UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

SEGMENTACIÓN SEMÁNTICA CON SUPERVISIÓN DÉBIL UTILIZANDO DESCRIPCIONES EN LENGUAJE NATURAL

TESIS PARA OPTAR AL GRADO DE MAGÍSTER EN CIENCIAS DE LA INGENIERÍA, MENCIÓN ELÉCTRICA

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

DANIEL ROJAS VILAR

PROFESOR GUÍA:
DR. CLAUDIO PÉREZ FLORES

MIEMBROS DE LA COMISIÓN:
DR. PABLO ESTÉVEZ VALENCIA
DR. PABLO ZEGERS FERNÁNDEZ

Este trabajo ha sido parcialmente financiado por ANID (CONICYT) a través de los proyectos FONDEF ID16I20290, FONDECYT 1191610, así como el Departamento de Ingeniería Eléctrica (DIE) y el Advanced Mining Technology Center (ANID Project AFB180004).

SANTIAGO DE CHILE
2021

RESUMEN DE LA TESIS PARA OPTAR
AL GRADO DE MAGÍSTER EN CIENCIAS
DE LA INGENIERÍA, MENCIÓN ELÉCTRICA
RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO
POR: DANIEL ROJAS VILAR
FECHA: 2021
PROF. GUÍA: DR. CLAUDIO PÉREZ FLORES

SEGMENTACIÓN SEMÁNTICA CON SUPERVISIÓN DÉBIL UTILIZANDO DESCRIPCIONES EN LENGUAJE NATURAL

En esta tesis se propone una metodología para entrenar modelos de segmentación semántica utilizando imágenes anotadas únicamente con descripciones en lenguaje natural. En los trabajos previamente publicados, se asignan etiquetas semánticas a fragmentos de descripciones mediante búsqueda exacta de nombres de clases. Luego se utilizan modelos entrenados para localizar texto arbitrario en imágenes para generar máscaras de segmentación artificiales, con las que se entrenan métodos supervisados.

En esta tesis, se propone aprovechar la estructura sintáctica de las descripciones, junto con información semántica de una base de conocimiento, para mejorar la detección de categorías relevantes, además de identificar atributos y categorías complementarias. El método propuesto no requiere de anotaciones adicionales, por lo que puede extenderse fácilmente a nuevas aplicaciones. Se presenta además una red de localización, que se entrena para predecir exclusivamente las etiquetas generadas, lo que focaliza su entrenamiento en información relevante, mejorando los mapas de localización resultantes. Finalmente, se describe un método para obtener máscaras de segmentación más precisas y completas, aprovechando todos los tipos de mapas generados.

Esta metodología se valida mediante varios experimentos en la base de datos MS-COCO, demostrando que supera por un amplio margen todos los métodos anteriores basados en supervisión a nivel de imagen.

A mis padres y familia,
gracias por el apoyo incondicional.

Agradecimientos

En primer lugar, me gustaría agradecer a mi familia, y muy especialmente a mis padres, por siempre confiar en mí y en todos los proyectos en los que me embarco.

A Sofía, por el aliento y cariño durante todos estos años que hemos pasado juntos.

A mis compañeros de laboratorio, por las conversaciones y carcajadas que hicieron más amenas las tardes de trabajo.

A Felipe Fredes, por la información privilegiada sobre el proceso de titulación.

Y a mi profesor guía, Dr. Claudio Pérez, por su consejo y dedicación durante el desarrollo de esta tesis.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

La tarea de Segmentación Semántica consiste en clasificar cada pixel de una imagen según la categoría del objeto al que pertenece. Este es uno de los problemas más fundamentales en los campos de procesamiento de imágenes y visión computacional, y cuenta con aplicaciones en áreas tales como análisis de imágenes médicas, comprensión de escenas, conducción autónoma, vigilancia, compresión de imágenes, y robótica, entre muchas otras. Durante los últimos años, el estado del arte para segmentación semántica ha experimentado un avance dramático, gracias en parte al desarrollo de modelos basados en redes neuronales convolucionales o *convolutional neural networks* (CNNs). Sin embargo, para entrenar de forma efectiva este tipo de modelos se requiere en general de enormes cantidades de imágenes anotadas con máscaras de segmentación a nivel de pixeles. El alto costo de generar manualmente este tipo de supervisión a gran escala ha limitado severamente la expansión de estos modelos hacia dominios de aplicación más complejos y diversos.

Estas limitaciones han motivado recientemente el desarrollo de múltiples trabajos que buscan entrenar redes convolucionales para segmentación semántica usando supervisión más "débil", con el objetivo de reducir los costos de anotación. Los tipos de anotaciones alternativas que han sido estudiados en la literatura incluyen *bounding boxes* [1, 2], garabatos (*scribbles*) [3], puntos [4], e incluso supervisión solo a nivel de imagen [5, 6, 7, 8, 9, 10, 11]. Este último escenario es particularmente desafiante, al no disponer de ningún tipo de información de localización para guiar el entrenamiento del modelo. Sin embargo, debido a que esta es la formulación que reduce más drásticamente los costos de anotación, ha sido también la que ha recibido mayor atención de parte de la comunidad que investiga esta área.

La gran mayoría de los trabajos que abordan el problema de Segmentación Semántica con Supervisión Débil (WSSS, por sus siglas en inglés) se centran en el caso en el que las anotaciones a nivel de imagen corresponden a etiquetas de clasificación. Actualmente, los métodos que ofrecen los mejores resultados para este problema siguen un esquema de auto-supervisión [7, 8, 9, 10, 11], basado en la generación automática de anotaciones a nivel de pixel. Con este propósito, las etiquetas disponibles se utilizan en primer lugar para entrenar un clasificador convolucional. Una vez entrenado, las representaciones internas generadas por este tipo de modelo permiten obtener mapas rudimentarios de activación por clase, llamados *Class Activation Maps* (CAMs) [12], que destacan las regiones de una imagen que

más contribuyen a la predicción de cada categoría. Los pixeles con mayor confianza de estos CAMs se utilizan para generar máscaras de segmentación artificiales sobre el mismo conjunto de entrenamiento, que posteriormente sirven para entrenar un modelo de segmentación en régimen supervisado. El desarrollo de este tipo de métodos ha sido en gran medida motivado y posibilitado por la existencia de diversas bases de datos de gran tamaño anotadas con etiquetas de clasificación, tales como ImageNet [13], MS-COCO [14], o PascalVOC [15]. Sin embargo, este no es el caso para la mayoría de las categorías de objetos o dominios de interés, y el costo de anotar grandes bases de datos sigue siendo alto incluso para este tipo más débil de supervisión.

Una alternativa poco explorada de supervisión a nivel de imagen corresponde a descripciones en lenguaje natural, que tienen la ventaja de poder ser extraídas en grandes cantidades y para una amplia gama de aplicaciones desde Internet [16]. Adicionalmente, incluso en el contexto de anotación manual mediante plataformas de *crowdsourcing*, tales como Amazon Mechanical Turk (AMT), utilizar descripciones libres puede resultar significativamente más económico y menos propenso a errores que etiquetar exhaustivamente presencia o ausencia de múltiples categorías de objetos. Lo anterior es especialmente relevante en casos con un gran número de categorías de interés, o en aplicaciones con escenas más complejas. Las descripciones además contienen información adicional sobre contexto y atributos visuales complementarios, que pueden explotarse para mejorar la calidad de las máscaras generadas. A pesar de estas ventajas, muy pocos trabajos han sido propuestos en la literatura para abordar este problema, y los métodos existentes de WSSS basados en etiquetas de clasificación no son directamente aplicables al caso de descripciones. Motivados por lo anterior, el presente trabajo de tesis se enfoca en desarrollar una metodología para abordar el problema de **segmentación semántica utilizando descripciones en lenguaje natural**.

El único trabajo anterior que aborda este mismo problema corresponde a [17], en el que se presenta un método para obtener mapas de activación similares a CAMs a partir de descripciones. Siguiendo avances previos en la tarea de *visual grounding* [18, 19, 20, 21, 22], los autores proponen un modelo que proyecta tanto imágenes como segmentos de texto arbitrario a un mismo espacio vectorial multi-modal, preservando relaciones semánticas. Posteriormente, se identifican referencias a categorías de objetos relevantes dentro de las descripciones de entrenamiento mediante una simple regla heurística, que consiste en buscar menciones del nombre de cada clase. Los segmentos de texto que contienen nombres de clases son proyectados al espacio multi-modal, y utilizados para generar mapas de activación mediante simple producto punto con representaciones visuales localizadas. Finalmente, los mapas resultantes se utilizan de forma análoga a CAMs para entrenar un modelo de segmentación supervisado.

Sin embargo, al entrenar su modelo para localizar texto arbitrario, se introduce también información ambigua, redundante y no visual a las señales de supervisión, lo que entorpece el entrenamiento. Más aún, el modelo tiene que aprender que múltiples estructuras gramaticales pueden ser equivalentes, que una misma palabra puede corresponder a distintas categorías semánticas dependiendo del contexto (homonimia), y que distintas palabras pueden referenciar una misma categoría. Por ejemplo, la palabra "ratón" puede referirse a un ratón de computador o al roedor, mientras que la categoría de MS-COCO "persona" puede ser mencionada con un sinónimo (e.g., "individuo"), un hipónimo (e.g., "hombre"), o un holónimo (e.g., "público"). Además de la dificultad adicional que esto impone sobre el entrenamiento

del modelo, no existe una forma directa de recuperar este conocimiento para guiar la generación de las máscaras de segmentación, lo que obliga a utilizar una metodología separada para asociar segmentos de texto a categorías semánticas útiles. A su vez, la técnica de búsqueda exacta utilizada en [17] para hacer esta asignación falla en muchos de los casos más complejos descritos anteriormente, y descarta otros tipos de información útil presente en las descripciones.

En este trabajo de tesis se presenta una metodología para abordar los problemas descritos anteriormente. En lugar de entrenar un modelo para localizar texto arbitrario directamente, se desarrolla un módulo de procesamiento de texto para extraer información visual útil a partir de descripciones de manera estructurada. Este módulo combina información derivada de la estructura sintáctica de las descripciones, dada por sus correspondientes grafos de escena [23, 24], con relaciones semánticas obtenidas desde una base de conocimiento (WordNet [25]), para identificar explícitamente todas las menciones a clases de objeto relevantes, además de otras categorías complementarias, y correspondientes atributos visuales. Este módulo no requiere de supervisión adicional, es independiente del dominio de aplicación visual, y puede extenderse fácilmente a nuevas categorías de objetos.

Se propone además una red de localización que puede ser entrenada con la supervisión extraída, generalizando los clasificadores convolucionales utilizados normalmente para producir CAMs. La etapa de pre-procesamiento de las descripciones permite simplificar el diseño y el entrenamiento de este modelo, y focaliza su supervisión en información relevante y localizable. Una vez entrenada, esta red puede utilizarse para producir mapas de activación asociados a categorías de objetos relevantes y contextuales, además de categorías de objetos enriquecidas con atributos visuales específicos. Finalmente, se propone una metodología para obtener máscaras de segmentación semántica aprovechando todos los tipos de mapas de activación generados por la red de localización, que se utilizan para entrenar un modelo de segmentación semántica de forma supervisada.

1.1 Hipótesis

El trabajo desarrollado en este documento está motivado por las siguientes hipótesis de investigación:

- Las descripciones de imágenes en lenguaje natural son una alternativa efectiva a las etiquetas de clasificación como fuentes de información para entrenar modelos de segmentación semántica.
- Identificar menciones de objetos en descripciones según su estructura sintáctica, sumado a considerar explícitamente sus relaciones semánticas con categorías relevantes, permite extraer etiquetas de clasificación de mejor calidad que las obtenidas con la estrategia de búsqueda exacta utilizada en trabajos previos.
- La información sobre categorías de objetos adicionales y atributos visuales contenida en descripciones permite complementar las etiquetas de categorías relevantes, mejorando la calidad de las máscaras de segmentación generadas.
- Limitar la supervisión del modelo de localización a etiquetas de categorías y atributos visuales mejora la calidad de los mapas de activación aprendidos, en comparación con

los generados por modelos entrenados para localizar texto arbitrario.

1.2 Objetivos

1.2.1 Objetivo General

El objetivo general del presente trabajo de tesis consiste en desarrollar un modelo de segmentación semántica basado en supervisión débil utilizando descripciones en lenguaje natural, que avance el estado del arte para este problema mediante el uso de una representación estructurada de la información visual presente en descripciones.

1.2.2 Objetivos Específicos

Para lograr el objetivo general, se proponen los siguientes objetivos específicos:

- Diseñar, implementar, y probar un módulo de procesamiento de texto para extraer etiquetas de categorías y atributos visuales a partir de descripciones, utilizando información sobre estructuras sintácticas, y relaciones semánticas definidas en WordNet.
- Diseñar, implementar, y probar un modelo de localización que permita generar mapas de activación asociados a categorías semánticas, y a categorías enriquecidas con atributos visuales específicos.
- Evaluar la calidad de las etiquetas semánticas extraídas usando el método propuesto, comparando su efectividad como supervisión para entrenar modelos de segmentación semántica con la de las etiquetas generadas usando el método de búsqueda exacta propuesto previamente, y con las etiquetas de clasificación *ground truth*.
- Evaluar el efecto de incluir mapas de activación asociados a categorías complementarias y a pares compuestos del tipo categoría-atributo sobre la calidad de las máscaras de segmentación generadas.
- Comparar la calidad de las máscaras de segmentación generadas por el modelo propuesto con las obtenidas usando modelos entrenados para localizar texto arbitrario.

1.3 Contribuciones de la Tesis

Las principales contribuciones del presente trabajo de tesis pueden resumirse como sigue:

- En primer lugar, se propone un módulo de procesamiento de texto para identificar menciones de categorías de objetos relevantes, categorías de fondo complementarias, y atributos visuales a partir de descripciones en lenguaje natural, que no requiere de supervisión adicional, lo que simplifica el etiquetado de bases de datos para entrenar modelos de segmentación.
- Se desarrolla además una red de localización que extiende los clasificadores convolucionales utilizados normalmente en aplicaciones de WSSS, permitiendo generar mapas de activación para categorías de objetos, y para parejas del tipo categoría-atributo.
- Adicionalmente, se presenta una metodología para obtener máscaras de segmentación artificiales aprovechando la información complementaria de todos los tipos de mapas de activación generados por la red de localización.

- Finalmente, se realizan extensivos experimentos sobre la base de datos MS-COCO que validan la efectividad de cada componente de la metodología propuesta, y se demuestra que el método desarrollado supera el estado del arte para segmentación semántica con supervisión a nivel de imagen en esta base de datos por un margen de 7.6% mIoU.

1.4 Estructura de la Tesis

El resto de este documento se estructura de la siguiente manera:

En el Capítulo 2 se formaliza el problema de segmentación semántica con descripciones en lenguaje natural, así como las principales métricas de desempeño utilizadas para evaluar la metodología propuesta.

El Capítulo 3 presenta una revisión bibliográfica de trabajos previos relevantes. En primer lugar, se detalla una serie de herramientas generales utilizadas por el método propuesto, o relevantes para entender el funcionamiento de otros trabajos similares. Posteriormente, se revisan métodos existentes que abordan el problema de segmentación semántica con supervisión débil, y otros problemas relacionados.

En el Capítulo 4 se presentan los detalles de la metodología propuesta, incluyendo la implementación del módulo de procesamiento de texto, así como la arquitectura y método de entrenamiento de la red de localización. Se describe además la metodología utilizada para obtener máscaras de segmentación artificiales a partir de los mapas generados por la red de localización, así como el procedimiento seguido para entrenar el modelo de segmentación supervisado. Finalmente, se presenta la metodología experimental utilizada para validar el desempeño del modelo propuesto.

En el Capítulo 5 se presentan los resultados obtenidos en los distintos experimentos realizados, junto con sus respectivos análisis. Finalmente, en el Capítulo 6 se presentan las conclusiones derivadas del trabajo desarrollado, y se describen posibles direcciones de trabajo futuro.

Capítulo 2

Definición del Problema

2.1 Segmentación Semántica

La tarea de segmentación semántica consiste en particionar una imagen de entrada en múltiples regiones o segmentos, de acuerdo a las categorías de los distintos objetos que la componen. Más formalmente, dada una imagen de entrada $\mathbf{I}$, representada digitalmente como una matriz de $h \times w$ pixeles, se busca obtener una máscara de segmentación $\boldsymbol{M} \in (\mathcal{C}_{\text{seg}})^{h \times w}$ tal que cada elemento $M_{i,j}$ de $\boldsymbol{M}$ es $M_{i,j} = c$ si y solo si el pixel con coordenadas (i, j) en $\mathbf{I}$ forma parte de un objeto de la clase c. De esta forma, la segmentación semántica puede entenderse como un problema de clasificación sobre un conjunto fijo de categorías de objetos $\mathcal{C}_{\text{seg}}$, aplicado a cada pixel de la imagen de entrada.

Típicamente, además del conjunto de categorías de objetos relevantes para una aplicación, denotado como $\mathcal{C}_{\text{fg}}$, $\mathcal{C}_{\text{seg}}$ incluye una etiqueta especial de "fondo", $\langle \mathbf{bg} \rangle$, que se utiliza para etiquetar todos los pixeles que no pertenecen a ninguna categoría en $\mathcal{C}_{\text{fg}}$. Con esta notación, se tiene que $\mathcal{C}_{\text{seg}} = \mathcal{C}_{\text{fg}} \cup \{\langle \mathbf{bg} \rangle\}$. La composición y granularidad de $\mathcal{C}_{\text{fg}}$ puede ser muy variada, y depende de la aplicación particular. A modo de ejemplo, en la Fig. 2.1 se muestra una imagen de la base de datos MS-COCO [14], junto con una representación gráfica de su máscara de segmentación *ground truth*.

Figura 2.1: Ejemplo de una imagen de MS-COCO [14] junto a su máscara de segmentación semántica *ground truth*.

2.2 Supervisión

Las estrategias que actualmente constituyen el estado del arte para segmentación semántica corresponden a modelos de *deep learning* entrenados en régimen de supervisión completa o *fully-supervised*, es decir, utilizando una base de datos compuesta por imágenes anotadas con máscaras de segmentación *ground truth*. Para aliviar el alto costo que conlleva generar manualmente bases de datos con este tipo de anotación, recientemente se han propuesto estrategias para entrenar modelos de segmentación utilizando distintos tipos de supervisión más débil, destacando el caso con supervisión a nivel de imagen, es decir, sin ningún tipo de información de localización. El tipo de supervisión más estudiado para este problema consiste en etiquetas de clasificación, de modo que para cada imagen $\mathbf{I}_n$ de la base de datos se cuenta únicamente con el conjunto $\mathcal{C}_n^{\text{fg}}$ de categorías de objetos relevantes presentes en la imagen, con $\mathcal{C}_n^{\text{fg}} \subseteq \mathcal{C}_{\text{fg}}$.

En el presente trabajo se estudia utilizar una forma alternativa de supervisión débil a nivel de imagen, que consiste en descripciones en lenguaje natural. Concretamente, se busca entrenar un modelo de segmentación semántica utilizando para su supervisión una base de datos de la forma:

$$\mathcal{D} = \{(\mathbf{I}_n, \{S_{n,m}\}_{m=1}^{M_n})\}_{n=1}^{N}, \tag{2.1}$$

de modo que cada una de las N imágenes $\mathbf{I}_n$ disponibles se encuentra anotada únicamente con un conjunto de $M_n \geqslant 1$ descripciones $S_{n,m}$.

