de comparar el impacto de las distintas estrategias en el problema de arbolado urbano. Bajo este criterio, se seleccionaron los siguientes modelos: SETR [27], Segformer [23], MaskFormer [4] y Mask2Former [3].

Como se menciona en la Sección 1.3, uno de los objetivos que se busca es evaluar el impacto de la transferencia de características del *image encoder* de SAM al ámbito de árbolado urbano. Para esto, se proponen dos modelos compuestos por (1) el *encoder* de SAM entrenado sobre SA-1B y (2) un *decoder* con los pesos inicializados desde cero. Los modelos seleccionados se detallan a continuación.

3.3.2. SAM-PUP

El primer modelo que se propone lo denominamos SAM-PUP. Este se compone del *image encoder* de SAM y del *decoder* PUP. Este modelo lleva a cabo los siguientes pasos. Dada una imagen $H \times W \times 3$, esta es particionada y dada como entrada al *encoder* de SAM que genera *image embeddings* de dimensión $256 \times \frac{H}{16} \times \frac{W}{16}$. A partir de estos, PUP produce las máscaras de segmentación mediante 4 capas convolucionales, cada una seguida de un aumento en la resolución por un factor de 2 mediante una interpolación bilineal. Un esquema general se presenta en la Figura 3.4.

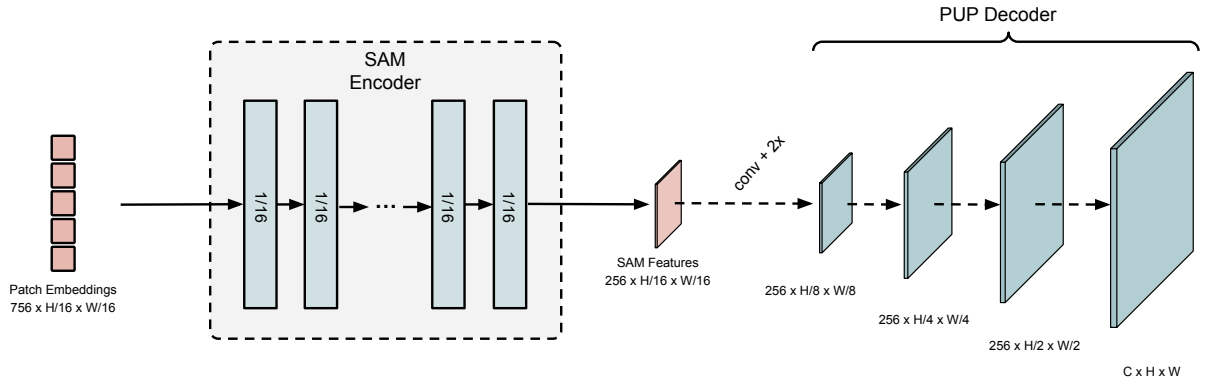

Figura 3.4: Arquitectura SAM-PUP. El *encoder* de SAM general los *image embeddings* que son la entrada para el *decoder* PUP.

3.3.3. SAM-Mask

El segundo modelo consiste en la utilización del *mask decoder* de SAM ligeramente modificado para generar distribuciones de probabilidad. Esto debido a que SAM es agnóstico a la clase. El diseño detallado se muestra en la Figura 3.5. Este recibe como entrada los *image embeddings* y una secuencia de N tokens (*embeddings*) entrenables de dimensión 256 para generar como salida N máscaras binarias asociadas cada una a una distribución de probabilidad sobre las $K + 1$ clases.

En una primera instancia, los *tokens* son actualizados con la información de los *image embeddings* y viceversa, mediante el uso de capas de *cross-attention*. Luego, los *tokens* atienden nuevamente sobre los *image embeddings*. Estos *tokens* actualizados son procesados por una 3-MLP para generar la distribución de probabilidad. Por último, las máscaras son obtenidas mediante un producto punto entre los *image embeddings* re-escalados por un factor de 4 –mediante convoluciones transpuestas– y los *tokens* actualizados procesados por una nueva 3-MLP.

Un hiperparámetro importante es el número de *tokens*. Para los entrenamientos se utiliza $N = 3$ para los tres modelos que llevan a cabo clasificación de máscaras: MaskFormer, Mask2Former y SAM-Mask. En el Anexo B.1 se presentan los resultados obtenidos variando el número de *tokens* para el modelo MaskFormer.