2.3 Métricas de Desempeño

2.3.1 Segmentación Semántica

La métrica de desempeño más utilizada en la literatura para evaluar modelos de segmentación semántica, o sus variantes con supervisión débil, corresponde a la intersección sobre unión media o *mean Intersection-over-Union* (mIoU), definida como:

$$mIoU = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{|\mathcal{P}_c^{\text{pred}} \cap \mathcal{P}_c^{\text{gt}}|}{|\mathcal{P}_c^{\text{pred}} \cup \mathcal{P}_c^{\text{gt}}|}, \tag{2.2}$$

donde $\mathcal{P}_c^{\text{pred}}$ es el conjunto de pixeles que el modelo etiqueta con la categoría c, $\mathcal{P}_c^{\text{gt}}$ es el conjunto de pixeles etiquetados con la categoría c en las máscaras de segmentación *ground truth*, y $\mathcal{C}_{\text{seg}}$ es el conjunto de categorías, incluyendo la categoría "fondo". Se suele utilizar esta versión de la intersección sobre unión, que promedia los resultados obtenidos separadamente para cada clase, para evitar que categorías más comunes dominen por sobre otras menos frecuentes.

Otras métricas populares relacionadas con mIoU son precisión o *precision*, y exhaustividad o *recall*, definidas como:

$$Precision = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{|\mathcal{P}_c^{\text{pred}} \cap \mathcal{P}_c^{\text{gt}}|}{|\mathcal{P}_c^{\text{pred}}|}, \text{y} \tag{2.3}$$

$$Recall = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{|\mathcal{P}_c^{\text{pred}} \cap \mathcal{P}_c^{\text{gt}}|}{|\mathcal{P}_c^{\text{gt}}|}, \tag{2.4}$$

usando la misma notación que en (2.2), y nuevamente promediando los resultados para cada clase. Intuitivamente, la precisión indica la frecuencia con la que las etiquetas predichas por el modelo para una determinada categoría son correctas, mientras que la exhaustividad indica la proporción de pixeles correspondientes a una determinada categoría que el modelo fue capaz de etiquetar correctamente.

2.3.2 Predicción de Etiquetas de Clasificación

Durante el desarrollo y evaluación del modelo propuesto, se considera un sub-problema correspondiente a predecir las categorías de objetos relevantes presentes en una imagen a partir de sus descripciones en lenguaje natural. Para evaluar la calidad de las etiquetas de clasificación predichas, se utilizan métricas similares de *precision*, *recall*, e IoU, definidas en este caso como:

$$ImagePrecision = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{|\mathcal{I}_c^{\text{pred}} \cap \mathcal{I}_c^{\text{gt}}|}{|\mathcal{I}_c^{\text{pred}}|}, \tag{2.5}$$

$$ImageRecall = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{|\mathcal{I}_c^{\text{pred}} \cap \mathcal{I}_c^{\text{gt}}|}{|\mathcal{I}_c^{\text{gt}}|}, \tag{2.6}$$

$$ImageIoU = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{|\mathcal{I}_c^{\text{pred}} \cap \mathcal{I}_c^{\text{gt}}|}{|\mathcal{I}_c^{\text{pred}} \cup \mathcal{I}_c^{\text{gt}}|}, \tag{2.7}$$

donde $\mathcal{I}_c^{\text{pred}}$ corresponde al conjunto de índices de las imágenes que el modelo predice que contienen algún objeto de la categoría c, mientras que $\mathcal{I}_c^{\text{gt}}$ es el conjunto de índices de las imágenes que contienen algún objeto de la clase c según las etiquetas de clasificación *ground truth*.

Adicionalmente, se utiliza la siguiente variación de *recall* que incorpora información a nivel de pixel:

$$PixelRecall = \frac{1}{|\mathcal{C}_{\text{seg}}|} \sum_{c \in \mathcal{C}_{\text{seg}}} \frac{\sum_{n \in \mathcal{I}_c^{\text{pred}} \cap \mathcal{I}_c^{\text{gt}}} |\mathcal{P}_{n,c}^{\text{gt}}|}{|\mathcal{P}_c^{\text{gt}}|}. \tag{2.8}$$

En este caso, el numerador para cada categoría c corresponde a la suma del número de pixeles etiquetados con esta clase en las máscaras de segmentación *ground truth*, para cada imagen n a la que el modelo asigna correctamente la categoría c. $\mathcal{P}_{n,c}^{\text{gt}}$ es el conjunto de pixeles de la imagen n etiquetados con la categoría c en el *ground truth*, mientras que $\mathcal{P}_c^{\text{gt}}$, al igual que en (2.2) - (2.4), es el conjunto de todos los pixeles etiquetados con la clase c en los *ground truth* de la base de datos, i.e., $\mathcal{P}_c^{\text{gt}} = \bigcup_n \mathcal{P}_{n,c}^{\text{gt}}$. Esta métrica permite aproximar mejor el impacto de asignar correctamente una categoría de clasificación a una imagen para efectos del modelo de segmentación, al tomar en cuenta el área que ocupa esa categoría en la imagen.

Capítulo 3

Trabajos Relacionados

3.1 Herramientas Generales

En esta sección, se presenta una revisión de diversas herramientas generales utilizadas en el desarrollo del método propuesto, o que son relevantes para entender otros trabajos relacionados que se presentan en secciones posteriores. En primer lugar, se realiza una breve descripción de tres familias de redes neuronales artificiales ampliamente utilizadas en aplicaciones recientes de procesamiento de imágenes y lenguaje, que corresponden a Redes Neuronales Convolucionales, Redes Neuronales Recurrentes, y Transformers. Posteriormente, se describe el concepto de *word embedding* y su aplicación a modelos de procesamiento de lenguaje, y luego se definen los modelos gráficos *Conditional Random Fields*, específicamente en el contexto de segmentación semántica de imágenes. Finalmente, se presentan dos componentes fundamentales del módulo de procesamiento de texto propuesto: en primer lugar, se define el concepto de un grafo de escena, y se describe el método utilizado para generar este tipo de representación a partir de descripciones en lenguaje natural. Posteriormente, se presenta la base de datos WordNet, que se utiliza para establecer relaciones entre categorías semánticas relevantes y los objetos del grafo de escena generado, y se describe el modelo utilizado para asignar palabras en contexto a su significado específico dentro de esta base de datos.

3.1.1 Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNNs, por sus siglas en inglés) son un tipo de red neuronal artificial que actualmente constituye el estado del arte en un sinnúmero aplicaciones modernas de procesamiento de imágenes y visión computacional. Forman parte de una familia de modelos de aprendizaje de máquinas conocidos como modelos de *deep learning*, que se caracterizan por sus estructuras jerárquicas compuestas por múltiples capas, que les permiten aprender directamente desde datos crudos mediante la integración progresiva de patrones complejos a partir de otros más simples.

En esta sección se realiza una breve descripción de sus componentes principales y sus arquitecturas más comunes, así como del procedimiento general de entrenamiento utilizado para optimizar los parámetros de las CNNs y otras redes neuronales. Finalmente, se revi-

san algunas técnicas complementarias comúnmente utilizadas en conjunto con este tipo de modelos.

Capas *fully-connected*

Las capas *fully-connected* siguen el diseño de las redes neuronales artificiales más tradicionales, tales como el Multilayer Perceptron (MLP). Formalmente, dado un vector de entrada $\boldsymbol{x} \in \mathbb{R}^{d_{\text{in}}}$, una capa *fully-connected* computa su salida $\boldsymbol{h} \in \mathbb{R}^{d_{\text{out}}}$ como:

$$\boldsymbol{h} = f(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}), \tag{3.1}$$

que corresponde a una transformación afín definida por la matriz de **pesos** $\boldsymbol{W} \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$ y el vector de ***bias*** $\boldsymbol{b} \in \mathbb{R}^{d_{\text{out}}}$, seguida de una **función de activación** f.

Tanto $\boldsymbol{W}$ como $\boldsymbol{b}$ son parámetros aprendidos del modelo, mientras que las funciones de activación f se incluyen para posibilitar el aprendizaje de relaciones no lineales. Históricamente, las funciones de activación más comunes para capas intermedias de la red han correspondido a tangente hiperbólica y función logística, pero en los últimos años estas han sido desplazadas por las funciones ***Rectified Linear Unit*** (ReLU), definidas como $\text{ReLU}(x) = \max\{0, x\}$, debido a que presentan mejores propiedades numéricas para el entrenamiento de modelos más profundos. Por otra parte, las funciones de activación para la última capa de la red se escogen normalmente de forma de permitir la interpretación de sus salidas como una distribución de probabilidad sobre algún espacio de interés. Las salidas de la red antes de esta última etapa de normalización se conocen con el nombre de ***logits***.

Los valores tanto del vector de entrada como del de salida pueden interpretarse como niveles de activación de neuronas artificiales, conectadas entre sí por pesos que representan sinapsis. Estas capas deben su nombre al hecho de que el cómputo de las activaciones de cada neurona de salida depende de las activaciones de todas las neuronas de entrada. Esto le permite al modelo considerar información global al momento de generar sus predicciones, pero también aumenta rápidamente el número de parámetros y cómputos necesarios en la medida en que se incrementa la cantidad y tamaño de este tipo de capas. Por esta razón, en el contexto de las CNNs las capas *fully-connected* se utilizan principalmente para generar las predicciones finales del modelo, mientras que las capas iniciales e intermedias de la red utilizan patrones de conectividad locales más eficientes, como se describe a continuación.

Capas Convolucionales

Las capas convolucionales constituyen la componente fundamental de las CNNs. En el caso usual de aplicación a imágenes, tanto la entrada como la salida de este tipo de capas corresponden a tensores (arreglos de números en múltiples dimensiones, que extienden las nociones de vector y matriz). Más concretamente, dada una entrada $\mathbf{I}$ con dimensiones $(c_{\text{in}}, h_{\text{in}}, w_{\text{in}})$, donde c_{in} representa el número de canales y $(h_{\text{in}}, w_{\text{in}})$ sus dimensiones espaciales, una capa convolucional genera una salida $\mathbf{S}$ con dimensiones $(c_{\text{out}}, h_{\text{out}}, w_{\text{out}})$. Esta transformación se define por un conjunto de c_{out} filtros o *kernels* convolucionales, cada uno con dimensiones $(c_{\text{in}}, h_{filter}, w_{filter})$, que operan sobre la entrada para computar cada canal de la salida como:

$$\boldsymbol{S}_{k_{\text{out}}} = \sum_{k_{\text{in}}=1}^{c_{\text{in}}} \boldsymbol{W}_{k_{\text{out}},k_{\text{in}}} \star \boldsymbol{I}_{k_{\text{in}}}, \tag{3.2}$$

donde $\boldsymbol{I}_{k_{\text{in}}}$ es el canal k_{in} de la entrada $\mathbf{I}$, $\boldsymbol{S}_{k_{\text{out}}}$ es el canal k_{out} de la salida $\mathbf{S}$, $\boldsymbol{W}_{k_{\text{out}},k_{\text{in}}}$ es el canal k_{in} del k_{out}-ésimo filtro convolucional $\mathbf{W}_{k_{\text{out}}}$, y $\star$ denota correlación cruzada bidimensional, que es la operación implementada normalmente por librerías de *deep learning*. Dado que en este contexto los pesos de los filtros son normalmente entrenados, la inversión o no de sus valores, que diferencia la correlación cruzada de la convolución, suele ser irrelevante, por lo que ambos nombres se utilizan de forma equivalente.

Gráficamente, la correlación cruzada puede entenderse como desplazar cada filtro a través de las dimensiones espaciales de un canal de la entrada, y en cada posición utilizar sus pesos para computar una suma ponderada sobre los valores locales de la entrada, lo que se ilustra en la Fig. 3.1. Los valores obtenidos para cada canal de la entrada se suman para computar el valor final en la salida. De esta forma, las activaciones de las neuronas en la salida pueden interpretarse como detecciones de patrones locales definidos por los pesos de los filtros de la red. El uso de múltiples filtros permite identificar distintos tipos de patrones, que se codifican en los diferentes canales de la salida.

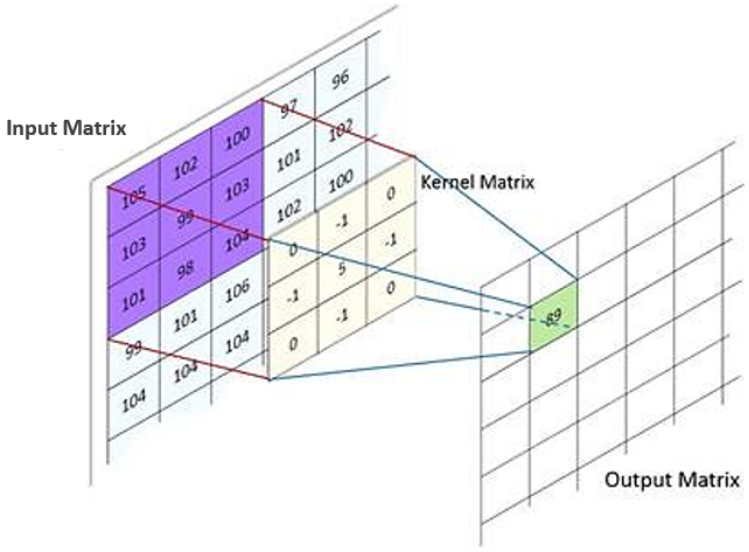


Figura 3.1: Visualización de la operación de convolución bidimensional discreta. Versión original extraída de [26].

Los valores de la entrada que influyen sobre el cómputo de la activación para una determinada neurona de salida constituyen lo que se conoce como su **campo receptivo**. Al considerar únicamente conexiones locales, la convolución permite reducir drásticamente el número de cómputos y parámetros en comparación con una capa *fully-connected*. Adicionalmente, estos mismos pesos son compartidos a través de toda la entrada, lo que reduce aún más el número de parámetros, sirve para minimizar el riesgo de sobre-ajuste del modelo, y le otorga un cierto grado de invariancia a traslaciones, ya que los mismos patrones pueden ser detectados independientemente del lugar de la entrada en el que aparecen. Estas características contribuyen a que las redes convolucionales sean capaces de extraer información visual de manera flexible y eficiente.

Al igual que en el caso *fully-connected*, las capas convolucionales suelen terminar en una función de activación (típicamente ReLU), y pueden incluir un término de *bias*, normalmente

definido por canal. Las dimensiones espaciales de la salida $(h_{\text{out}}, w_{\text{out}})$ dependen de las dimensiones espaciales tanto de la entrada como de los filtros. Generalmente, se utilizan técnicas de ***padding*** para poder aplicar los filtros convolucionales cerca de los bordes de la entrada, que consisten en extender artificialmente esta matriz, típicamente agregando ceros. Adicionalmente, es posible modificar el paso o *stride* de la convolución, que corresponde a la magnitud del desplazamiento de los filtros entre cada cómputo consecutivo de la salida. Utilizar *strides* mayores que 1 resulta en una reducción de las dimensiones espaciales respecto de la entrada.

Capas de *Pooling*

Finalmente, las capas de *pooling* permiten reemplazar pequeñas vecindades de valores en cada canal de la entrada por algún estadístico de los mismos (usualmente el máximo), como se ilustra en la Fig. 3.2. Esto ayuda a prevenir el sobre-ajuste del modelo, y le otorga invariancia frente a pequeñas traslaciones o deformaciones de la entrada. Este tipo de capas suele aplicarse con un *stride* superior a 1, por lo que también sirven para reducir el uso de memoria de las representaciones internas de la red durante el entrenamiento y las pruebas.

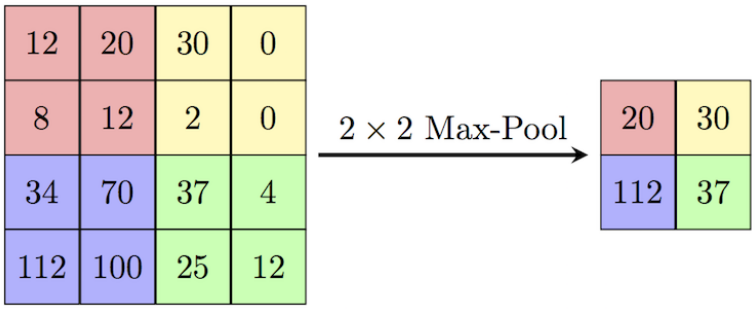


Figura 3.2: Ejemplo de aplicación de una capa de *max pooling* con filtro de 2×2 y *stride* 2. Extraída de [27].

Arquitectura General

La Fig. 3.3 muestra esquemáticamente la arquitectura genérica de una red neuronal convolucional profunda aplicada al problema de clasificación, que constituye uno de los dominios de aplicación más fundamentales para este tipo de modelo. Este esquema general forma la base de numerosas redes convolucionales desarrolladas en los últimos años para diversas aplicaciones de procesamiento de imágenes, algunas de las cuales se describen en secciones posteriores.

La entrada de las CNNs consiste típicamente de una imagen, representada como un tensor con dimensiones espaciales $(h_{\text{in}}, w_{\text{in}})$ y 3 canales de profundidad (en el caso estándar de imágenes a color en el espacio RGB). Esta entrada es procesada por una serie de bloques convolucionales, cada uno compuesto por distintas combinaciones de capas convolucionales y de *pooling*, generando una pirámide de representaciones tensoriales intermedias conocidas como **mapas de características** o *feature maps*. Típicamente, los bloques convolucionales reducen gradualmente la resolución espacial de los mapas de características, a la vez que aumentan su número de canales. La aplicación de sucesivas capas convolucionales aumenta también el campo receptivo efectivo de las neuronas de las capas ocultas. Esta estructura jerárquica le

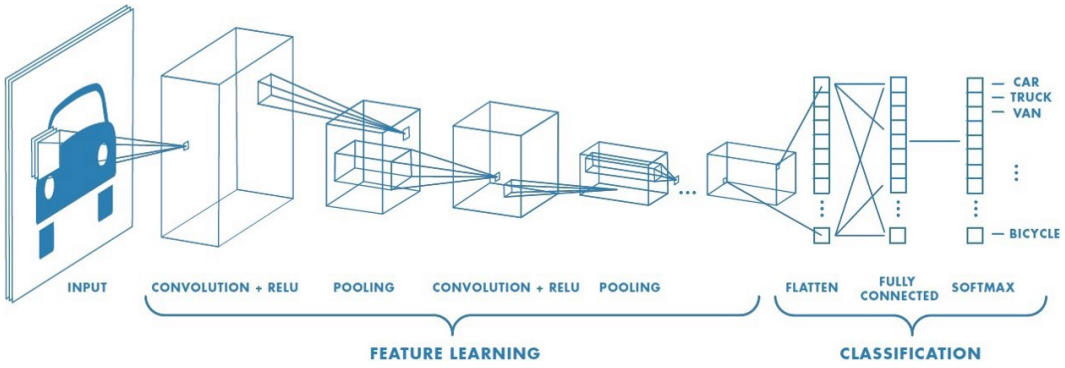


Figura 3.3: Arquitectura general de una red convolucional típica aplicada al problema de clasificación. Extraída de [28].

permite al modelo integrar patrones progresivamente más complejos, hasta obtener mapas de características que codifican información abstracta de alto nivel sobre los contenidos de la imagen. En el caso de clasificación, las redes normalmente terminan con una operación de estiramiento o de *pooling* global que transforma el último mapa de características en un vector unidimensional, seguida de una o más capas *fully-connected*.

Arquitecturas importantes

Uno de los primeros modelos de este tipo en demostrar resultados significativamente superiores a los de cualquier otra técnica existente hasta el momento para clasificación en la base de datos ImageNet [29] fue **AlexNet** [30]. Esta red se compone de cinco capas convolucionales, seguidas de tres capas *fully-connected*, y ya utilizaba ReLU como función de activación, además de *dropout* y *max pooling*, todas técnicas comunes en CNNs modernas.

Una variante ampliamente utilizada hasta el día de hoy corresponde a la arquitectura **VGGNet** [31]. Estos modelos aumentan el número de capas con respecto a AlexNet, mostrando que esto facilita el aprendizaje de patrones más complejos. También reducen el tamaño de los filtros convolucionales, de 11×11 y 5×5 a 3×3, disminuyendo significativamente el número de parámetros, y demostrando que el aumento de la profundidad del modelo es un reemplazo efectivo para el tamaño de los filtros. Todas las versiones propuestas en el *paper* original reciben imágenes de entrada en una resolución fija de 224×224 pixeles, y luego de cinco bloques convolucionales aplican una operación de *flattening*, en la que los valores del último mapa de características se estiran para formar un vector, que sirve de entrada para tres capas *fully-connected*.

En [32] se propuso **GoogLeNet**, también llamada Inception-V1, que se caracteriza por el uso de bloques compuestos de múltiples capas convolucionales y de *pooling* en paralelo. Las distintas ramas de cada bloque utilizan filtros de distinto tamaño, lo que les permite capturar información espacial a distintas escalas, que luego se concatenan para formar la salida final. En lugar de una operación de estiramiento, este modelo utiliza una operación de *pooling* global para convertir el último mapa de características en un vector, que sirve de entrada para un clasificador *fully-connected*. Esta estrategia permite reducir drásticamente el número de parámetros, manteniendo la calidad de clasificación.

Por otra parte, en [33] se propone la arquitectura **ResNet**, caracterizada por el uso de conexiones residuales que combinan las entradas y salidas de cada bloque convolucional mediante una suma elemento a elemento antes de aplicar el bloque siguiente. Esto permite reducir la distancia efectiva entre la salida del modelo y las primeras capas, lo que facilita la propagación de los gradientes durante el entrenamiento, posibilitando la optimización de modelos más profundos. Estas arquitecturas han sido altamente influyentes en el desarrollo de modelos convolucionales posteriores, permitiendo acelerar la convergencia durante el entrenamiento y aumentar la profundidad de las redes.

Entrenamiento

El proceso de aprendizaje o entrenamiento de una red neuronal artificial consiste en ajustar sus pesos y *biases* de modo de mejorar la calidad de sus predicciones. Esto se modela en general como un problema de optimización, en el que se minimiza una **función de pérdida** que cuantifica el nivel de error de las salidas del modelo, para un conjunto determinado de datos de entrenamiento. Existen diversos métodos para realizar esta optimización, con distintas propiedades y ventajas, pero todos los más usados tienen en común el hecho de ser algoritmos iterativos que utilizan los gradientes de la función de pérdida con respecto de los parámetros del modelo para las actualizaciones.

El método más sencillo de optimización, y uno de los más utilizados, corresponde a ***Stochastic Gradient Descent*** (SGD), que computa los gradientes de la función de pérdida a partir de pequeños subconjuntos aleatorios de los datos de entrenamiento denominados ***minibatches***. Esto permite obtener una aproximación de la dirección de máximo descenso local en el espacio de los parámetros del modelo en cada iteración. Las actualizaciones de los pesos se realizan avanzando en esta dirección, con una magnitud dada por un parámetro escalar denominado **tasa de aprendizaje** o *learning rate*.

Otra componente fundamental para el entrenamiento de modelos con múltiples capas es el algoritmo de ***backpropagation***, que permite computar los gradientes para todos los parámetros de la red de manera eficiente. Este algoritmo aprovecha la regla de la cadena para computar los gradientes de cada capa utilizando los gradientes ya computados para capas anteriores, comenzando desde la salida de la red y avanzando iterativamente hasta la entrada, evitando así cómputos redundantes.

Técnicas Complementarias

En esta sección se revisan brevemente algunas técnicas comunes utilizadas en conjunto con redes neuronales convolucionales, y modelos de *deep learning* en general.

Debido a la enorme capacidad de la mayoría de las redes neuronales profundas, uno de los principales desafíos que surgen al entrenar este tipo de modelo consiste en prevenir que estos se sobre-ajusten a los datos de entrenamiento, llevando a un desempeño deficiente sobre datos nuevos. Para evitar este problema, se han propuesto diversas estrategias de regularización, que pueden interpretarse como mecanismos para favorecer el aprendizaje de modelos más simples, lo que ayuda a mejora sus habilidades de generalización. Una de las estrategias de regularización más comunes consiste en incorporar un término adicional a la función de pérdida que penaliza la norma L^1 o L^2 de los parámetros del modelo, lo que se conoce

normalmente como ***weight decay***.

Otra estrategia popular de regularización es ***dropout***, que consiste en excluir de manera aleatoria un subconjunto de las neuronas del modelo cada vez que se procesa un dato de entrenamiento. Esta técnica previene la co-adaptación excesiva de neuronas de una misma capa, promoviendo el aprendizaje de características más robustas. El *dropout* es más comúnmente utilizada en capas *fully-connected*. Por otra parte, ***batch normalization*** consisten en estandarizar las salidas de cada capa, antes de aplicar la capa siguiente. En la práctica, esto resulta en una convergencia más rápida y estable, además de introducir un efecto adicional de regularización.

En general, uno de los métodos más efectivos para mejorar la generalización de un modelo de aprendizaje de máquinas consiste en aumentar la cantidad de datos de entrenamiento. Sin embargo, esto es costoso de conseguir en la mayoría de los casos, por lo que la cantidad de ejemplos disponibles suele ser limitada. Una alternativa sencilla que permite aproximar este efecto consiste en generar ejemplos artificiales a partir de los datos de entrenamiento disponibles, mediante la aplicación de transformaciones a la entrada que preserven la validez de las etiquetas, lo que se conoce como aumentación de datos o ***data augmentation***. Por ejemplo, en el caso de clasificación de imágenes, algunas transformaciones comunes que preservan la categoría de un ejemplo de entrenamiento incluyen pequeñas rotaciones, deformaciones, recortes, inversiones horizontales, adición de ruido, o aplicación filtros de color, entre otras.

Finalmente, la técnica de transferencia de conocimiento o ***transfer learning*** consiste en entrenar un modelo para un determinado problema, y luego aprovechar el conocimiento adquirido por el modelo para aplicarlo en un problema diferente, pero relacionado. Esto permite reducir los tiempos de entrenamiento en el problema final, mejorar en muchos casos la calidad de los resultados, y reducir el sobre-ajuste de modelos de alta capacidad cuando se cuenta con pocos datos en el dominio de destino. En el caso de redes neuronales convolucionales, la metodología de *transfer learning* más común consiste en pre-entrenar el modelo en un problema para el que se cuenta con gran cantidad de datos (típicamente para clasificación en ImageNet [29]), y luego reemplazar solo las últimas capas de la red para adaptarla al problema de destino. Esto se justifica debido a que las capas iniciales del modelo suelen aprender a detectar patrones sencillos, que generalizan bien a distintos dominios y aplicaciones, mientras que las capas finales codifican información de más alto nivel, que está generalmente más ligada a la aplicación particular. El modelo puede aplicarse directamente, o re-entrenarse en el dominio de destino bajo un régimen conocido como *fine-tuning*, en el que se entrena solo un subconjunto de los parámetros de la red, o se utiliza una tasa de aprendizaje reducida.

3.1.2 Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNNs, por su sigla en inglés) son un tipo de redes neuronales artificiales, ampliamente utilizadas para modelar datos secuenciales, tales como texto, audio, videos, o series temporales. La característica principal de este tipo de modelo es la presencia de un estado interno que actúa como memoria, permitiendo modelar la dependencia de los datos en cada instante de tiempo, con respecto a los datos observados previamente. La Fig. 3.4 muestra el esquema general de una red recurrente básica. En cada instante de tiempo t, el modelo recibe una entrada $\boldsymbol{x}_t$, además del estado interno del instante anterior

$\boldsymbol{h}_{t-1}$, actualizando el estado de acuerdo a la entrada, y produciendo una salida $\boldsymbol{o}_t$.

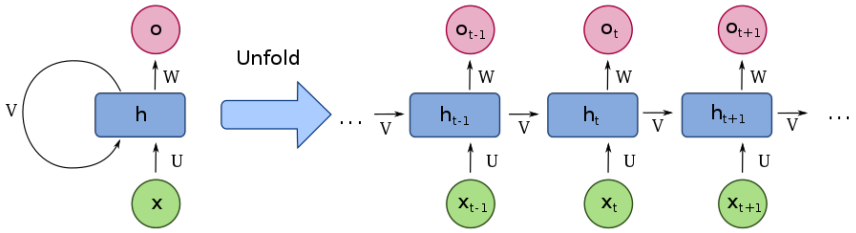


Figura 3.4: Esquema general de una red recurrente básica. A la izquierda se muestran las conexiones recurrentes de forma explícita, mientras que a la derecha se muestran sus recurrencias "desenrolladas" en el tiempo. Extraída de [34].

Las redes recurrentes se entrenan típicamente utilizando un esquema similar al descrito para las redes convolucionales, pero propagando los gradientes de una función de pérdida a través del tiempo, lo que se conoce como *backpropagation through time*. Debido a que los parámetros de la red son los mismos en cada instante, las RNNs más sencillas son particularmente propensas a experimentar problemas de crecimiento o disminución exponencial de sus gradientes a medida que se retrocede en el tiempo, además de presentar dificultades para modelar relaciones entre instantes alejados de tiempo.

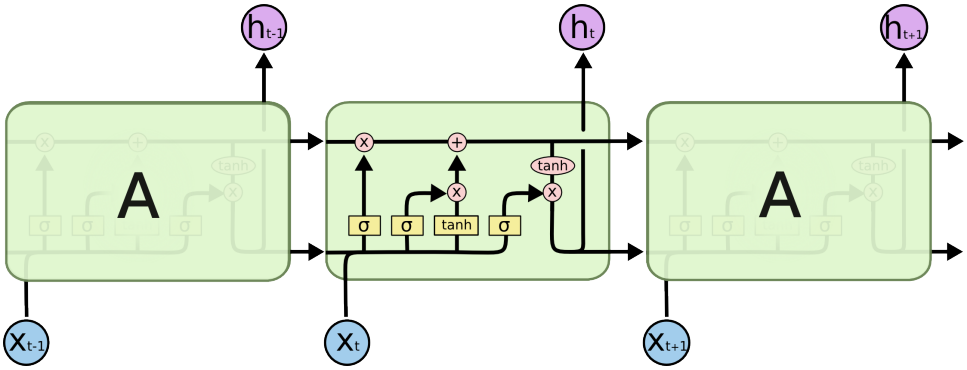


Figura 3.5: Esquema general de una red *Long Short-Term Memory* (LSTM) [35]. Extraída de [36].

Una de las variantes más utilizadas para enfrentar estas limitaciones son las redes *Long Short-Term Memory* (LSTM) [35], que incorporan una memoria separada del estado interno, además de "compuertas" que permiten controlar operaciones de escritura y lectura sobre la memoria, condicionadas a la entrada y al estado actual. La Fig. 3.5 muestra el esquema general de una LSTM típica, cuyo comportamiento está gobernado por las siguientes ecuaciones:

$$\boldsymbol{f}_t = \sigma\left(\boldsymbol{W}^{(f)} \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}^{(f)}\right) \tag{3.3}$$

$$\boldsymbol{i}_t = \sigma\left(\boldsymbol{W}^{(i)} \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}^{(i)}\right) \tag{3.4}$$

$$\boldsymbol{o}_t = \sigma\left(\boldsymbol{W}^{(o)} \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}^{(o)}\right) \tag{3.5}$$

$$\tilde{\boldsymbol{C}}_t = \tanh\left(\boldsymbol{W}^{(C)} \cdot [\boldsymbol{x}_t; \boldsymbol{h}_{t-1}] + \boldsymbol{b}^{(C)}\right) \tag{3.6}$$

$$\boldsymbol{C}_t = \boldsymbol{f}_t \odot \boldsymbol{C}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{C}}_t \tag{3.7}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh\left(\boldsymbol{C}_t\right) \tag{3.8}$$

donde $[\boldsymbol{x}_t; \boldsymbol{h}_{t-1}]$ es la concatenación del vector de entrada y el estado anterior, que sirve de entrada para distintas capas *fully-connected*. Esto permite obtener 3 "compuertas" ($\boldsymbol{f}_t$, $\boldsymbol{i}_t$ y $\boldsymbol{o}_t$), que utilizan una sigmoide como función de activación para mapear sus valores al intervalo $[0, 1]$, y un "candidato a memoria" $\tilde{\boldsymbol{C}}_t$. $\odot$ denota multiplicación elemento a elemento, de modo que la compuerta de olvido $\boldsymbol{f}_t$ controla los valores de la memoria inicial $\boldsymbol{C}_{t-1}$ que deben ser olvidados ($\boldsymbol{f}_t = 0$) o mantenidos ($\boldsymbol{f}_t = 1$), mientras que la compuerta de entrada $\boldsymbol{i}_t$ controla los valores de $\tilde{\boldsymbol{C}}_t$ que deben escribirse en la memoria. La compuerta de salida $\boldsymbol{o}_t$ determina qué valores de la memoria actualizada deben considerarse para generar el nuevo estado, a partir del cual se computa normalmente la salida del modelo.

Se han propuesto numerosas variantes de esta arquitectura en la literatura, que presentan distintos balances de velocidad, tamaño, facilidad de entrenamiento, y capacidad de representación, como por ejemplo las *gated recurrent units* (GRU) [37] o *simple recurrent units* (SRU) [38]. Otras variantes han utilizado recurrencias bidireccionales, de modo que la predicción en un determinado instante pueda utilizar información contextual tanto de eventos ocurridos anteriormente en la secuencia, como después.

3.1.3 Transformers

Los Transformers [39] son un tipo de arquitectura neuronal propuesta recientemente para modelar datos secuenciales, y que han fijado numerosos récords en aplicaciones de procesamiento de lenguaje. Estos modelos prescinden de conexiones recurrentes utilizadas en RNNs, y en su lugar utilizan bloques basados en mecanismos de auto-atención.

Una función de atención recibe un conjunto de valores, cada uno asociado a una llave, y una consulta (siendo valores, llaves, y consultas todos vectores), y permite computar una salida que corresponde a una suma ponderada sobre los valores, donde los pesos están dados por una función de compatibilidad entre la consulta y las llaves correspondientes. En el caso de la auto-atención, tanto las consultas, como los valores y llaves se obtienen a partir de la misma secuencia de entrada, lo que se utiliza para obtener salidas correspondientes a versiones "enriquecidas con contexto" de los vectores de entrada.

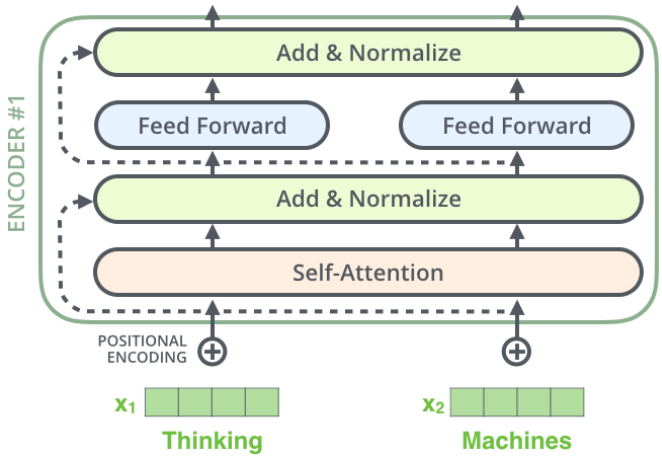


Figura 3.6: Bloque fundamental del Transformer [39]. Extraída de [40].

La Fig. 3.6 muestra esquemáticamente el bloque fundamental del Transformer, que con-

siste en una operación de auto-atención seguida de una red *fully-connected* que se aplica posición a posición. Ambas operaciones incorporan conexiones residuales, sumando la entrada a la salida, seguida de normalización. En la práctica, la primera etapa consiste de múltiples operaciones de auto-atención en paralelo, lo que permite al modelo extraer distintos tipos de información desde diferentes puntos de la secuencia. Múltiples copias de este bloque se concatenan secuencialmente para formar un *encoder* que genera representaciones más útiles de la secuencia de entrada, mientras que un *decoder* formado por bloques similares se encarga de generar la secuencia de salida. Dado que la operación de atención es en principio deslocalizada, se incorporan *position embeddings* a la entrada del modelo para codificar información de posición.

Las principales ventajas de este modelo consisten en su mayor capacidad de paralelización en comparación con redes recurrentes, además de permitir modelar dependencias de largo alcance de forma más directa.

3.1.4 *Word Embeddings*

Los *word embeddings* son representaciones vectoriales de palabras en un espacio de alta dimensionalidad, usualmente entrenados para conservar relaciones semánticas relevantes, y forman parte fundamental de prácticamente todas las aplicaciones modernas de procesamiento de lenguaje. Estos vectores pueden ser entrenados desde cero para cada aplicación particular, pero en general resulta beneficioso comenzar con representaciones pre-entrenadas en grandes bases de texto, lo que facilita el entrenamiento de los modelos. Esto puede entenderse como un esquema particular de *transfer learning* aplicable a modelos de lenguaje.

Uno de los primeros y más conocidos tipos de *word embeddings* es **Word2vec** [41]. Estas representaciones son generadas por una red neuronal de dos capas, entrenada de manera no-supervisada para dos problemas complementarios: predecir la palabra más probable dada su contexto, y predecir el contexto dada una palabra. Estos objetivos resultan en representaciones similares para palabras que pueden aparecer en contextos similares (e.g., las palabras "*boat*" y "*ship*" tienen representaciones cercanas en el espacio vectorial aprendido), además de capturar relaciones semánticas más complejas. Un ejemplo frecuente es que la operación aritmética "*king*" − "*man*" + "*woman*" en el espacio vectorial resulta en un vector cercano a la representación para "*queen*". Otro enfoque popular corresponde a **GloVe** [42], que aprende vectores por palabra utilizando la matriz de co-ocurrencia global, extendiendo el contexto local utilizado por Word2vec.

Una limitación importante de ambas representaciones descritas anteriormente es que, al generar vectores estáticos para cada palabra, son incapaces de manejar polisemia (es decir, palabras que pueden tener más de un significado, dependiendo del contexto). Para enfrentar este problema, más recientemente han ganado popularidad representaciones contextuales, que permiten computar *embeddings* vectoriales para cada palabra de forma dinámica, dentro de un contexto específico.

Uno de los primeros modelos en utilizar esta formulación fue **ELMo** [43], que utiliza una LSTM bi-direccional para generar vectores con información de contexto, a partir de representaciones individuales definidos en términos de los caracteres de cada palabra. Esto último le permite al modelo manejar palabras fuera del vocabulario de entrenamiento. Más

recientemente, en [44] se propuso **BERT**, que utiliza una arquitectura basada en Transformers para generar representaciones contextualizadas más flexibles. Las representaciones en este caso se generan a nivel de sub-palabras (e.g., la palabra "*playing*" se separaría en dos *tokens*, "*play*" e "*##ing*"), lo que permite reducir drásticamente el tamaño del vocabulario, y manejar de mejor manera palabras no vistas durante el entrenamiento.