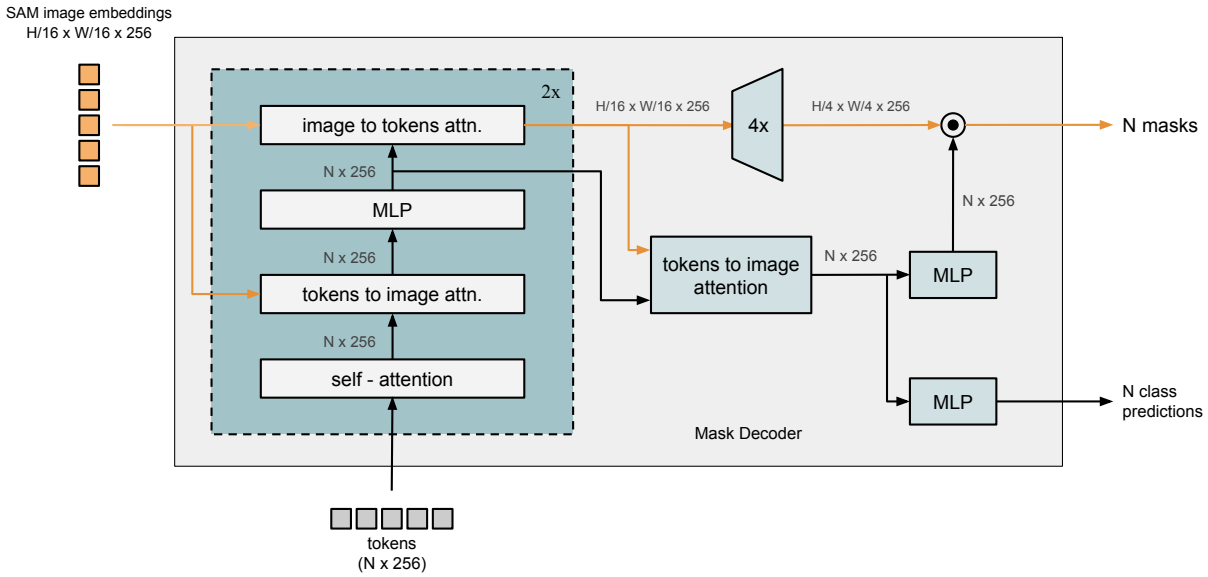

Figura 3.5: Esquema de SAM-Mask. Para los entrenamientos se utiliza $N = 3$.

Capítulo 4

Resultados experimentales

En este capítulo se presentan los detalles y los resultados de los experimentos realizados durante este trabajo. La Sección 4.1 detalla las tecnologías usadas durante los experimentos. La Sección 4.2 presenta una evaluación general de los distintos tipos de modelos sobre Tree-Segmentation. Complementariamente, la Sección 4.3 entrega una evaluación más completa sobre los modelos que incorpran SAM. Finalmente, un conjunto de resultados de modelos pre-entrenados sobre *Urban Street Tree* son presentados en la Sección 4.4.

4.1. Hardware y Software

El hardware utilizado para el presente trabajo fue una GPU Nvidia RTX A6000 con 48GB de memoria. Con respecto al software, se hace uso del código y de las implementaciones de los modelos provistas por el framework *mmsegmentation*[1]. En el caso particular del modelo SAM, las implementaciones utilizadas provienen del repositorio original[2] y son modificadas para su uso dentro del framework señalado.

4.2. Evaluación general

La Tabla 4.1 muestra la comparación general de los distintos tipos de modelos sobre el dataset TreeSegmentation. Para los entrenamientos se utiliza AdamW como optimizador con una tasa de aprendizaje de 6 x 10^{-5} para *backbones* basados en *transformers* y 10^{-4} para convolucionales. Cada entrenamiento se lleva a cabo con 10K iteraciones, un *batch* de tamaño 8 y se aplican las transformaciones por defecto de *mmsegmentation*: volteo horizontal, random re-scaling, random cropping 512 x 512 y distorsión fotométrica.

En términos generales, los modelos obtienen desempeños sobre los 80 mIoU, logrando predicciones ajustadas como las presentadas en la Figura 4.1. El modelo con mejor desempeño

[1] https://mmsegmentation.readthedocs.io/
[2] https://github.com/facebookresearch/segment-anything

es Mask2Former (ResNet-50) con 86.95 mIoU, destacándose principalmente en la clase *cortex*. Esto posiblemente favorecido por la incorporación de mapas de características multiescala en el *transformer decoder* como se observa en la Figura 2.8.

No obstante, Mask2Former y los demás modelos presentan similitudes en el desempeño, específicamente en la métrica mIoU. Esta similitud tiene su origen en la baja cantidad de instancias por imagen, existiendo un único árbol etiquetado en la mayoría de ellas. Por una parte, los modelos lograron transversalmente buenos desempeños en imágenes con un claro árbol objetivo (Figura 4.1). Sin embargo, también se tradujo en que los distintos modelos tuvieran bajos desempeños en las pocas imágenes que tenían más de un árbol etiquetado o en aquellas donde los límites con los árboles adyacentes eran difíciles de distinguir.

Modelo	Pre	Backbone	#Params	mIoU	$\text{IoU}_{\text{crown}}$	$\text{IoU}_{\text{cortex}}$
DeepLabv3+	IN-1K	R-50c	41.2 M	85.96	84.62	78.61
SegFormer	IN-1K	MiT-B3	44.6 M	86.06	86.45	76.67
		MiT-B4	61.4 M	86.31	86.30	77.52
MaskFormer	IN-1K	R-50	41.2 M	85.26	83.0	78.65
		Swin-T	41.7 M	85.92	84.17	79.06
Mask2Former	IN-1K	R-50	43.9 M	**86.95**	86.06	79.69
		Swin-T	47.4 M	86.91	85.72	**80.07**
SETR-PUP	IN-21K	ViT-B	89.9 M	82.89	84.96	69.42
SETR-MLA			91.4 M	84.08	86.45	71.03
SAM-PUP	SA-B1	ViT-B	89.6 M	85.63	87.65	74.0
		ViT-L	307 M	86.01	87.64	75.07
SAM-Mask	SA-B1	ViT-B	92.8 M	86.27	**87.95**	75.47
		ViT-L	310 M	86.60	87.77	76.66

Tabla 4.1: Resultados sobre el conjunto de validación de TreeSegmentation. SETR y SAM son modelos de escala uniforme, todos los demás incorporan algún elemento multiescala. La columna "Pre" indica el pre-entrenamiento del *backbone*. "IN-1K" indica un pre-entrenamiento sobre ImageNet-1K y "IN-21K" sobre ImageNet-21K.

Independientemente de lo anterior, una observación relevante de la Tabla 4.1 es que los modelos de escala uniforme obtienen un menor desempeño en la clase *cortex* que aquellos

que incorporan alguna componente multiescala, existiendo una diferencia de ~ 5 $\text{IoU}_{\text{cortex}}$ en promedio. En particular, SETR-PUP obtiene el menor desempeño con 69.42 $\text{IoU}_{\text{cortex}}$. La diferencia entre ambos tipos de modelos puede deberse a la naturaleza no-local de las características de los modelos de escala uniforme, lo que dificulta la segmentación de bordes, objetos pequeños o delgados como las ramificaciones.

Otro aspecto a destacar es el desempeño de SAM-Mask en la clase *crown*, logrando el mejor resultado con 87.95 $\text{IoU}_{\text{crown}}$. Una comparación entre SAM y SETR, ambos modelos basados en ViT, sugiere un impacto positivo en la transferencia de características aprendidas sobre SA-1B. En particular, SAM-PUP (ViT-B) obtiene +2.69 $\text{IoU}_{\text{crown}}$ y +4.58 $\text{IoU}_{\text{cortex}}$ en comparación con SETR-PUP. Sin embargo, la real magnitud de este impacto no puede ser concluida dado que ambas implementaciones de ViT presentan diferencias.