3.1.5 *Conditional Random Fields*

Un campo aleatorio condicional o *conditional random field* (CRF) [45] es un tipo de modelo probabilístico, típicamente utilizado para generar predicciones estructuradas sobre secuencias de datos. Dada una secuencia de observaciones $\mathbf{I}$, una secuencia de variables aleatorias $\mathbf{X}$, y un grafo no dirigido $G = (V, E)$, tales que $\mathbf{X} = (X_v)_{v \in V}$ (i.e., los elementos de $\mathbf{X}$ son indexados por los vértices de G), y las variables de $\mathbf{X}$ condicionadas en $\mathbf{I}$ obedecen la propiedad de Markov con respecto a G, entonces un CRF permite modelar la probabilidad de $\mathbf{X}$ condicionada a las observaciones $\mathbf{I}$, $P(\mathbf{X}|\mathbf{I})$. La propiedad de Markov para grafos puede expresarse en este caso como:

$$P(X_v|\mathbf{I}, X_w, w \neq c) = P(X_v|\mathbf{I}, X_w, w \sim v), \tag{3.9}$$

donde $w \sim v$ significa que w y v están conectadas por una arista en G. Es decir, la probabilidad para X_v condicionada a $\mathbf{I}$ depende solo de los elementos de $\mathbf{X}$ cuyos índices son vecinos de v en G.

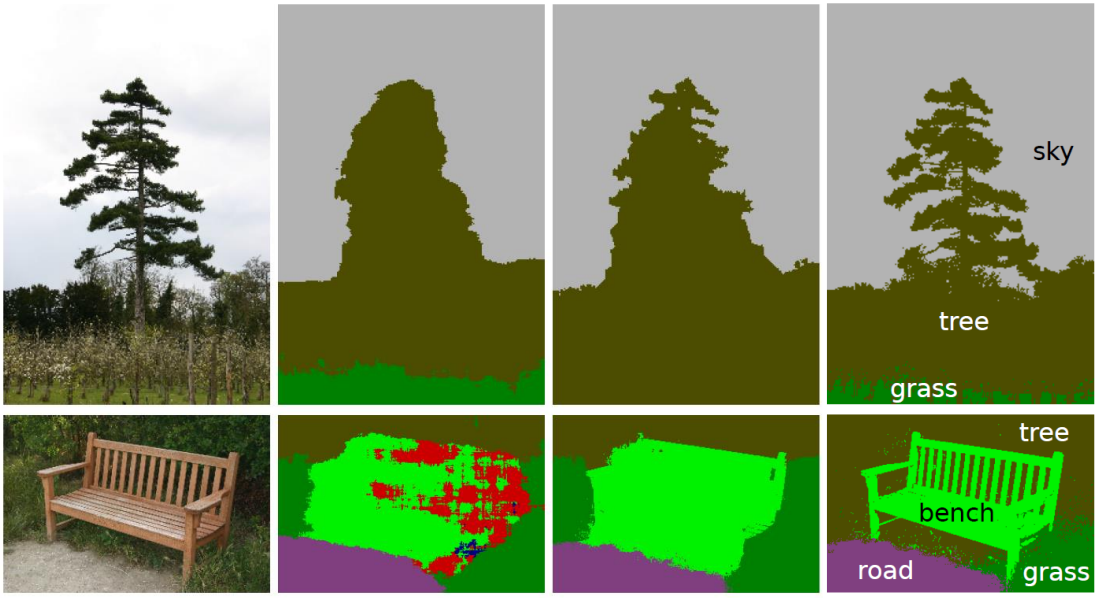
En el caso particular de segmentación semántica, $\mathbf{I} = (I_v)_{v \in V}$ corresponde a los $|V|$ pixeles de una imagen, y $\mathbf{X} = (X_v)_{v \in V}$ representa las etiquetas asignadas a los pixeles respectivos, con cada variable aleatoria tomando valores en un conjunto fijo de etiquetas $\mathcal{C}_{\text{seg}}$. Un enfoque común consiste en utilizar estimación del máximo a posteriori (MAP) sobre el CRF, considerando un potencial unario dado por predicciones independientes para cada pixel, y potenciales binarios que favorecen la asignación de etiquetas homogéneas para pixeles similares.

Dado que incluso imágenes pequeñas pueden contener decenas de miles de pixeles, lo que se traduce en miles de millones de posibles interacciones binarias, las aplicaciones iniciales de CRF limitaban las interacciones a pixeles dentro de una vecindad local, o bien modelaban el etiquetado a nivel de super-pixeles (grupos de pixeles similares agrupadas previamente usando métodos de bajo nivel). Sin embargo, el primer enfoque tiende a generar máscaras de segmentación excesivamente suavizadas, y no permite aprovechar información de contexto global, mientras que el segundo está limitado por la calidad del método de generación de super-pixeles.

Para enfrentar estas limitaciones, en [46] se propuso una metodología que permite resolver CRFs que modelan las interacciones entre todos los pares de pixeles (*dense Conditional Random Fields* o dCRF) de manera eficiente. Esto permite modelar relaciones de más largo alcance, y con mayor nivel de detalle, obteniéndose máscaras de segmentación más precisas que se ajustan a bordes de bajo nivel de la imagen, tal como se muestra en la Fig. 3.7.

Más concretamente, en este caso denso la función de energía del CRF a minimizar se reduce a:

$$E(\boldsymbol{x}) = \sum_i \theta_i(x_i) + \sum_{ij} \theta_{ij}(x_i, x_j), \tag{3.10}$$



(a) Image (b) Unary classifiers (c) Robust P^n CRF (d) Fully connected CRF

Figura 3.7: Visualización del efecto de dCRF para segmentación. (a) Imagen de entrada. (b) Potenciales unarios. (c) Resultado de aplicar un método de CRF basado en super-pixeles. (d) Resultado de aplicar dCRF. Versión original extraída de [46].

donde $\boldsymbol{x} \in (\mathcal{C}_{\text{seg}})^{|V|}$ representa una asignación de etiquetas para los pixeles de la imagen. El potencial unario $\theta_i(x_i)$ usualmente se obtiene como $\theta_i(x_i) = -\log P(x_i)$, siendo $P(x_i)$ las probabilidades de clasificación generadas para cada ubicación de forma independiente por algún modelo de segmentación. Por otra parte, en [46] se propone limitar los potenciales binarios $\theta_{ij}(x_i, x_j)$ a una combinación lineal de filtros Gaussianos, lo que permite optimizar el modelo de forma eficiente utilizando un método iterativo aproximado basado en convoluciones. En concreto, los autores utilizan el siguiente potencial binario:

$$\theta_{ij}(x_i, x_j) = \mu(x_i, x_j)\left[w_1 \underbrace{\exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\alpha^2} - \frac{\|I_i - I_j\|^2}{2\sigma_\beta^2}\right)}_{\text{appearance kernel}} + w_2 \underbrace{\exp\left(-\frac{\|p_i - p_j\|^2}{2\sigma_\gamma^2}\right)}_{\text{smoothness kernel}}\right] \quad (3.11)$$

donde μ es una función de compatibilidad, típicamente tal que $\mu(x_i, x_j) = 1$ si $x_i \neq x_j$, y 0 en caso contrario, de modo que solo se penalizan pares de pixeles con etiquetas distintas. El primer filtro modela la similitud entre pixeles en términos de sus posiciones p_i y sus vectores de color RGB I_i, mientras que el segundo considera solo las posiciones. Los parámetros σ_α, σ_β y σ_γ controlan la escala de los filtros Gaussianos, y los parámetros w_1 y w_2 controlan sus pesos relativos. Esto favorece que pixeles cercanos con colores similares sean etiquetados con la misma categoría.

3.1.6 Grafos de Escena

Un grafo de escena o *scene graph* [47] es una estructura de datos que permite describir el contenido de una escena visual de manera eficiente. La información de la escena se codifica en términos de los **objetos** que la componen, sus respectivos **atributos**, y **relaciones** entre

objetos. La Fig. 3.8 muestra un ejemplo de grafo de escena, junto a la imagen que describe. Como se muestra en este ejemplo, los objetos pueden corresponder a gente, lugares, cosas, o partes de otros objetos; los atributos pueden describir características tales como colores, formas, y estados; mientras que las relaciones pueden ser geométricas, posesivas, y acciones. Esto les otorga a los grafos de escena una enorme flexibilidad para describir diversos tipos de escenas en distintos niveles de detalle.

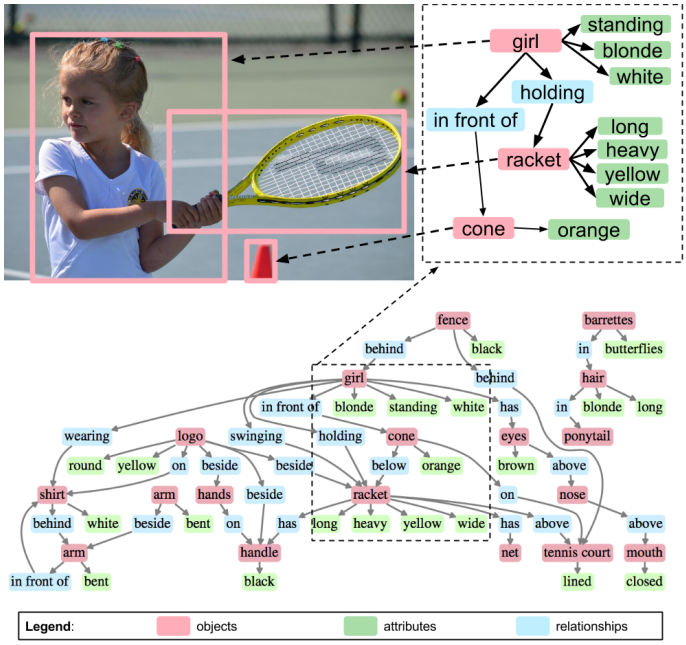


Figura 3.8: Ejemplo de un grafo de escena, junto a la imagen que describe. Extraída de [47].

Este tipo de representaciones han sido utilizadas extensivamente en aplicaciones de visión computacional, incluyendo *image retrieval* [47, 48], *image captioning* [49, 50], e *image generation* [23]. También se han propuesto modelos automatizados para generar grafos de escena, tanto a partir de imágenes [51, 52], como a partir de descripciones en lenguaje natural [24].

Generador de grafos de escena a partir de descripciones

En esta sección, se describe el modelo basado en reglas propuesto en [24] para generar grafos de escena a partir de descripciones de imágenes en lenguaje natural. Este modelo utiliza en primer lugar el Staford Parser v3.5.2 [53] para generar un árbol de dependencia a partir de la descripción de entrada, que sirve de punto de partida para generar el grafo de escena. Un árbol de dependencia [54] entrega una representación jerarquizada de las palabras de una frase, unidas entre sí por vértices dirigidos que indican el tipo de dependencia gramatical entre los nodos, como se muestra en la Fig. 3.9. Estas relaciones pueden indicar, por ejemplo, el objeto y sujeto de cada verbo, los miembros de una misma conjunción, compuestos, y modificadores de distintos tipos, entre otros.

Para facilitar la generación del grafo de escena, los autores aplican una serie de transformaciones sobre el árbol de dependencia para obtener una representación intermedia que denominan como *grafo semántico*. Concretamente, se aplica un algoritmo para resolver las

referencias de los pronombres, lo que es necesario para asignar relaciones correctamente, y se identifican las frases sustantivas que describen cuantificadores utilizando una lista compilada manualmente, tales como "*a dozen of*" o "*a lot of*". Las dependencias de estas frases se modifican para tratarlas como atributos de la frase sustantiva siguiente.

Adicionalmente, se utilizan todos los modificadores numéricos presentes en el árbol de dependencia para replicar los nodos correspondientes según la cantidad indicada, y se reemplazan los plurales por su raíz. Esto tiene por objetivo facilitar la asignación de cada nodo del grafo de escena a una región específica de la imagen. La Fig. 3.9 muestra un ejemplo de árbol de dependencia y su respectivo grafo semántico.

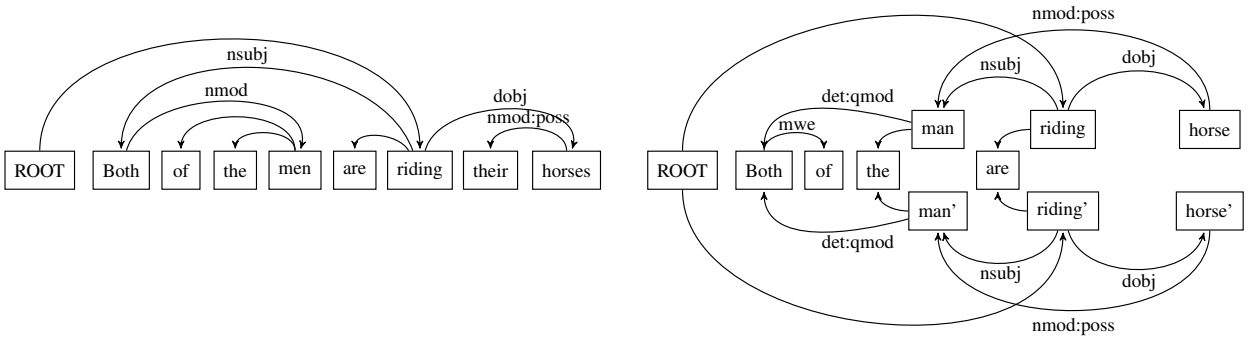


Figura 3.9: Ejemplo de árbol de dependencia (izquierda) y su respectivo grafo semántico (derecha). Extraída de [24].

Finalmente, se extraen objetos, atributos, y relaciones directamente desde el grafo semántico, utilizando 9 reglas genéricas que identifican diferentes tipos de patrones sintácticos, manejando fenómenos tales como adjetivos modificadores, construcciones tipo sujeto-predicado-objeto o sujeto-predicado, frases preposicionales, y construcciones posesivas, entre otras. Por ejemplo, para la frase "*a man riding a white horse*", la parte de su sub-grafo semántico dada por $man \xleftarrow{\text{nsubj}} riding \xrightarrow{\text{dobj}} horse \xrightarrow{\text{amod}} white$ coincidiría con el patrón para construcciones tipo sujeto-predicado-objeto, permitiendo identificar una relación del tipo "*riding*" entre los objetos "*man*" y "*horse*". Adicionalmente, el patrón de adjetivos modificadores, permitiría asignar un atributo "*white*" al mismo objeto "*horse*" en el grafo de escena final.

3.1.7 WordNet

WordNet [25] es una base de datos léxica, desarrollada originalmente por investigadores de la Universidad de Princeton en 1985 para el idioma inglés. Esta base de datos se estructura en forma de grafo, en el que cada nodo corresponde a un *synset*, definido como un conjunto de palabras con el mismo significado, y las aristas indican distintos tipos de relaciones semánticas. Este grafo se compone de más de 117000 nodos, e incluye sustantivos, verbos, adjetivos, y adverbios. Cada *synset* se denota con un nombre compuesto de 3 partes separadas por puntos, de la forma "<palabra>.<categoría gramatical>.<índice de significado>". Por ejemplo, el *synset* "*mouse*.n.04" denota el cuarto significado como sustantivo (*noun*) de la palabra "*mouse*".

Los tipos principales de relaciones definidas por el grafo corresponden a hiponimia y

meronimia, definidas de la siguiente manera:

1. Hiponima: X es un hipónimo de Y (equivalentemente, Y es un hiperónimo de X) si X es un tipo de Y. Por ejemplo, "perro" es un hipónimo de "mamífero".
2. Meronimia: X es un merónimo de Y (equivalentemente, Y es un holónimo de X) si X es parte o miembro de Y. Por ejemplo, "rama" es un merónimo de "árbol", y "árbol" es un merónimo de "bosque".

Esta base de datos ha sido extensivamente utilizada en múltiples aplicaciones de procesamiento de lenguaje. Más recientemente, WordNet ha sido utilizada en aplicaciones de visión computacional, tales como detección de objetos [55], búsqueda de imágenes [56], y segmentación con conjunto abierto [57, 58].

WordNet es también la herramienta *de facto* utilizada para definir la tarea de *Word Sense Disambiguation* (WSD), que consiste en identificar el significado específico de una palabra en un contexto particular [59, 60]. A continuación se describe el modelo del estado del arte para WSD propuesto en [59], que se utiliza en la implementación del módulo de procesamiento de texto desarrollado en el presente trabajo de tesis.

Modelo de *Word Sense Disambiguation*

El modelo de WSD propuesto en [59] utiliza *embeddings* contextualizados pre-entrenados para representar las palabras a desambiguar, que luego son procesados por una serie de capas tipo Transformer para adaptarlos a la tarea de WSD. Finalmente, se aplica un clasificador *fully-connected* sobre las representaciones procesadas de cada palabra para predecir sus significados en contexto, lo que se modela como una clasificación sobre *synsets* de WordNet. Este modelo es entrenado de manera supervisada, utilizando bases de datos de texto en las que cada palabra se encuentra anotada manualmente con su significado específico.

Este tipo de modelos suelen estar limitados principalmente por la escasez de datos disponibles, dado que las bases de datos existentes cubren solo una pequeña parte de las palabras con significados distintos definidas en WordNet. Si el modelo no observa un determinado sentido durante el entrenamiento, será en principio incapaz de reconocerlo con posterioridad. Para reducir la complejidad del problema, mejorar la capacidad de generalización del modelo, y aprovechar mejor los datos disponibles, los autores proponen una metodología para comprimir el vocabulario de WordNet basándose en relaciones de sinonimia e hiponimia, lo que se ilustra en la Fig. 3.10. Por ejemplo, si el verbo "*help*" aparece anotado con su primer significado en los datos de entrenamiento, esto permite reutilizar su contexto para aprender a etiquetar sus sinónimos "*assist*" o "*aid*" con el mismo *synset*.

De forma similar, se observa que para diferenciar entre dos significados de una misma palabra en general basta con poder diferenciar entre sus respectivos ancestros más altos en la jerarquía de hipernimia que no son compartidos por ambos significados. Por ejemplo, para diferenciar entre el primer significado de la palabra "*mouse*" (el roedor) y el cuarto (el dispositivo electrónico), bastaría con que el modelo fuera capaz de diferenciar entre sus respectivos hiperónimos "*living_thing*.n.01" y "*artifact*.n.01", pudiendo prescindirse de todos los sentidos más específicos de la jerarquía. Basados en esta observación, los autores proponen un algoritmo sencillo para comprimir drásticamente el vocabulario de salida al menor subconjunto

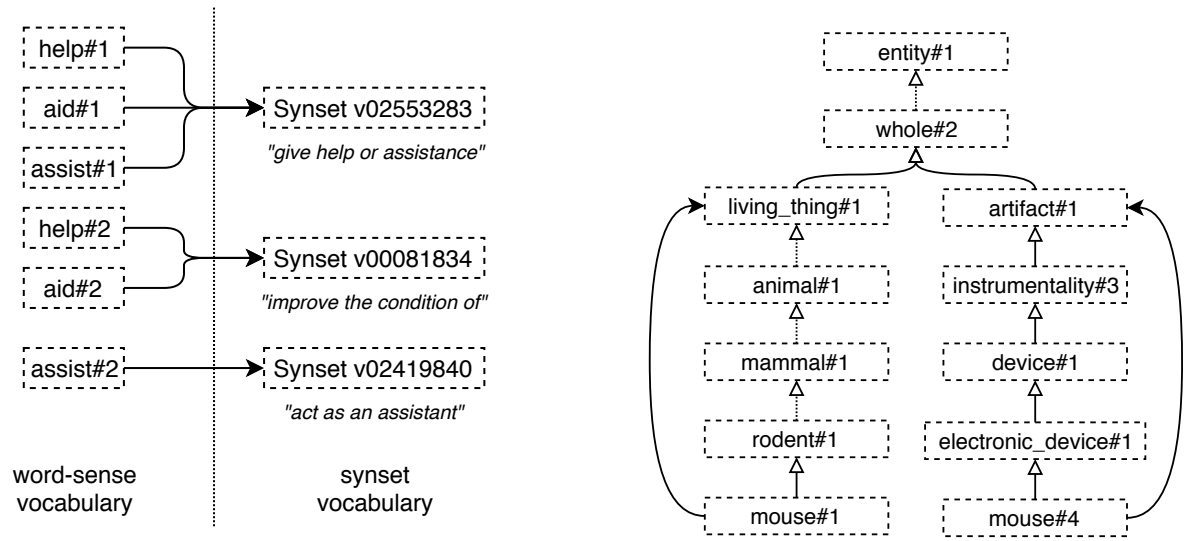


Figura 3.10: Compresión de vocabulario basado en relaciones de sinonimia (izquierda) e hiponimia (derecha). Extraída de [59].

de *synsets* que permite desambiguar todas las palabras definidas en WordNet.