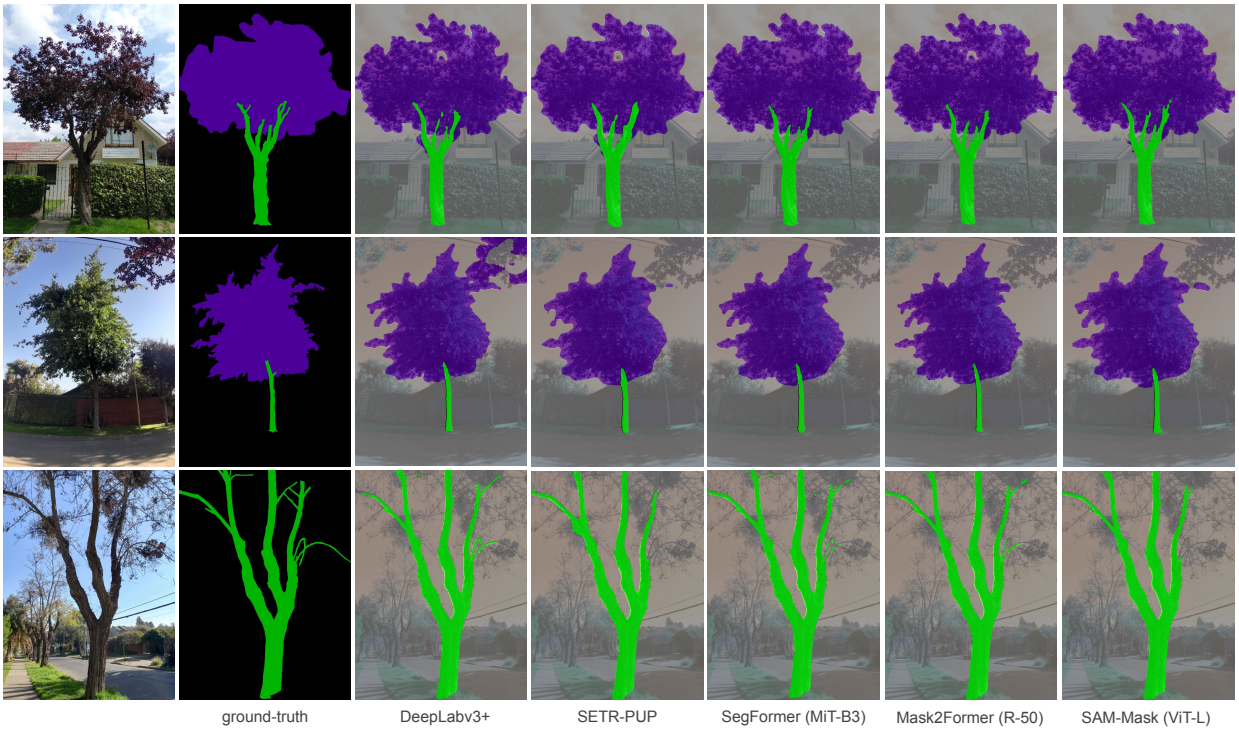

Figura 4.1: Ejemplo de predicciones realizadas por distintos modelos sobre TreeSegmentation.

4.2.1. Análisis de casos

Un primer caso difícil es la segmentación de ramas (Figura 4.2). Esto se debe principalmente a la complejidad asociada a la segmentación de objetos con forma delgada y ramificada donde las características de bajo nivel cobran mayor relevancia. Esto explicaría, en parte, el menor desempeño que presentan los modelos de escala uniforme en la clase *cortex*. Además, estas estructuras suelen estar parcialmente cubiertas con follaje, complejizando tanto el etiquetado como la distinción entre ambas clases por parte de los modelos.

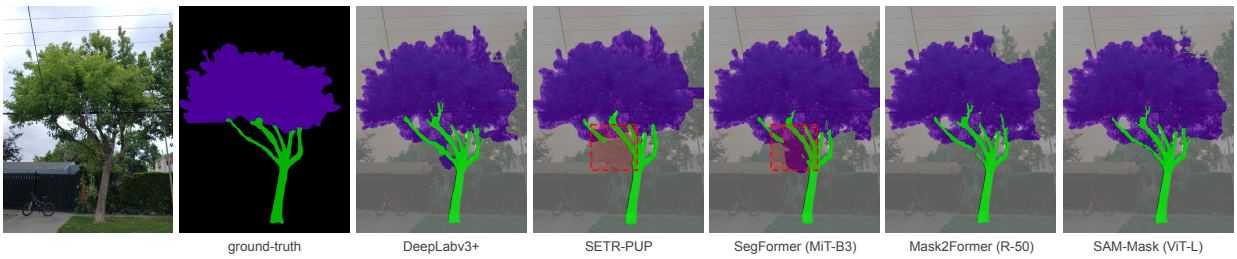

Figura 4.2: Ejemplo de segmentación de ramas. El área demarcada de color rojo indica el error en la predicción del modelo.