3.2 Segmentación Semántica

Numerosas estrategias han sido exploradas en la literatura para abordar el problema de segmentación semántica. Sin embargo, durante los últimos años han sido métodos basados en redes neuronales profundas los que han dominado esta área, posibilitando dramáticos avances en diversos dominios importantes. A continuación se realiza una breve revisión de las principales técnicas y herramientas utilizadas en estos trabajos recientes, seguida de una revisión más detallada de la familia de modelos DeepLab. Esta serie de modelos es ampliamente utilizada en diversas aplicaciones de segmentación semántica en general, y constituye el estándar *de facto* para aplicaciones de segmentación semántica con supervisión débil en particular, por lo que reviste un interés especial para el presente trabajo.

3.2.1 *Fully Convolutional Networks*

Una de las componentes fundamentales de casi todos los modelos modernos de segmentación semántica consiste en el uso de *Fully Convolutional Networks* (FCNs), o redes completamente convolucionales. Uno de los primeros trabajos en explorar el uso de este tipo de arquitecturas para segmentación semántica fue [61]. En este trabajo, los autores modifican arquitecturas propuestas originalmente para clasificación, tales como VGG-16 y GoogLeNet, reemplazando todas las capas *fully-connected* por capas convolucionales. Esto permite obtener mapas de clasificación localizados a la salida del modelo en lugar de predicciones globales, tal como se muestra en la Fig. 3.11.

Las funciones de pérdida empleadas para entrenar este tipo de modelos suelen tratar cada pixel como un problema de clasificación independiente, y luego agregar los valores para toda

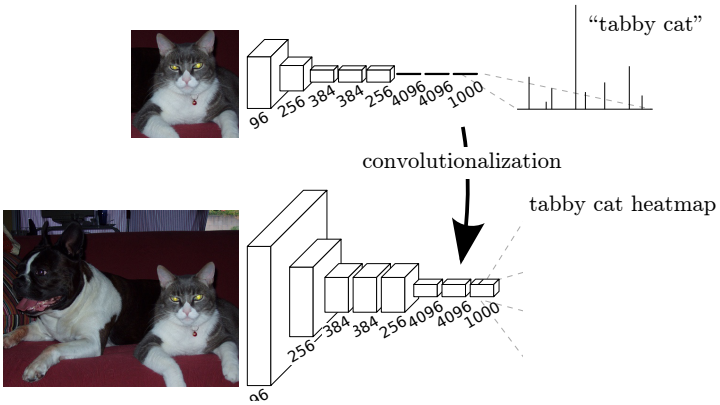


Figura 3.11: Esquema básico de una *fully convolutional network* (FCN) aplicada a segmentación semántica. Extraída de [61].

una imagen. Una vez entrenada la red, las máscaras de segmentación finales pueden ser obtenidas mediante simple *upsampling* de las predicciones localizadas de clasificación a la resolución de entrada, seguido de una operación de *argmax* por pixel.

Pese a sus resultados prometedores, las FCN por sí mismas tienen también importantes limitaciones. En [62], se identifican 3 desafíos principales para aplicar redes convolucionales profundas al problema de segmentación semántica: (1) estos modelos reducen drásticamente la resolución de la entrada antes de generar las predicciones finales, (2) no poseen un mecanismo intrínseco para manejar objetos a distintas escalas, y (3) sus propiedades de invariancia frente a pequeñas traslaciones, útiles para extraer información abstracta de alto nivel, en este caso limitan la precisión de las máscaras generadas, sobre todo en los bordes entre categorías. Las próximas secciones resumen algunas de las estrategias principales que han sido desarrolladas para enfrentar estas limitaciones.

3.2.2 Pirámides de Características

Una estrategia clásica para manejar objetos en múltiples escalas en algoritmos de procesamiento de imágenes consiste en generar una pirámide de réplicas de la imagen de entrada en distintas resoluciones, y luego aplicar el mismo modelo sobre cada versión de la imagen, combinando adecuadamente las salidas respectivas. Pese a que esta misma estrategia es aplicable a modelos de *deep learning*, la naturaleza piramidal y multi-escala de las representaciones internas que generan las redes convolucionales modernas permite implementar este tipo de esquema con mínimos costos adicionales.

Uno de los primeros modelos basados en este concepto fue *Feature Pyramid Network* (FPN) [63], propuesto principalmente para detección, pero adaptado posteriormente al problema de segmentación semántica. Este modelo complementa una FCN agregando un camino de "arriba a abajo" con conexiones laterales, que combina iterativamente la salida de cada bloque convolucional con la salida del bloque anterior mediante adición pixel a pixel, tal como se muestra en la Fig. 3.12. El modelo utiliza *upsampling* y convoluciones de 1×1 para igualar las resoluciones espaciales y el número de canales de cada par de mapas de características, respectivamente. Esto permite generar una pirámide de mapas de características

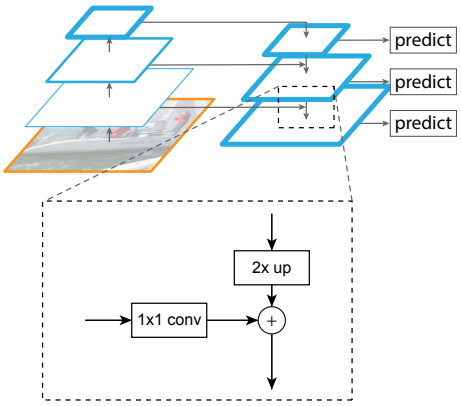


Figura 3.12: Arquitectura de *Feature Pyramid Network* (FPN). Extraída de [63].

en distintas resoluciones, donde todos los niveles tienen acceso a la información de más alto nivel adquirida por las capas finales de la red, posibilitando la segmentación de objetos en múltiples escalas.

3.2.3 Arquitecturas *Encoder-Decoder*

Las arquitecturas del tipo *encoder-decoder* constituyen una alternativa popular para obtener máscaras de segmentación de mayor resolución y nivel de detalle. En este tipo de arquitectura, los modelos convolucionales más clásicos se interpretan como un *encoder* responsable de extraer información de alto nivel sobre el contenido de la imagen, a los que se agrega un *decoder* también convolucional, encargado de aumentar progresivamente la resolución de las representaciones generadas. Un ejemplo de este tipo de arquitectura se muestra en la Fig. 3.13.

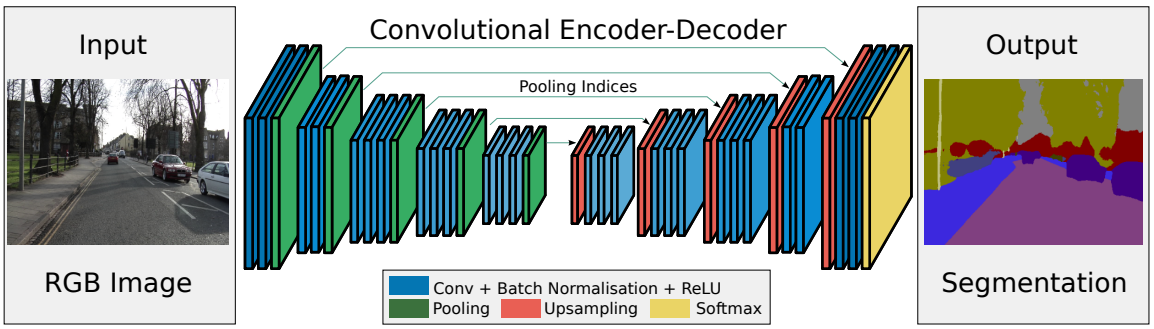


Figura 3.13: Diagrama general de SegNet, un modelo completamente convolucional del tipo *encoder-decoder* propuesto para el problema de segmentación semántica. Extraída de [64].

Algunos de los primeros modelos en aplicar este esquema al problema de segmentación semántica generaban una representación vectorial con información global de la imagen antes de aplicar el *decoder*, siguiendo aplicaciones más clásicas de arquitecturas *encoder-decoder*, como es el caso de [65]. Otras variantes más modernas en general utilizan redes completamente convolucionales, manteniendo siempre representaciones internas con información espacial, como es el caso de SegNet[64] o HRNet [66]. Este tipo de arquitecturas han sido ampliamente utilizadas en problemas de segmentación de imágenes médicas, con algunos de estos modelos

siendo posteriormente adaptados a otros dominios de aplicación, como es el caso de U-Net [67] y V-Net [68].

3.2.4 Modelos Gráficos

Una estrategia alternativa explorada por diversos trabajos recientes consiste en combinar modelos de *deep learning* con modelos gráficos tales como *Markov Random Fields* o *Conditional Random Fields* (ver Sección 3.1.5). Si bien este tipo de modelos ha sido utilizado históricamente principalmente para suavizar predicciones ruidosas generadas por modelos locales, en [69] se propuso utilizar *dense Conditional Random Fields* (dCRFs) [46] como una forma para refinar las predicciones generadas por redes convolucionales. En este caso, dCRF permite mejorar la precisión de las máscaras, ajustando las predicciones a los bordes de bajo nivel presentes en la imagen, como se ilustra en la Fig. 3.14. Esta misma estrategia ha sido posteriormente adapta y extendida en múltiples otros trabajos, tales como [70, 71, 62].

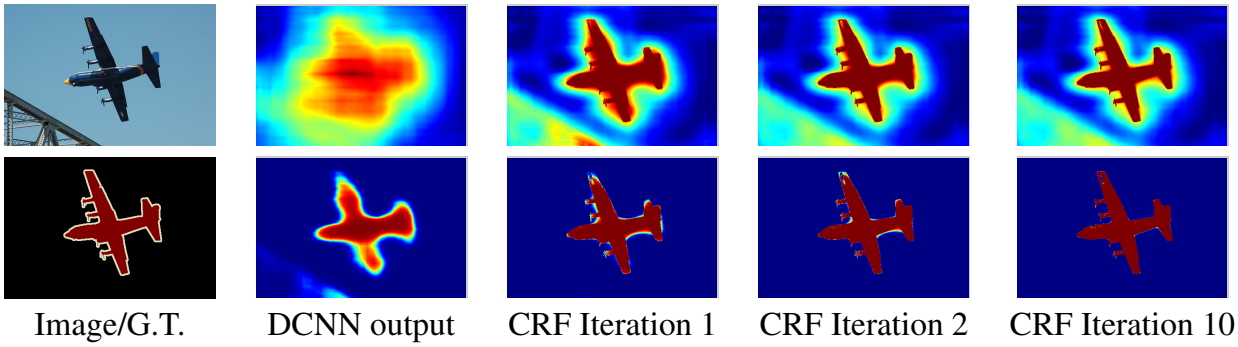


Figura 3.14: Visualización del efecto de aplicar CRF para refinar las predicciones generadas por un modelo convolucional. Extraída de [69].

3.2.5 La Familia de Modelos DeepLab

La familia de modelos DeepLab se caracteriza por el uso de convoluciones dilatadas como un mecanismo para regular el campo receptivo de distintas capas de la red, manteniendo constante el número de parámetros. Este tipo de convoluciones introduce un hiperparámetro adicional r, llamado tasa de dilatación o *dilation rate*, que define el número de espacios entre los pesos del *kernel* convolucional. En la Fig. 3.15 se muestra gráficamente el efecto de distintas tasas de dilatación en el caso bidimensional, para un *kernel* de 3×3. Cabe destacar que el caso con tasa de dilatación $r = 1$ corresponde a la convolución usual.

El primero de estos modelos, **DeepLabv1** o DeepLab-Large-FOV [69], utiliza una arquitectura completamente convolucional basada en VGG-16, similar a la de la Fig. 3.11. Además de reemplazar las capas *fully-connected* por capas convolucionales, se eliminan las dos últimas etapas de sub-muestreo, lo que se traduce en salidas de mayor resolución (1/8 de la resolución de la entrada, en lugar del usual 1/32). Para preservar el tamaño del campo receptivo de las neuronas finales sin aumentar el número de parámetros del modelo, los autores reemplazan las capas convolucionales de los dos últimos bloques por convoluciones dilatadas con tasa de dilatación 2 (para el penúltimo bloque) y 4 (para el último bloque). La mayor resolución

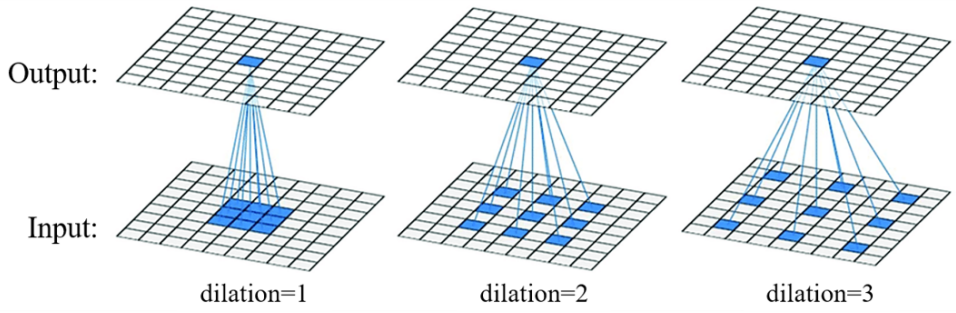


Figura 3.15: Visualización del efecto de distintas tasas de dilatación sobre un filtro convolucional de 3×3. Versión original extraída de [72].

de la salida, combinada con una etapa de post-procesamiento con dCRF, permite generar máscaras de segmentación de gran calidad con un simple *forward pass*.

Este trabajo fue expandido posteriormente en [62], en el que se propone **DeepLabv2** o DeepLab-ASPP, que incorpora una etapa adicional de *Atrous Spatial Pyramid Pooling* (ASPP) antes de la clasificación final. Esta etapa consiste en múltiples convoluciones dilatadas con distintas tasas de dilatación, aplicadas en paralelo sobre el mismo mapa de características, permitiéndole al modelo detectar patrones en diferentes escalas de manera eficiente. Las salidas de cada convolución se combinan seleccionando las máximas activaciones en cada posición para obtener las predicciones finales. La Fig. 3.16 ilustra la implementación de ASPP utilizada por este modelo.

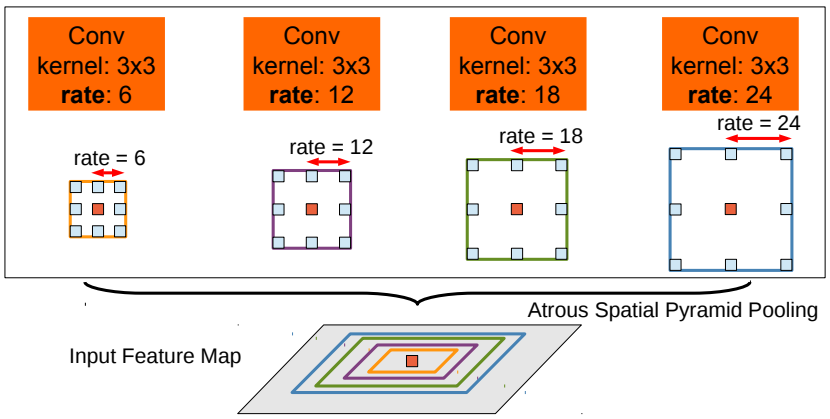


Figura 3.16: Bloque de *Atrous Spatial Pyramid Pooling* (ASPP). Extraída de [62].

Posteriormente, con **DeepLabv3** [73] se introducen diversas mejoras adicionales al modelo, incluyendo arquitecturas más profundas, y una representación global que complementa ASPP capturando información contextual de la imagen completa. Por otra parte, con **DeepLabv3+** [74] se combina la información rica en contexto de modelos basados en convoluciones dilatadas y ASPP, con las fronteras mejor definidas de arquitecturas tipo *encoder-decoder*. Para esto, el modelo utiliza DeepLabv3 como *encoder*, e introduce un *decoder* sencillo que combina una versión escalada de la salida con un mapa de características intermedio de la red mediante concatenación seguida de una convolución, similar al esquema de la Fig. 3.12.

3.3 Segmentación Semántica con Supervisión Débil

Diversos tipos de supervisión débil han sido explorados en la literatura para entrenar modelos de segmentación semántica, incluyendo *bounding boxes* [1, 2], garabatos (*scribbles*) [3], y puntos [4]. Otros trabajos han explorado complementar etiquetas de clasificación con *saliency maps* [75, 76, 77, 78, 79], que corresponden a mapas de activación que destacan objetos relevantes en la imagen, independiente de sus categorías. Sin embargo, esto requiere de modelos adicionales entrenados con supervisión fuerte a nivel de pixel, que no son independientes del dominio de imágenes de interés, lo que dificulta la extensión de este tipo de estrategias a nuevas aplicaciones. La presente tesis se enfoca en el caso supervisado únicamente con información a nivel de imagen, es decir, sin ningún tipo de información de localización.

Los trabajos que constituyen el estado del arte en WSSS con anotaciones a nivel de imagen siguen un esquema de auto-supervisión, en el que las etiquetas débiles disponibles son utilizadas para generar máscaras de segmentación artificiales de forma automática, que luego se emplean como pseudo-*ground truths* para entrenar un modelo supervisado. En la Sección 3.3.1 se resume la metodología propuesta en [12] para generar mapas de activación por clase a partir de clasificadores convolucionales, que es la técnica más común utilizada por estos métodos para inicializar las máscaras de segmentación artificiales. En la Sección 3.3.2 se detalla el esquema general utilizado por los modelos modernos de WSSS para generar y utilizar las máscaras de segmentación artificiales, y en la Sección 3.3.3 se revisan algunos de los principales trabajos en esta área que utilizan etiquetas de clasificación como supervisión débil. Finalmente, en la Sección 3.3.4 se resume el trabajo previo propuesto en la literatura para el problema de WSSS utilizando descripciones en lenguaje natural, que es el tema central del presente estudio.

3.3.1 *Class Activation Maps* (CAMs)

En [12], se propone una metodología para obtener mapas de localización por clase a partir de clasificadores convolucionales entrenados solo con supervisión a nivel de imagen. Para esto, los autores estudian modelos conformados casi por completo por capas convolucionales, que utilizan una operación de *pooling* global justo antes de la capa final de clasificación *fully-connected*. Este tipo de arquitecturas ya habían sido propuestas previamente como una forma de reducir el número de parámetros de las redes convolucionales, manteniendo la calidad de la clasificación [32], pero en este caso se demuestra que permiten localizar sus predicciones sin necesidad de supervisión adicional.