Un segundo caso difícil está asociado con la presencia de árboles adyacentes. Los distintos modelos presentan casos donde segmentan parcialmente troncos o copas de árboles adyacentes, tal como se observa en la Figura 4.3. En la versión actual de TreeSegmentation se optó por re-etiquetar imágenes con un árbol objetivo claro (con algunas excepciones) para disminuir esta tendencia por parte de los modelos.

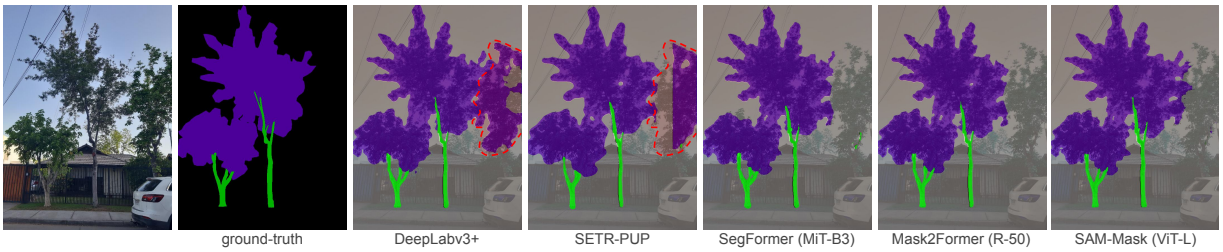

Figura 4.3: Ejemplo de segmentación de árbol adyacente. El área demarcada de color rojo indica el error en la predicción del modelo.

4.3. *Segment Anything Model*

En la Tabla 4.2 se muestra una evaluación más completa de los modelos que incorporan el *image encoder* de SAM sobre el dataset TreeSegmentation. En esta se agrega la evaluación de los modelos con el *encoder* congelado. Esto tiene como objetivo observar la capacidad de los *decoders* de aprovechar la información contenida en los *image embeddings*. El esquema de entrenamiento usado es el descrito en la Sección 4.2.

En la Tabla 4.2 se observa que el mejor modelo es SAM-Mask (ViT-L) con 86.60 mIoU. Sin embargo, la ganancia con respecto a SAM-Mask (ViT-B) es mínima con sólo 0.33 mIoU de diferencia. Esto se repite en el modelo SAM-PUP, lo que indica que el uso de modelos de mayor capacidad no se traduce en una mejora significativa en el desempeño. Esto es coherente con el número de imágenes y la baja cantidad de categorías presentes en el dataset TreeSegmentation.

Un resultado a destacar es el desempeño obtenido por los modelos con el *encoder* congelado, siendo >81 mIoU en todos los casos. El mejor de estos modelos SAM-Mask (ViT-L)

obtiene 82.42 mIoU, acercándose al modelo SETR-PUP (Tabla 4.1). Estos valores sugieren un alto valor semántico de los *image embeddings* entregados por SAM. Sin embargo, la Tabla 4.2 también muestra que la contraparte con el *encoder* ajustado sobre TreeSegmentation tiene un desempeño superior de ~ 4 mIoU en promedio.

Modelo	Backbone	#Params	mIoU	$\text{IoU}_{\text{crown}}$	$\text{IoU}_{\text{cortex}}$
SAM-PUP	ViT-B (F)	89.6 M	81.25	81.13	69.52
	ViT-B		85.63	87.65	74.0
	ViT-L (F)	307 M	81.84	82.32	69.77
	ViT-L		86.01	87.64	75.07
SAM-Mask	ViT-B (F)	92.8 M	82.21	82.78	70.20
	ViT-B		86.27	**87.95**	75.47
	ViT-L (F)	310 M	82.42	82.97	70.63
	ViT-L		**86.60**	87.77	**76.66**

Tabla 4.2: Comparación de los modelos que incorporan el *image encoder* de SAM sobre el conjunto de validación de TreeSegmentation. "F" denota el uso del *backbone* congelado.

4.4. *Urban Stree Tree*

En la Tabla 4.3 se presentan los resultados obtenidos sobre TreeSegmentation con modelos pre-entrenados sobre UST-Branch y UST-Tree (Sección 3.2.2). También se muestran los resultados obtenidos de pre-entrenar sobre ambos datasets. Para esto último, las etiquetas de UST-Branch y UST-Tree se trataron bajo una única categoría general denominada *tree*. Los entrenamientos se llevaron a cabo usando el esquema descrito en la Sección 4.2.

Como se puede observar, los modelos obtienen una ganancia en el desempeño al ser pre-entrenados sobre ambos datasets (Branch + Tree), excepto DeepLabv3+. Sin embargo, este aumento es poco significativo, siendo menor a 1 mIoU en todos los casos. Otro punto es que el pre-entrenamiento tiene mayor impacto sobre los modelos con *backbones* basados en *transformer* que aquellos convolucionales, aunque también de manera poco significativa. Los primeros obtienen un incremento de +0.66 mIoU en promedio, mientras que los segundos de +0.19 mIoU si se considera el mejor resultado para cada uno.

Por último, cabe destacar al modelo Mask2Former (ResNet-50) que bajo dos de los esquemas de pre-entrenamiento logra superar los 87 mIoU, convirtiéndose en el mejor resultado obtenido en la presente memoria.

Modelo	Backbone	Pre	mIoU	IoU_{crown}	IoU_{cortex}
DeepLabv3+	R-50c	IN-1K	85.96	84.62	78.61
		UST-Tree	**86.07**	84.42	79.22
		UST-Branch	85.52	84.31	77.81
		Branch + Tree	85.69	84.03	78.55
SegFormer	MiT-B3	IN-1K	86.06	86.45	76.67
		UST-Tree	85.85	86.27	76.29
		UST-Branch	86.71	87.07	77.79
		Branch + Tree	**86.81**	87.90	77.0
SETR-PUP	ViT-B	IN-21K	82.89	84.96	69.42
		UST-Tree	83.26	85.25	70.10
		UST-Branch	82.88	82.71	72.11
		Branch + Tree	**83.45**	85.40	70.55
Mask2Former	R-50	IN-1K	86.95	86.06	79.69
		UST-Tree	86.70	85.95	79.09
		UST-Branch	**87.21**	86.36	80.07
		Branch + Tree	87.15	85.98	80.38

Tabla 4.3: Resultados sobre el conjunto de validación de TreeSegmentation con modelos pre-entrenados sobre UST-Tree y UST-Branch. "Branch + Tree" indica la utilización de ambos datasets.

Capítulo 5

Conclusiones

5.1. Conclusiones

Con respecto a la comparación entre los distintos tipos de modelos, los experimentos arrojaron que el mejor modelo corresponde a Mask2Former con independencia del *backbone* y el pre-entrenamiento utilizado. Este resultado, en conjunto con el buen desempeño de MaskFormer y SAM-Mask, permiten establecer que la estrategia de clasificación de máscaras es competitiva para su uso en arbolado urbano.

Otro resultado obtenido es la ventaja que muestran los modelos multiescala con respecto a los de escala uniforme en la clase *cortex*. Los primeros muestran una ventaja de apróximadamente +5 $\text{IoU}_{\text{cortex}}$ en promedio. Este resultado, el menor costo computacional y su buen desempeño general indicaría que el uso de modelos multiescala es más recomendable para el problema tratado en el presente trabajo.

Por último, los experimentos realizados sugieren que la transferencia de características desde el *image encoder* de SAM mejora el desempeño. Esto se extrae de la comparación con el otro modelo de escala uniforme SETR, también basado en ViT. Sin embargo, las implementaciones de ViT de ambos modelos difieren y, por ende, estos resultados no son concluyentes.

5.2. Trabajo Futuro

Acorde con lo presentado en el presente trabajo de memoria, se proponen los siguientes trabajos futuros:

- **Ampliar TreeSegmentation:** actualmente se cuenta con 1289 imágenes adicionales sin etiquetar o que requieren ser re-etiquetadas. Este trabajo consiste en aumentar el número de imágenes de TreeSegmentation usando el protocolo diseñado (Anexo A.1). Esto permitiría re-evaluar los modelos sobre un dataset más amplio.

- **Ampliar evaluación SAM:** este trabajo consiste en incorporar a la evaluación nuevas implementaciones muy recientes y más eficientes de SAM [24] que, eventualmente, podrían ser más adecuadas para el problema tratado en la memoria dada su menor complejidad.

Bibliografía

[1] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. volume 40, pages 834–848, 2018.