En estos casos, el *logit* asociado a una determinada clase c se computa como una sumatoria $\sum_{k=1}^{d_{\text{enc}}} \boldsymbol{w}_c(k)\boldsymbol{f}(k)$, donde $\boldsymbol{w}_c(k)$ es el k-ésimo elemento del vector de pesos de la capa de clasificación asociado a la clase c, y $\boldsymbol{f}(k)$ es el k-ésimo elemento del vector $\boldsymbol{f} \in \mathbb{R}^{d_{\text{enc}}}$ resultante de aplicar la operación de *pooling* espacial sobre el último mapa de características $\mathbf{F} \in \mathbb{R}^{d_{\text{enc}} \times h_{\text{enc}} \times w_{\text{enc}}}$. De esta manera, $\boldsymbol{w}_c(k)$ esencialmente indica la importancia del canal k para la predicción de la categoría c. Por lo tanto, los autores proponen computar una suma ponderada sobre los canales de $\mathbf{F}$ utilizando los pesos de la capa de clasificación para generar mapas de activación por clase o *Class Activation Maps* (CAMs) $\boldsymbol{M}_c \in \mathbb{R}^{h_{\text{enc}} \times w_{\text{enc}}}$, de modo que la

activación en cada posición espacial está dada por:

$$\boldsymbol{M}_c(i,j) = \sum_{k=1} \boldsymbol{w}_c(k) \cdot \mathbf{F}(k,i,j), \tag{3.12}$$

donde se utilizan paréntesis para denotar indexación.

De esta forma, los valores de $\boldsymbol{M}_c$ indican la contribución de cada elemento espacial a la predicción de la categoría c. Mediante simple *upsampling* a la resolución original, estos CAMs permiten identificar las regiones de la imagen de entrada que más contribuyen a la predicción de cada categoría. La Fig. 3.17 muestra una visualización del proceso, así como del tipo de mapa de activación generado.

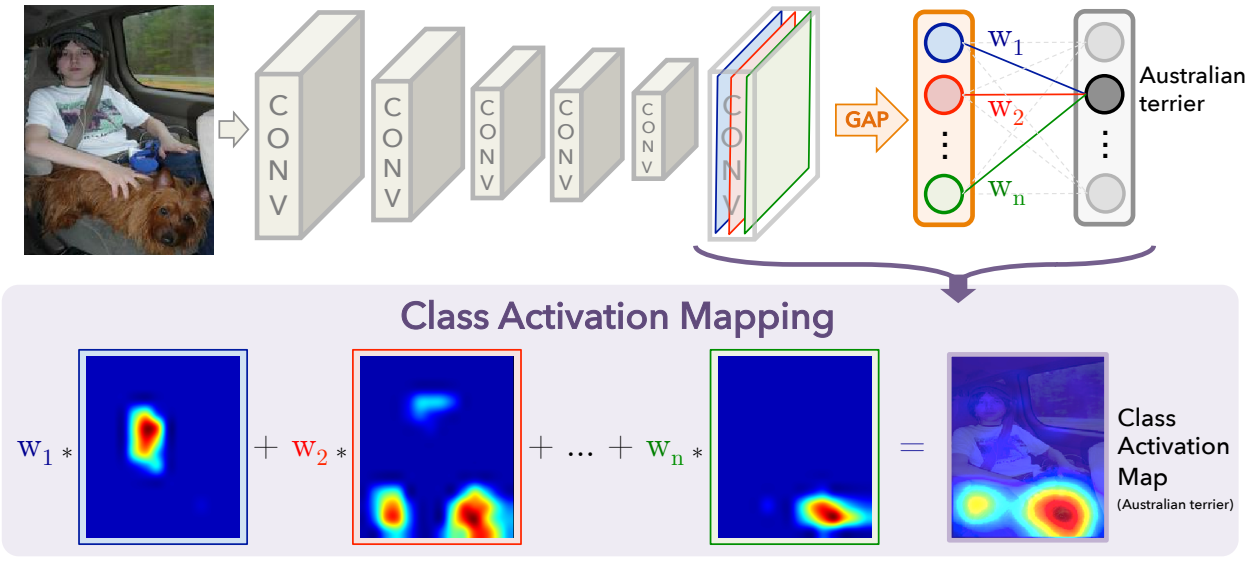


Figura 3.17: Proceso de generación de *Class Activation Maps* (CAMs). Los pesos de la capa de clasificación asociados a una determinada clase se proyectan sobre el último mapa de características convolucional, generando un CAM que destaca las regiones más relevantes para la clasificación. Extraída de [12].

3.3.2 Esquema General de WSSS con Auto-Supervisión

Los métodos que utilizan un esquema de auto-supervisión para WSSS comienzan entrenando un clasificador convolucional con las etiquetas de clasificación disponibles, que posteriormente es utilizado para generar CAMs sobre cada imagen de entrenamiento, siguiendo el proceso descrito anteriormente. Una estrategia común consiste en seleccionar los pixeles con mayores activaciones en los CAMs para obtener semillas de segmentación o *seeds*, que corresponden a regiones con alta confianza de pertenecer a cada categoría presente en la imagen según las etiquetas del *ground truth*. Para obtener semillas similares para la clase de fondo, algunos trabajos han utilizado *saliency maps* generados por modelos supervisados [6, 7, 8, 10]. Sin embargo, esto introduce requerimientos de anotaciones adicionales, por lo que otros trabajos han explorado metodologías para obtener estas semillas a partir de la misma supervisión a nivel de imagen [5, 9, 17, 80, 81]. Las semillas generadas son finalmente combinadas para

obtener máscaras de segmentación artificiales sobre el mismo conjunto de entrenamiento, que se utilizan para entrenar un modelo se segmentación de forma supervisada.

Dado que los CAMs tienden a destacar solo aquellas regiones que más aportan a la clasificación (por ejemplo, la cabeza de los animales, o las ruedas de los vehículos), las semillas generadas cubren usualmente solo una fracción de los objetos relevantes. Por esta razón, la mayor parte de los esfuerzos recientes en WSSS se han centrado en reducir la brecha entre los mapas de relevancia para clasificación, y las máscaras necesarias para segmentación. A continuación se presentan algunos de los principales trabajos y líneas de investigación en esta área.

3.3.3 Trabajos Relevantes

Uno de los primeros trabajos en emplear el esquema de auto-supervisión para WSSS fue [5], en el que se sentaron las bases para el desarrollo posterior en esta tarea. Este trabajo utiliza una combinación de 3 funciones de pérdida para entrenar un modelo supervisado a partir de semillas de segmentación incompletas generadas a partir de CAMs. En primer lugar, una *seed loss* guía a la red para predecir correctamente las categorías de pixeles etiquetados, ignorando el resto. Se define como:

$$\ell_{seed} = -\frac{1}{\sum_{c \in \mathcal{C}_{\text{seg}}} |\mathcal{S}_c|} \sum_{c \in \mathcal{C}_{\text{seg}}} \sum_{(i,j) \in \mathcal{S}_c} \log(\boldsymbol{H}_c(i,j)), \tag{3.13}$$

donde $\mathcal{C}_{\text{seg}}$ es el conjunto de todas las categorías de objetos, incluyendo la categoría fondo, $\mathcal{S}_c$ es el conjunto de ubicaciones espaciales (i, j) asignados a la categoría c en la máscara de segmentación parcial, y $\boldsymbol{H}_c(i, j)$ es la probabilidad predicha por el modelo para la categoría c en la ubicación (i, j).

Por otra parte, se computa una *expansion loss* aplicando una operación de *pooling* global sobre las predicciones de segmentación generadas por el modelo para obtener predicciones de clasificación globales, sobre las que luego se aplica la misma función de pérdida utilizada para entrenar los modelos de clasificación. Esto tiene por objetivo favorecer la predicción de regiones de mayor tamaño para categorías positivas, además de suprimir predicciones de segmentación para categorías que no están presentes en la imagen. Finalmente, se utiliza una *constrain-to-boundary loss*, que corresponde a la divergencia KL entre el mapa de segmentación predicho por el modelo, y el mismo mapa después de ser refinado con dCRF. Esta componente busca favorecer predicciones que se ajustan a los bordes de bajo nivel de la imagen. La Fig. 3.18 resume el procedimiento seguido, incluyendo la generación de semillas de segmentación y el entrenamiento del modelo supervisado.

Por otra parte, en [7] se extiende la metodología de [5], proponiendo una estrategia que permite expandir las máscaras de segmentación artificiales de forma iterativa. Una vez entrenado el modelo de segmentación utilizando las semillas iniciales, las representaciones aprendidas por el mismo modelo se aprovechan para extender las semillas de segmentación hacia pixeles similares no etiquetados, mediante un procedimiento que se denomina *Deep Seeded Region Growing* (DSRG). Concretamente, para cada pixel etiquetado en la máscara original, se asigna su misma etiqueta a todos sus vecinos no etiquetados para los cuales el modelo predice una probabilidad de pertenecer a esa misma categoría que supera un cierto umbral. Las semillas

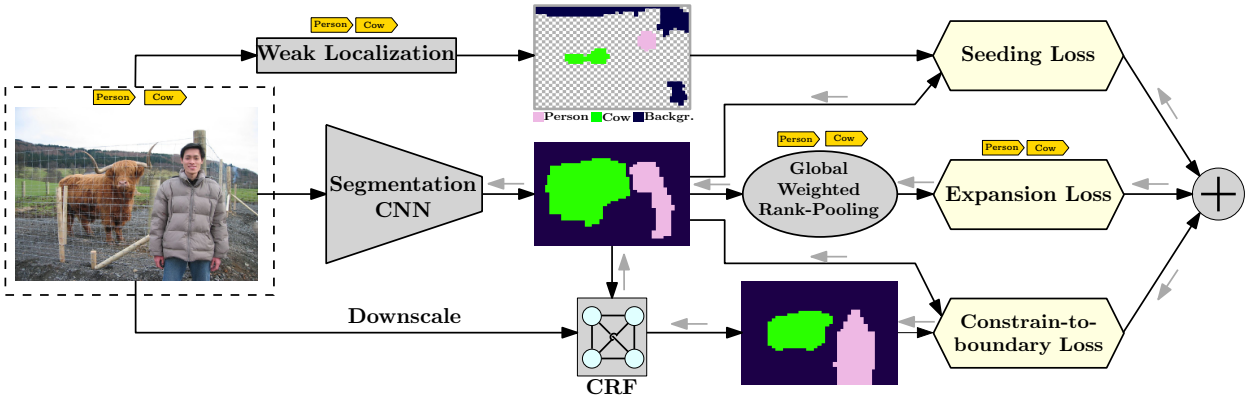


Figura 3.18: Esquema general de *Seed, Expand and Constrain*. Extraída de [5].

extendidas resultantes son utilizadas para re-entrenar el modelo, repitiendo el procedimiento de forma iterativa para expandir progresivamente las regiones relevantes cubiertas por las máscaras de segmentación artificiales, con efectos como los que se muestran en la Fig. 3.19.

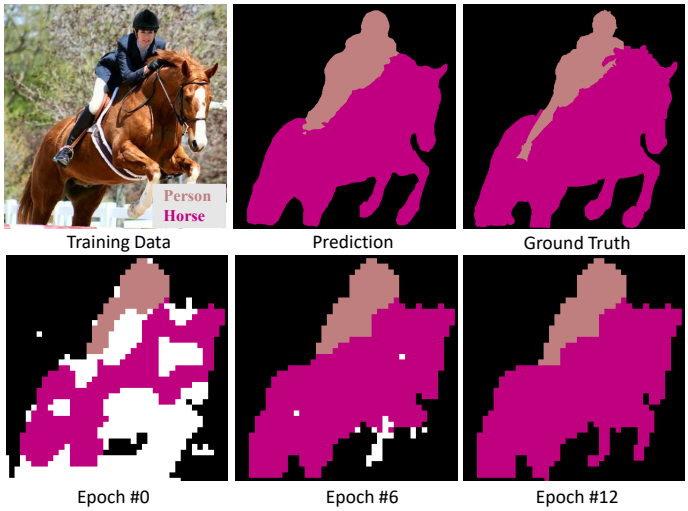


Figura 3.19: Ejemplo de expansión progresiva de las máscaras de segmentación obtenidas usando *Deep Seeded Region Growing*. Extraída de [7].

En [6] se propone una estrategia de *Adversarial Erasing* para extender la cobertura de las máscaras generadas. En este trabajo, las regiones de más alta confianza de los CAMs son borradas de las imágenes de entrenamiento, reemplazando los pixeles correspondientes por el valor de color medio en la base de datos. Las imágenes resultantes se utilizan para re-entrenar el clasificador, lo que obliga al modelo a identificar otras regiones de los mismos objetos para predecir sus categorías, resultando en una segunda iteración de CAMs que cubre regiones menos características de los objetos originales. El procedimiento se repite iterativamente, hasta que el modelo de clasificación es incapaz de converger adecuadamente. Una visión esquemática de este procedimiento se muestra en la Fig. 3.20. Finalmente, la unión de todas las regiones eliminadas en cada iteración se utilizan para generar máscaras de segmentación más completas, con las que se entrena el modelo supervisado.

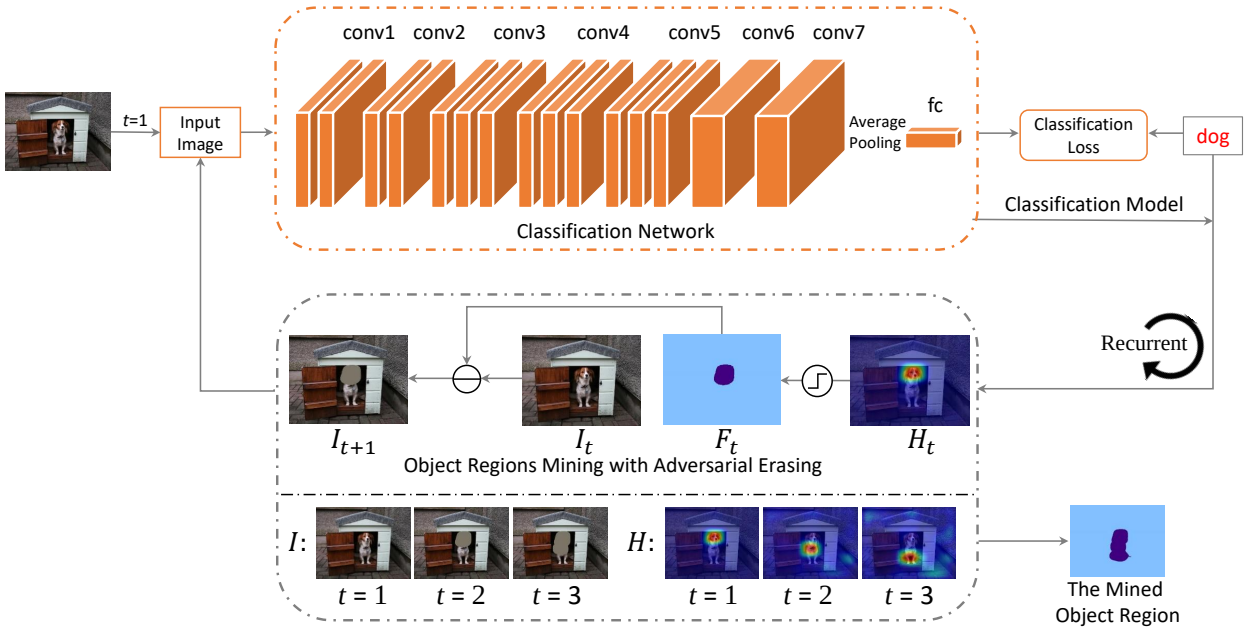


Figura 3.20: Esquema general del procedimiento de *Adversarial Erasing* para WSSS. Extraída de [6].

Esta idea de borrar las regiones más características de la imagen para obligar al modelo a extender los mapas aprendidos hacia otros pixeles relevantes ha sido expandida en otros trabajos posteriores [82, 83]. Por ejemplo, en [82] se propone una metodología para aplicar el proceso de borrado durante el entrenamiento del modelo de clasificación de forma *online*. Para esto, los CAMs generados por el clasificador para categorías positivas se utilizan para borrar las regiones más características de la imagen de entrada, y el resultado se utiliza como entrada para la misma red. El modelo se entrena simultáneamente para maximizar las predicciones de categorías positivas para la imagen original, y para minimizar las predicciones de cada categoría después del borrado de su CAM correspondiente. Esto promueve el aprendizaje de mapas de atención que cubren todas las regiones de la imagen que pueden utilizarse para predecir una determinada clase.

Otra línea de investigación explora el uso de convoluciones dilatadas en el modelo de clasificación como un método para expandir los CAMs hacia regiones menos características de los objetos. En [8], se agrega una pirámide de convoluciones con distintas tasas de dilatación justo antes de la capa de clasificación del modelo con supervisión débil, inspirados por la capa de *Atrous Spatial Pyramid Pooling* (ASPP) propuesta originalmente en [62] para segmentación semántica con supervisión completa. Los autores observan que el campo receptivo extendido de las convoluciones con mayores tasas de dilatación le permiten al modelo transferir información útil desde regiones más características hacia otras menos representativas, resultando en CAMs más completos. Este efecto se ilustra en la Fig. 3.21. Los mapas de activación finales se obtienen en este caso promediando los CAMs generados por cada rama de la pirámide de convoluciones dilatadas, y sumando el resultado al CAM generado con la convolución usual, sin dilatación. Esta estrategia ayuda a suprimir regiones que se destacan incorrectamente al usar tasas de dilatación más grandes.

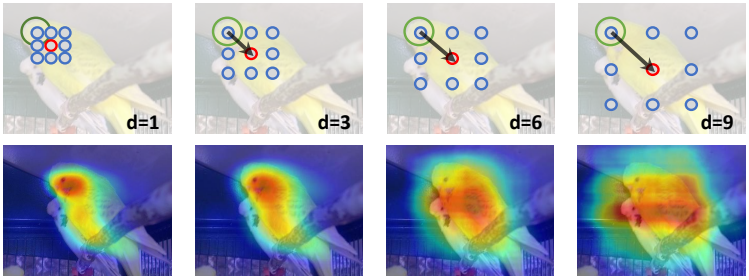


Figura 3.21: Visualización del efecto de distintas tasas de dilatación sobre los CAMs generados por el modelo de clasificación. Extraída de [8].

En [10], se propone una estrategia de *dropout* espacial para la última capa convolucional del clasificador como una forma de propagar información de regiones características a vecindades cercanas, con patrones de distintas formas. Para cada posición del *kernel* convolucional, se selecciona un conjunto aleatorio de elementos que son omitidos del cómputo de la salida (equivalente a reemplazar sus pesos correspondientes por 0), pero manteniendo siempre el pixel central activo. Para generar los mapas de activación por clase, se promedian los resultados de 200 iteraciones independientes de la última capa, correspondientes a distintas configuraciones aleatorias del *kernel* convolucional. Los autores argumentan que esta estrategia generaliza la combinación de convoluciones con distintas tasas de dilatación propuesta en [8], ya que se espera que algunas de las configuraciones aleatorias del *kernel* correspondan a patrones similares a convoluciones dilatadas, mientras que otras permiten la propagación de información de contexto en formas más flexibles, como se muestra en la Fig. 3.22.