[2] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[3] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1290–1299, June 2022.

[4] Bowen Cheng, Alex Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 17864–17875. Curran Associates, Inc., 2021.

[5] Kwanghun Choi, Wontaek Lim, Byungwoo Chang, Jinah Jeong, Inyoo Kim, Chan-Ryul Park, and Dongwook W Ko. An automatic approach for tree species detection and profile estimation of urban street trees using deep learning and google street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 190:165–180, 2022.

[6] Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Twins: Revisiting the design of spatial attention in vision transformers. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 9355–9366. Curran Associates, Inc., 2021.

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[8] Jiaqi Gu, Hyoukjun Kwon, Dilin Wang, Wei Ye, Meng Li, Yu-Hsin Chen, Liangzhen Lai, Vikas Chandra, and David Z. Pan. Multi-scale high-resolution vision transformer

for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12094–12103, June 2022.

[9] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 16000–16009, June 2022.

[10] Jitesh Jain, Anukriti Singh, Nikita Orlov, Zilong Huang, Jiachen Li, Steven Walton, and Humphrey Shi. Semask: Semantically masked transformers for semantic segmentation. *CoRR*, abs/2112.12782, 2021.

[11] Danilo Samuel Jodas, Sergio Brazolin, Takashi Yojo, Reinaldo Araujo de Lima, Giuliana Del Nero Velasco, Aline Ribeiro Machado, and João Paulo Papa. A deep learning-based approach for tree trunk segmentation. In *2021 34th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, pages 370–377, 2021.

[12] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloé Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross B. Girshick. Segment anything. *CoRR*, abs/2304.02643, 2023.

[13] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 280–296, Cham, 2022. Springer Nature Switzerland.

[14] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2015.

[16] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10012–10022, October 2021.

[17] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.

[18] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[19] Robin Strudel, Ricardo Garcia Pinel, Ivan Laptev, and Cordelia Schmid. Segmenter: Transformer for semantic segmentation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 7242–7252. IEEE, 2021.

[20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[21] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 568–578, October 2021.

[22] Junde Wu, Rao Fu, Huihui Fang, Yuanpei Liu, Zhaowei Wang, Yanwu Xu, Yueming Jin, and Tal Arbel. Medical SAM adapter: Adapting segment anything model for medical image segmentation. *CoRR*, abs/2304.12620, 2023.

[23] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, José M. Álvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 12077–12090, 2021.

[24] Yunyang Xiong, Bala Varadarajan, Lemeng Wu, Xiaoyu Xiang, Fanyi Xiao, Chenchen Zhu, Xiaoliang Dai, Dilin Wang, Fei Sun, Forrest Iandola, Raghuraman Krishnamoorthi, and Vikas Chandra. Efficientsam: Leveraged masked image pretraining for efficient segment anything. *arXiv:2312.00863*, 2023.

[25] Tingting Yang, Suyin Zhou, Zhijie Huang, Aijun Xu, Junhua Ye, and Jianxin Yin. Urban street tree dataset for image classification and instance segmentation. *Computers and Electronics in Agriculture*, 209:107852, 2023.

[26] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions, 2016.

[27] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6881–6890, June 2021.

ANEXOS

Anexo A

Protocolo de etiquetamiento

A.1. Protocolo TreeSegmentation

El dataset TreeSegmentation tiene como objetivo el entrenamiento de modelos capaces de segmentar diversas partes del árbol principal presente en una imagen. Algunas definiciones relevantes para el proceso de etiquetamiento son las siguientes:

- **Corteza:** toda la estructura leñosa del árbol. Esta se compone tanto del tronco como de las ramas.
- **Copa:** estructura del árbol que presenta follaje. En caso de ausencia de follaje, la estructura leñosa descubierta es considerada como corteza.

El procedimiento de etiquetamiento se compone de las siguientes reglas:

1. **Polígono Corteza.** La corteza de cada árbol debe ser etiquetada con un polígono de categoría *Cortex*. Este proceso se lleva a cabo de manera diferente según exista presencia de follaje o no:
   (a) En ausencia de follaje las reglas son las siguientes:
      - El polígono se debe ajustar lo más posible a la estructura leñosa del árbol.
      - El polígono debe incluir como mínimo (1) el tronco principal y (2) las ramas secundarias

- Adicionalmente, se debe incluir toda rama que sea relevante dentro de la estructura general del árbol. Se debe considerar principalmente el ancho y contraste de las ramas.
- El fondo presente entre las ramas debe ser eliminado.

(b) En presencia de follaje las reglas son las siguientes:

- El polígono consiste en toda la estructura leñosa (tronco y ramas) desde la base del tronco hasta su límite con el follaje. En los casos en que el límite sea poco claro, el follaje comienza cuando predomina por sobre la estructura leñosa y la forma de esta última no se alcanza a distinguir claramente.
- El polígono debe ser continuo. Las ramas aisladas presentes dentro del follaje no deben ser etiquetadas como corteza.
- El fondo presente entre las ramas no debe ser parte del polígono.

2. **Polígono Crown.** La copa de cada árbol está etiquetada con un polígono de categoría *Crown*. El proceso de etiquetamiento de cada copa se lleva a cabo mediante las siguientes reglas:

   (a) Los árboles con escaso o nulo follaje no deben presentar este polígono.

   (b) El polígono se debe ajustar lo más posible al follaje del árbol.

   (c) En caso que una rama o tronco tape el follaje, el polígono de la copa debe conectar los dos puntos de follaje más cercanos a ambos lados de la rama o tronco.

3. **Prioridad Polígonos.** Cada polígono debe tener asociado un número que indique su prioridad durante el proceso de creación de la máscara de anotación. Este número está representado por el tag *priority*. Algunas consideraciones:

   (a) Si dos polígonos distintos tienen asociado el mismo número, entonces su orden relativo durante la generación de la etiqueta de segmentación no está determinada.

   (b) Si en una imagen está presente una sola categoría, no es necesario el uso de este tag.

4. **Alumbrado Público.** Los elementos pertenciente al alumbrado público, letrero u otro no deben pertenecer a ningún polígono. Estos deben ser rodeados o, en caso de no ser posible, dividir el polígono correspondiente.

Anexo B

Entrenamientos

B.1. Variación número de *embeddings*

En la Tabla 5.1 muestra los resultados del modelo MaskFormer al variar el número de *embeddings*.

#Embedding	mIoU	$\text{IoU}_{\text{crown}}$	$\text{IoU}_{\text{cortex}}$
3	85.26	83.0	78.65
5	56.27	49.29	38.75
10	56.95	48.98	48.80
100	*NC*	*NC*	*NC*

Tabla 5.1: Resultados de MaskFormer (ResNet-50) sobre el conjunto de validación de Tree-Segmentation variando el número de *embeddings*. "*NC*" indica que el modelo no converge.

B.2. Resultados UST-Branch y UST-Tree

A continuación, se muestran los resultados obtenidos sobre los datasets UST-Branch y UST-Tree. El esquema de entrenamiento utilizado es el descrito en la Sección 4.2.

Modelo	Pre	Backbone	#Params	mIoU	IoU_{branch}
DeepLabv3+	IN-1K	R-50c	41.2 M	92.75	86.91
SegFormer	IN-1K	MiT-B3	44.6 M	**92.83**	**87.06**
Mask2Former	IN-1K	R-50	43.9 M	92.66	86.76
SETR-PUP	IN-21K	ViT-B	89.9 M	92.43	86.33

Tabla 5.2: Resultados sobre el conjunto de validación de UST-Branch.

Modelo	Pre	Backbone	#Params	mIoU	IoU_{tree}
DeepLabv3+	IN-1K	R-50c	41.2 M	90.16	86.77
SegFormer	IN-1K	MiT-B3	44.6 M	90.58	87.28
Mask2Former	IN-1K	R-50	43.9 M	88.03	84.19
SETR-PUP	IN-21K	ViT-B	89.9 M	**91.06**	**87.86**

Tabla 5.3: Resultados sobre el conjunto de validación de UST-Tree.

Modelo	Pre	Backbone	#Params	mIoU	IoU_{tree}
DeepLabv3+	IN-1K	R-50c	41.2 M	89.14	84.12
SegFormer	IN-1K	MiT-B3	44.6 M	**89.68**	**84.90**
Mask2Former	IN-1K	R-50	43.9 M	87.61	82.13
SETR-PUP	IN-21K	ViT-B	89.9 M	89.40	84.44

Tabla 5.4: Resultados sobre el conjunto de validación de UST-Branch y UST-Tree combinados.