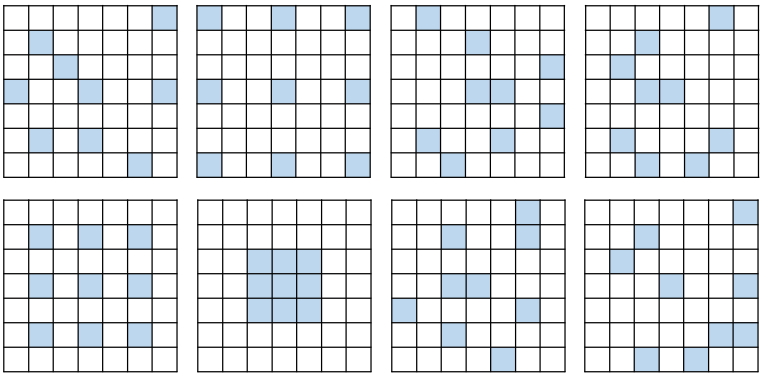


Figura 3.22: Ejemplos de posibles patrones de activación al utilizar el esquema de *dropout* espacial de FickleNet sobre un *kernel* convolucional de 7×7. Extraída de [8]

En [9], se propone entrenar un modelo adicional llamado AffinityNet, que permite generar matrices que codifican la afinidad entre cada par de pixeles de una imagen. Esta información se convierte a matrices de probabilidad de transición, que se utilizan para propagar las semillas de segmentación generadas a partir de CAMs hacia otras regiones de la imagen, mediante un procedimiento iterativo basado en *random walk* [84]. Estas máscaras de segmentación refinadas son las que finalmente se utilizan para entrenar un modelo supervisado. El entrenamiento de AffinityNet requiere de etiquetas de afinidad entre pares de pixeles, las que se obtienen directamente a partir de las mismas semillas extraídas desde los CAMs: para

cada pixel etiquetado, se seleccionan todos los pixeles dentro de un cierto radio para formar pares positivos (si comparten la misma etiqueta) o negativos (si tienen etiquetas distintas). Los pixeles no etiquetados se ignoran. La Fig. 3.23 resume el proceso completo.

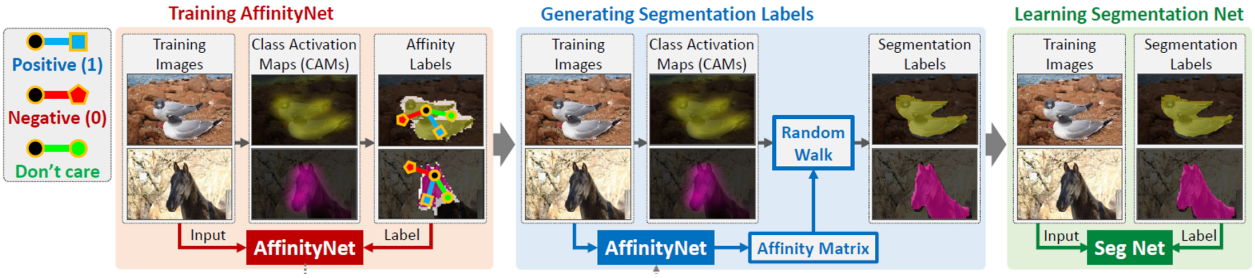


Figura 3.23: Esquema general del entrenamiento en 3 etapas para AffinityNet. Original extraída de [9].

Esta idea de modelar explícitamente las afinidades entre pares de pixeles para refinar los mapas de activación unarios dados por CAMs ha sido expandida en otros trabajos posteriores [81, 11]. En [81] se propone IRNet, que extiende AffinityNet permitiendo generar mapas de fronteras entre clases, los que luego pueden convertirse a mapas de afinidad que se utilizan como en [9], además de generar mapas que codifican el desplazamiento de cada pixel relevante respecto del centro de su respectiva instancia. Esto último permite aplicar el modelo al problema más complejo de segmentación de instancia, y puede ser entrenado utilizando los mismos CAMs obtenidos a partir de etiquetas de clasificación. Por otra parte, en [11] se propone un modelo con dos ramas que simultáneamente aprende a generar máscaras de segmentación unarias, y matrices de afinidad entre pares de pixeles. Esto permite refinar iterativamente las predicciones del modelo de segmentación, a partir de regiones de alta confianza.

Similar a los trabajos descritos anteriormente, la metodología presentada en la presente tesis utiliza un esquema de auto-supervisión, en el que máscaras de segmentación generadas por un modelo entrenado con supervisión débil se utilizan para entrenar un modelo de segmentación semántica en régimen supervisado. Sin embargo, contrario a los trabajos anteriores, el desarrollo se enfoca en la generación de mapas de localización precisos a partir de descripciones en lenguaje natural. Dado que la mayor parte de los trabajos existentes para WSSS con supervisión a nivel de imagen se centran en refinar los mapas de localización generados por el modelo con supervisión débil, o en regularizar el entrenamiento del modelo supervisado, estos pueden ser fácilmente adaptados para complementar la metodología propuesta en la presente tesis.

3.3.4 WSSS Utilizando Descripciones

El único trabajo existente que aborda el problema de WSSS utilizando solo descripciones en lenguaje natural es [17], en el que se presenta una metodología para generar mapas de activación por clase similares a CAMs a partir de descripciones, a los que se denomina *Text Activation Maps* (TAMs). El modelo propuesto para generar estos TAMs, TAM-Net, utiliza dos ramas independientes para mapear imágenes y segmentos de texto arbitrario a un mismo espacio vectorial multi-modal, inspirado por trabajos previos en el problema de *visual groun-*

ding (ver Sección 3.4). Una vez entrenado, este modelo permite generar mapas de activación para segmentos de texto utilizando el producto punto entre las representaciones de ambas modalidades en este espacio vectorial, como se muestra en la Fig. 3.24. Para asociar segmentos de texto de las descripciones de la base de datos con categorías semánticas relevantes, los autores simplemente detectan menciones exactas del nombre de las categorías, o su versión plural.

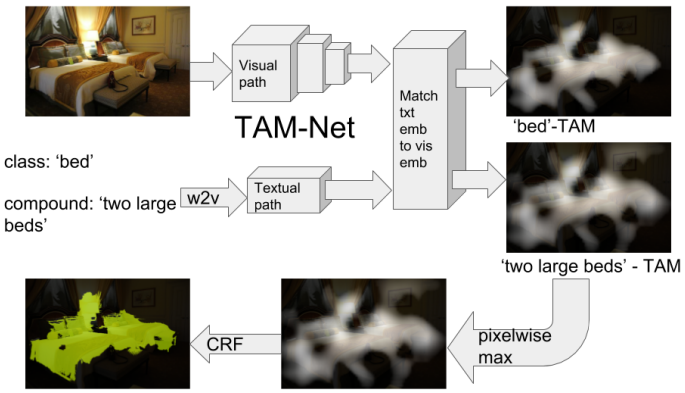


Figura 3.24: Esquema general de TAM-Net. Extraída de [17].

La rama visual de TAM-Net se implementa usando una red neuronal convolucional, para la que se consideran dos variantes, una basada en VGG-16 [31], y la otra en ResNet-38 [85]. Por otra parte, la rama textual utiliza representaciones Word2vec [41] pre-entrenadas para codificar cada palabra de la frase de entrada, las que luego son promediadas y proyectadas al mismo espacio de las representaciones extraídas por la rama visual. El modelo se entrena utilizando entropía cruzada binaria, con pesos separados para las frases que contienen los nombres de categorías relevantes, y aquellas que no. Las frases negativas para la supervisión se muestrean aleatoriamente a partir de las descripciones de la base de datos.

Sin embargo, esto introduce información no visual al entrenamiento, además de falsos negativos debidos al hecho de que segmentos cortos de texto con frecuencia describen adecuadamente regiones de imágenes con las que no están emparejadas. Los autores observan que el modelo tiende a aprender soluciones degeneradas, lo que evitan agregando un término residual en la rama textual, que limita la capacidad del modelo para modificar los *embeddings* pre-entrenados, como se muestra en la Fig. 3.25.

La red de localización utilizada en el presente trabajo sigue un diseño y entrenamiento similar a TAM-Net, pero simplificando la rama textual para codificar únicamente categorías de objetos, y pares categoría-atributo. La principal diferencia entre ambos métodos reside en el uso de la supervisión estructurada generada por el módulo de procesamiento de texto propuesto en esta tesis. Esta supervisión destilada permite aprovechar información visual más diversa y completa presente en descripciones en lenguaje natural al momento de generar las máscaras de segmentación, a la vez que suprime señales ruidosas durante el entrenamiento de la red de localización. Los experimentos realizados demuestran que ambos aspectos son cruciales para la calidad de la supervisión generada.

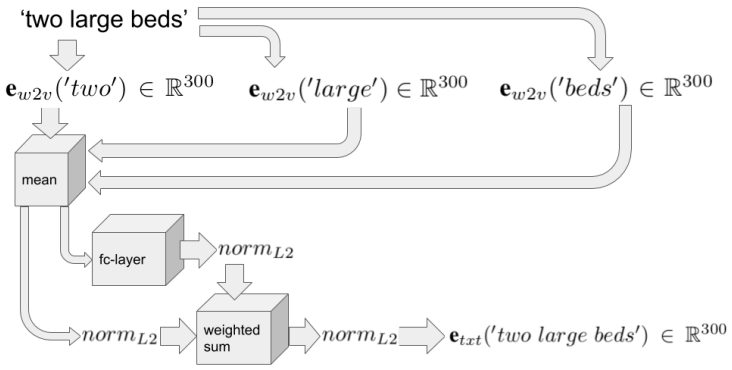


Figura 3.25: Rama visual de TAM-Net. Extraída de [17].

3.4 Visual Grounding

El problema de WSSS utilizando descripciones que se estudia en la presente tesis está relacionado también con la tarea de *visual grounding* (VG). Dado una imagen y un segmento de texto correspondiente, esta tarea tiene por objetivo identificar la región de la imagen descrita por el texto. Similar a lo que ocurre en el caso de segmentación semántica, los trabajos iniciales en VG dependían de bases de datos anotadas con supervisión completa, es decir, imágenes anotadas con regiones de interés, cada una asociada a una o más descripciones en lenguaje natural. Más recientemente, otros enfoques han sido desarrollados para poder entrenar modelos de VG utilizando solo información a nivel de imagen, que en este caso corresponden a descripciones globales, permitiendo reducir dramáticamente los costos de anotación.

La mayoría de estos métodos dependen de propuestas de regiones generadas por modelos externos, que usualmente requieren de supervisión más fuerte, y localizan el texto utilizando *bounding boxes* [86, 87, 20, 21], lo que los hace inadecuados para generar máscaras a nivel de pixel, como se requiere en WSSS. Sin embargo, recientemente se han propuesto también otros estudios que formalizan el problema de VG como la obtención de un mapa de calor para representar la relevancia de cada pixel de una imagen, dada una frase de entrada [18, 22, 88]. Todos estos trabajos utilizan *visual-semantic embeddings* (VSE) para mapear tanto imágenes como texto a un mismo espacio vectorial multi-modal, lo que permite modelar afinidades entre las dos modalidades en términos de similitudes en el espacio vectorial. Los modelos que generan estas codificaciones son entrenados utilizando funciones de pérdida tipo *contrastive loss*, que se basan en contrastar pares positivos y negativos de ejemplos multi-modales.

La red de localización propuesta en el presente trabajo de tesis utiliza un esquema similar a estos trabajos para generar mapas de activación a partir de descripciones usando un espacio vectorial visual-semántico. Sin embargo, contrario a los trabajos de VG, el objetivo del modelo propuesto no es localizar texto novedoso en imágenes arbitrarias, sino que generar máscaras de segmentación precisas para el mismo conjunto de entrenamiento. Por esta razón, el espacio semántico en el caso del modelo propuesto se limita a codificar etiquetas visuales (tales como categorías y atributos) generadas por el módulo de procesamiento de texto propuesto. Este enfoque simplifica el diseño y entrenamiento de la red de localización, y focaliza su supervisión en información visual relevante.

Modelo de *Visual Grounding*

El modelo propuesto en [22] constituye el estado del arte en *visual grounding* con supervisión débil basado en mapas de activación. Este modelo es representativo del esquema general de VSE, y es utilizado como referencia para la evaluación del modelo propuesto durante los experimentos desarrollados, por lo que se describe en mayor detalle a continuación.

Este modelo consta de dos ramas, que mapean por separado imágenes y descripciones a un mismo espacio vectorial multi-modal, siguiendo el esquema de la Fig. 3.26. En primer lugar, la rama visual utiliza una red convolucional basada en ResNet-152 para extraer un mapa de características $\mathbf{G} \in \mathbb{R}^{D' \times h \times w}$ con D' canales y dimensiones espaciales (h, w) a partir de una imagen de entrada. Posteriormente, se aplica una operación de *pooling* global para transformar $\mathbf{G}$ en un vector $\boldsymbol{h} \in \mathbb{R}^{D'}$, seguida de una capa *fully-connected* y normalización L^2 para obtener el *embedding* visual final $\boldsymbol{x} \in \mathbb{R}^d$.

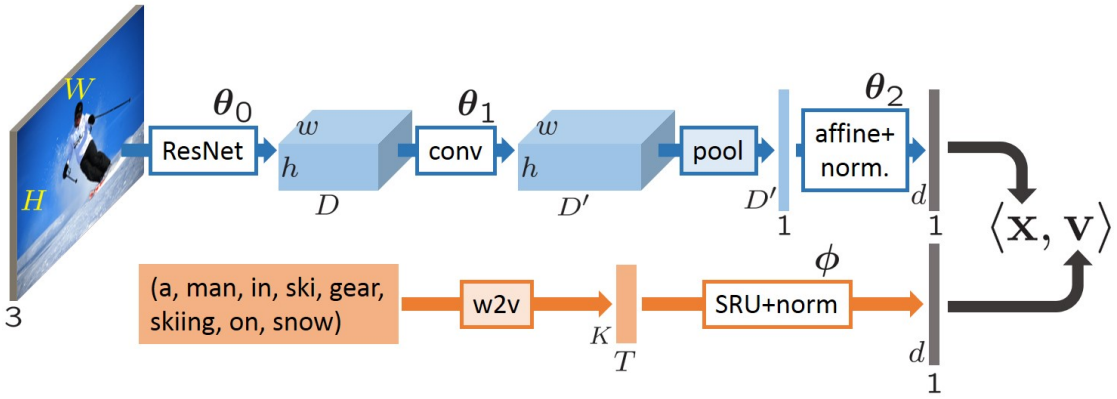


Figura 3.26: Esquema del modelo de *visual grounding* propuesto en [22]. El modelo proyecta imágenes y descripciones en lenguaje natural a un espacio vectorial compartido que preserva relaciones semánticas. Extraída de [22].

Para generar *embeddings* visuales que se centran en las regiones más informativas de la imagen, la capa de *pooling* global se implementa utilizando WELDON *pooling* [89], definido como:

$$\boldsymbol{h}(k) = \max_{\boldsymbol{H} \in \mathcal{H}_K} \frac{1}{K} \sum_{i,j} \boldsymbol{H}(i,j)\mathbf{G}(k,i,j) + \min_{\boldsymbol{H} \in \mathcal{H}_K} \frac{1}{K} \sum_{i,j} \boldsymbol{H}(i,j)\mathbf{G}(k,i,j), \tag{3.14}$$

donde $\mathcal{H}_K$ es el conjunto de matrices $\boldsymbol{H}$ con las mismas dimensiones espaciales que $\mathbf{G}$, tales que todos sus elementos cumplen con $\boldsymbol{H}(i,j) \in \{0,1\}$, y cuya suma es $\sum_{i,j} \boldsymbol{H}(i,j) = K$. Esta operación generaliza el *average pooling* usual, permitiendo restringir el cómputo del promedio para cada canal a los K pixeles con mayor y menor activación, que aportan información positiva y negativa para la predicción de la característica correspondiente, respectivamente. Esto suprime la propagación de gradientes asociados a regiones menos informativas de la imagen durante el entrenamiento.

Por otra parte, la rama textual codifica las palabras de la descripción de entrada utilizando *embeddings* Word2vec [41] pre-entrenados, que sirven de entrada para una red neuronal

recurrente. La salida final de esta red se divide por su norma L^2 para obtener el *embedding* textual final $\boldsymbol{v} \in \mathbb{R}^d$.

Este modelo se entrena utilizando la siguiente función de pérdida basada en *triplet loss*:

$$\ell_{contrastive} = \frac{1}{|\mathcal{B}|} \sum_{n \in \mathcal{B}} \Big(\max_{m \in \mathcal{B} \setminus \{n\}} \ell_{triplet}(\boldsymbol{x}_n, \boldsymbol{v}_n, \boldsymbol{v}_m) + \max_{m \in \mathcal{B} \setminus \{n\}} \ell_{triplet}(\boldsymbol{v}_n, \boldsymbol{x}_n, \boldsymbol{x}_m) \Big) \quad (3.15)$$

donde $\boldsymbol{x}_n$ es el *embedding* visual de la n-ésima imagen, $\boldsymbol{v}_n$ es el *embedding* textual de la n-ésima descripción, y $\mathcal{B}$ es el conjunto de índices que conforma un determinado *minibatch*. La función $\ell_{triplet}$ corresponde a una *triplet loss*, definida para una tripleta de *embeddings* de la forma $(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{z'})$ como:

$$\ell_{triplet}(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{z'}) = \max\big\{0, \alpha - \langle \boldsymbol{y}, \boldsymbol{z} \rangle + \langle \boldsymbol{y}, \boldsymbol{z'} \rangle\big\}, \quad (3.16)$$

donde $\alpha > 0$ es un margen escalar. Esta función de pérdida promueve que el *embedding* "ancla" $\boldsymbol{y}$ se parezca más al *embedding* relevante (positivo) $\boldsymbol{z}$, que a un *embedding* irrelevante (negativo) $\boldsymbol{z'}$. La selección de los máximos valores de $\ell_{triplet}$ para cada ejemplo del *minibatch* en (3.15) se conoce como *hard negative mining*, y permite focalizar la supervisión en los ejemplos más complejos, lo que mejora significativamente la calidad de los resultados. Sin embargo, esto también obliga a computar los productos internos entre cada par multi-modal del *minibatch*, lo que aumenta significativamente el costo computacional de entrenar este modelo.

Para generar mapas de localización una vez entrenada la red, se descarta la capa de *pooling* en la rama visual, y los pesos de la última capa *fully-connected* se proyectan directamente sobre $\mathbf{G}$ para obtener un mapa de características localizadas $\mathbf{G'} \in \mathbb{R}^{d \times h \times w}$. Dado un *embedding* textual $\boldsymbol{v}$ asociado a una descripción que se desea localizar, se puede computar un mapa de activación $\boldsymbol{M} \in \mathbb{R}^{h \times w}$ en el que cada elemento $\boldsymbol{M}(i, j)$ está dado por:

$$\boldsymbol{M}(i, j) = \sum_{k \in K(\boldsymbol{v})} |\boldsymbol{v}(k)| \cdot \mathbf{G'}(k, i, j), \quad (3.17)$$

donde $K(\boldsymbol{v})$ representa el conjunto de índices de los 180 componentes de $\boldsymbol{v}$ con la mayor activación. Los autores proponen binarizar $\boldsymbol{M}$ utilizando como umbral el promedio entre su menor y mayor activación.

3.5 Detección de Objetos Utilizando Descripciones

Recientemente, se han propuesto en la literatura algunos trabajos que abordan el problema relacionado de detección de objetos con supervisión débil (WSOD, por sus siglas en inglés) utilizando únicamente descripciones en lenguaje natural [90, 91]. Estos métodos modelan la tarea de WSOD como un problema de *multiple-instance learning* (MIL), en el que se considera una imagen como un conjunto de regiones independientes. Se utiliza entonces el supuesto de que las etiquetas negativas para la imagen son negativas para cada región, y que las etiquetas positivas para la imagen son positivas para al menos una región. Para dividir la imagen en regiones, estos métodos utilizan propuestas de *bounding boxes* generadas por modelos pre-entrenados, por lo que no pueden ser directamente aplicados a WSSS. Sin embargo, ambos problemas comparten algunos desafíos similares.

Concretamente, en [90], se propone un módulo de inferencia de etiquetas para identificar categorías de objetos relevantes mencionadas en descripciones, lo que coincide con uno de los objetivos del módulo de procesamiento de texto propuesto en esta tesis. Este módulo de inferencia de etiquetas extrae representaciones vectoriales para cada palabra de una descripción a partir de *embeddings* GloVe pre-entrenados, que luego combina mediante *max pooling*. El vector resultante se utiliza como entrada para un clasificador *fully-connected*, que predice las categorías de objetos relevantes que están implícitas en la descripción. Sin embargo, este modelo requiere de una base de datos de entrenamiento con pares de descripciones y etiquetas de clasificación asociados, lo que es costoso de generar, y no es aplicable para extraer información complementaria como atributos visuales, ni permite descubrir categorías de objetos adicionales. Por el contrario, el módulo de procesamiento de texto propuesto en la presente tesis no requiere de supervisión adicional, lo que es crucial para poder extenderlo fácilmente a otras bases de datos y categorías de objetos, y permite extraer una supervisión más completa a partir de descripciones.

Más recientemente, en [91] se sigue una estrategia similar a la utilizada en el presente trabajo de tesis para extraer objetos y atributos a partir de descripciones en lenguaje natural utilizando el extractor de grafos de escena de [24]. Sin embargo, en [91] esta información se utiliza para vincular las predicciones de categorías y atributos visuales de una determinada *bounding box* en un contexto de *multiple-instance learning*. Por el contrario, en la presente tesis se utilizan los pares categoría-atributo para entrenar un espacio vectorial multi-modal, que puede ser explotado para generar mapas de activación guiados por ambos tipos de información visual.

Capítulo 4

Metodología Propuesta

En este capítulo se detallan los pormenores de la metodología propuesta, cuyas componentes principales se muestran esquemáticamente en el diagrama de la Fig. 4.1. En primer lugar, se propone un módulo de procesamiento de texto para extraer una representación estructurada de la información visual relevante presente en descripciones de imágenes. Esta supervisión se utiliza tanto para entrenar una red de localización, como para guiar la síntesis de máscaras de segmentación a partir de los mapas de localización generados por el modelo ya entrenado. Las siguientes secciones presentan los detalles de cada etapa del proceso.

4.1 Módulo de Procesamiento de Texto

El módulo de procesamiento de texto propuesto tiene por objetivo consolidar las menciones a distintas categorías de objetos, así como sus respectivos atributos visuales, a un nivel semántico que sea útil para la tarea de WSSS, además de filtrar información ambigua o no visual presente en las descripciones en lenguaje natural. Esto se logra utilizando la estructura sintáctica de las descripciones, dada por su árbol de dependencia, para identificar palabras y frases que describen objetos y sus atributos. Las palabras que describen objetos son posteriormente asignadas a su correspondiente *synset* en el grafo de WordNet, utilizando un modelo de *word sense disambiguation*. Esto permite utilizar las relaciones semánticas definidas por este grafo para relacionar cada objeto mencionado en la descripción con su correspondiente etiqueta semántica para segmentación. La Fig. 4.2 muestra un resumen del proceso, que se describe en detalle a continuación.

Cabe notar que, dado que las descripciones disponibles para el entrenamiento están en inglés, la descripción del modelo propuesto y los ejemplos presentados se enfocan en este idioma. No obstante lo anterior, los conceptos y herramientas utilizados son compartidos por otros idiomas similares, por lo que el desarrollo realizado podría generalizarse con facilidad.

4.1.1 Análisis Sintáctico

Para la etapa de análisis sintáctico se utiliza una versión modificada del extractor de grafos de escena de [24], que se describe en la Sección 3.1.6, basada en una re-implementación pública en

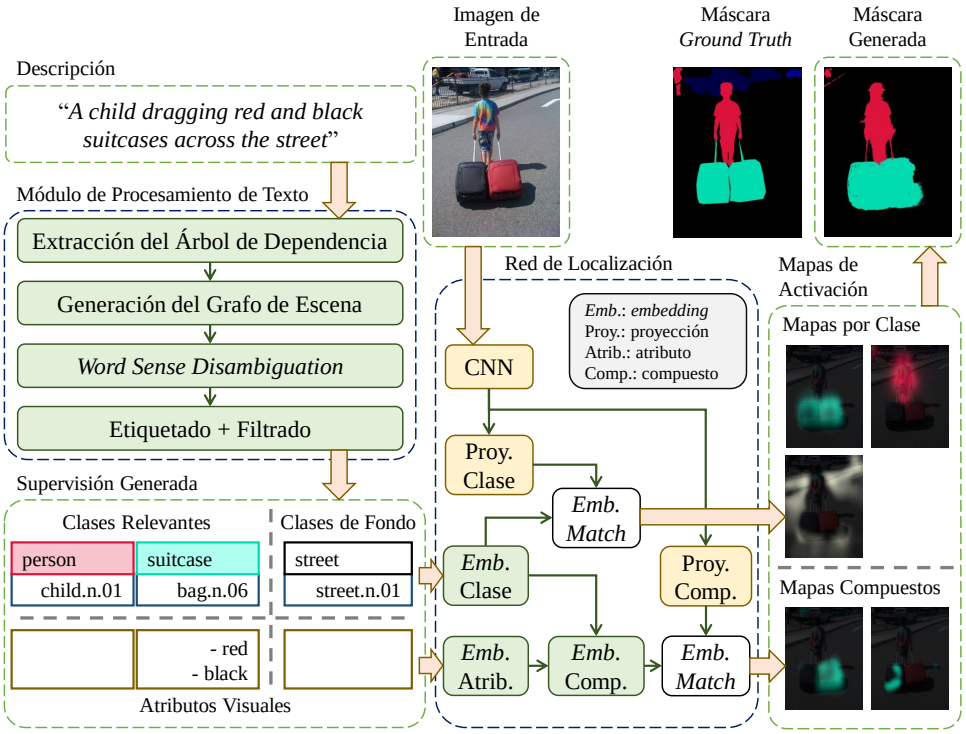


Figura 4.1: Diagrama general de la metodología propuesta. Un módulo de procesamiento de texto combina información sobre estructuras sintácticas con relaciones semánticas definidas por una base de conocimiento para extraer información visual útil a partir de descripciones. La supervisión resultante consiste en etiquetas de clasificación para objetos relevantes y de fondo, además de sus respectivos atributos visuales. Estas etiquetas se utilizan para entrenar una red de localización, que aprende a proyectar categorías y compuestos categoría-atributo a espacios vectoriales compartidos con imágenes. Este modelo permite obtener mapas de activación guiado por distintos tipos de información visual, que posteriormente se utilizan para generar máscaras de segmentación artificiales con las que se entrena un modelo supervisado.

Python[1]. En primer lugar, este extractor aplica una etapa de pre-procesamiento de lenguaje estándar para extraer un árbol de dependencia a partir de la descripción de entrada. El árbol de dependencia resultante contiene información acerca de las distintas estructuras sintácticas de la descripción y sus relaciones, como se muestra en la Fig. 4.2.

Posteriormente, se identifican todas las frases sustantivas como potenciales candidatos a objeto, y se asignan sus correspondientes atributos visuales utilizando los patrones sintácticos definidos por el modelo basado en reglas. Contrario a la implementación original del generador de grafos de escena, se omiten los pasos de réplica de nodos basado en cuantificadores, y la resolución de pronombres. Las relaciones entre objetos extraídas por el modelo también se ignoran.

En lugar de simplemente utilizar el núcleo de cada frase sustantiva como la categoría del objeto correspondiente como en la implementación original, en este caso se identifica un "núcleo extendido" como el segmento más largo de la frase que al mismo tiempo (1) incluye

[1] `https://github.com/vacancy/SceneGraphParser`

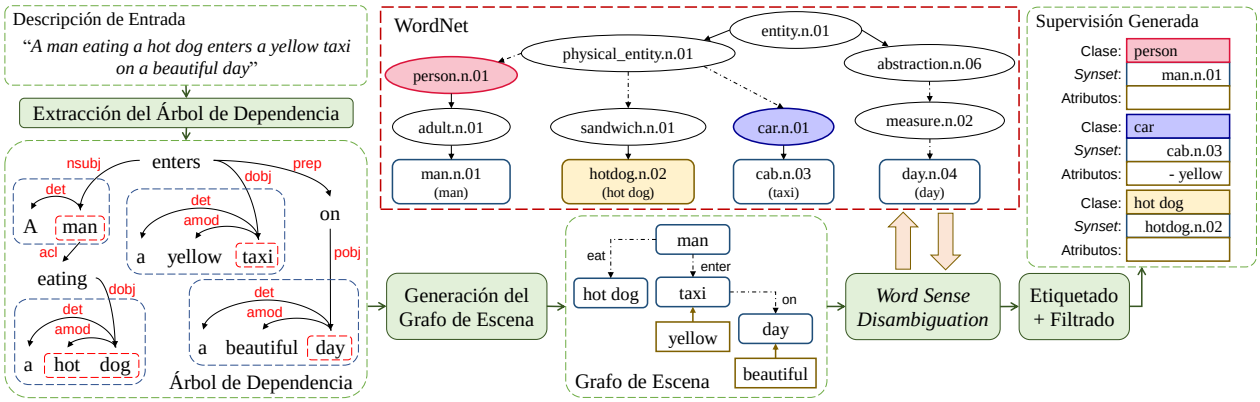


Figura 4.2: Diagrama del módulo de procesamiento de texto propuesto. En primer lugar, un árbol de dependencia es generado a partir de la descripción de entrada, y luego se utiliza un extractor de grafos de escena para identificar candidatos a objetos y sus respectivos atributos visuales. Posteriormente, los núcleos extendidos de cada frase sustantiva son ubicados dentro del grafo de WordNet utilizando un modelo de *word sense disambiguation*. Finalmente, se identifican aquellos candidatos a objeto que corresponden a categorías semánticas útiles utilizando las relaciones definidas por el grafo de WordNet, y el resto son descartados.

el sustantivo núcleo de la frase, y (2) está definido como sustantivo en WordNet. Este núcleo extendido se utiliza posteriormente para inferir la categoría semántica del candidato a objeto correspondiente.

Cabe destacar que, contrario a la estrategia usada en [17] que identifica categorías de objetos presentes en la descripción mediante búsqueda exacta del nombre de la clase, considerar la estructura sintáctica de la frase permite filtrar casos en los que nombres de clases se usan con otras categorías gramaticales. Por ejemplo, la palabra "*orange*" utilizada como un adjetivo o la palabra "*train*" utilizada como un verbo (ambas nombres de clases en MS-COCO), en lugar de sustantivos. Adicionalmente, al considerar únicamente los núcleos extendidos para inferir las categorías semánticas de los objetos, es posible manejar casos en los que sustantivos compuestos modifican la categoría del núcleo. Por ejemplo, el núcleo extendido "*microwave oven*" corresponde a la categoría "*microwave*" y no a la categoría "*oven*", mientras que "*hot dog*" es una categoría separada de "*dog*".

4.1.2 *Word Sense Disambiguation*

Para poder aprovechar la información codificada en el grafo de WordNet, en primer lugar es necesario mapear cada candidato a objeto a su *synset* correspondiente. Esta tarea de determinar el significado de una palabra en un contexto específico se conoce como *word sense disambiguation* (WSD), y constituye un problema complejo en el área de procesamiento de lenguaje natural. Para implementar esta etapa, se utiliza el modelo del estado del arte para WSD propuesto en [59], que se describe en la Sección 3.1.7.

Concretamente, se utiliza la versión del modelo basada en BERT [44], con los pesos preentrenados publicados por los autores originales[2], pero levemente modificado para manejar

[2] `https://github.com/getalp/disambiguate`

sustantivos compuestos. Para esto, se restringe la predicción de significados para cada frase sustantiva al conjunto de posibles *synsets* asociados a su núcleo extendido, utilizando las probabilidades predichas por el modelo para el primer *token* perteneciente a la frase. Esto sigue la estrategia utilizada en la implementación original para manejar palabras que se dividen en múltiples *tokens* por BERT.

Esta etapa es crucial tanto para prevenir falsos positivos (e.g., dependiendo del contexto, "*mouse*" puede referirse al animal, en lugar del dispositivo electrónico que corresponde para la clase de MS-COCO con ese nombre), así como para identificar sinónimos (e.g., la categoría "*couch*" puede ser mencionada de forma equivalente como "*sofa*").

Cabe destacar que, si bien las etapas de análisis sintáctico y WSD utilizan indirectamente supervisión adicional correspondiente a bases de datos de lenguaje, las anotaciones respectivas son independientes del dominio particular de imágenes, por lo que no implican una restricción para extender el modelo de segmentación a nuevas aplicaciones y categorías de objetos.

4.1.3 Etiquetado y Filtrado

Finalmente, se utilizan las relaciones definidas por el grafo de WordNet para asignar a cada candidato a objeto una etiqueta semántica escogida entre un conjunto pre-definido $\mathcal{C}$. Cada etiqueta en $\mathcal{C}$ se asocia previamente a un *synset* de WordNet, de modo que cualquier candidato que corresponda a un sinónimo, hipónimo, u holónimo de un *synset* etiquetado puede ser asignado a la misma categoría. Por ejemplo, esto permite identificar la palabra "*penguin*" como un subtipo de la clase "*bird*", y a "*people*" como un conjunto de objetos de la clase "*person*".

En casos en que un determinado objeto sea hipónimo de más de un *synset* etiquetado, se le asigna la categoría del hiperónimo más cercano dentro del grafo. Por otra parte, las relaciones de holonimia se consideran solo en el caso de que todos los merónimos de un objeto corresponden a la misma categoría, es decir, para conjuntos de elementos homogéneos. Los objetos que no correspondan a ninguna categoría en $\mathcal{C}$ son descartados. De forma similar, los atributos se asignan a un conjunto pre-definido de etiquetas $\mathcal{A}$, y los atributos sin etiqueta correspondiente son descartados.

4.1.4 Categorías Complementarias

En la práctica, las aplicaciones de segmentación semántica definen solo un conjunto fijo de categorías relevantes o de *foreground*, $\mathcal{C}_{\text{fg}}$, correspondientes a las categorías que se desea segmentar. Sin embargo, la información sobre otras categorías de objetos presentes en la imagen puede ser aprovechada para mejorar la calidad de las máscaras de segmentación generadas. Dado que todos los pixeles que no corresponden a clases en $\mathcal{C}_{\text{fg}}$ deben ser asignadas a una etiqueta especial de "fondo" $\langle \mathbf{bg} \rangle$ por el modelo de segmentación final, se denominan estas categorías como clases de fondo o *background*, y se asignan a un conjunto separado de etiquetas $\mathcal{C}_{\text{bg}}$. De esta manera, $\mathcal{C}$ puede obtenerse como la unión de ambos conjuntos, $\mathcal{C} = \mathcal{C}_{\text{fg}} \cup \mathcal{C}_{\text{bg}}$. Cabe destacar que la división entre categorías relevantes y categorías de fondo es arbitraria, y depende solo de la aplicación.

Dado que los *synsets* no asignados a etiquetas en $\mathcal{C}_{\text{fg}}$ contienen inicialmente sustantivos

no-localizables, además de una granularidad excesiva, se requiere de filtrados adicionales para construir $\mathcal{C}_{\text{bg}}$. Una posibilidad consiste en definir $\mathcal{C}_{\text{bg}}$ manualmente, eligiendo categorías de fondo útiles para una aplicación particular, de manera análoga a $\mathcal{C}_{\text{fg}}$. Alternativamente, se propone la siguiente metodología para inicializar este conjunto de manera semi-automática, a partir de los mismos *synsets* presentes en las descripciones de la base de datos:

1. Inicialmente, se descartan todos los hiperónimos de *synsets* etiquetados en $\mathcal{C}_{\text{fg}}$, dado que estos describen categorías de objetos demasiado diversas (e.g., "*furniture*", "*animal*", "*tool*", etc.), además de todos sus merónimos, ya que estos corresponden a partes de objetos etiquetados (e.g., "*wing*", "*wheel*", "*windshield*", etc.).
2. Se descartan los hipónimos de *synsets* para "*abstraction*" (dado que estos describen mayoritariamente información no visual) y "*part*" (correspondientes a objetos incompletos), además de "*room*" y "*location*" (que usualmente describen escenas completas).
3. Entre los *synsets* sin etiquetar restantes, se selecciona el que aparezca con mayor frecuencia en la base de datos, y se le asigna una nueva etiqueta, extendiendo $\mathcal{C}_{\text{bg}}$. Adicionalmente, todos sus hipónimos y holónimos se asignan a la misma etiqueta, mientras que todos sus hiperónimos y merónimos son descartados, por las razones descritas anteriormente.
4. El paso (3) se repite hasta que no queden *synsets* sin etiquetar que excedan una frecuencia dada T.

Este procedimiento prioriza la asignación de etiquetas a niveles semánticos que son utilizados frecuentemente por los anotadores para describir objetos, a la vez que aprovecha el grafo de WordNet para mantener la consistencia del conjunto. Las etiquetas resultantes pueden ser refinadas manualmente para ajustarse a la aplicación particular, como se describe en la Sección 4.5.4 para el caso de MS-COCO.

4.1.5 Base de Datos Procesada

Una vez aplicado el procedimiento descrito anteriormente para procesar las descripciones disponibles, la base de datos utilizada para el entrenamiento puede definirse más formalmente como un conjunto de tuplas de la siguiente forma:

$$\tilde{\mathcal{D}} = \{(\mathbf{I}_n, \mathcal{C}_n^{\text{fg}}, \mathcal{C}_n^{\text{bg}}, \{\mathcal{A}_{n,c}\}_{c \in \mathcal{C}_n})\}_{n=1}^{N}, \tag{4.1}$$

donde $\mathbf{I}_n$ es la n-ésima imagen de la base de datos; N es el número total de imágenes; $\mathcal{C}_n^{\text{fg}} \subseteq \mathcal{C}_{\text{fg}}$ y $\mathcal{C}_n^{\text{bg}} \subseteq \mathcal{C}_{\text{bg}}$ son los conjuntos de categorías relevantes y de fondo detectadas en las descripciones de la n-ésima imagen, respectivamente; $\mathcal{C}_n = \mathcal{C}_n^{\text{fg}} \cup \mathcal{C}_n^{\text{bg}}$ es el conjunto de todas las categorías detectadas para la n-ésima imagen; y $\mathcal{A}_{n,c} \subseteq \mathcal{A}$ es el conjunto de atributos visuales asociados con una determinada clase $c \in \mathcal{C}_n$.

Adicionalmente, se define como $\mathcal{P}_n$ al conjunto de todos los pares categoría-atributos detectados para la n-ésima imagen, i.e.,

$$\mathcal{P}_n = \{(c, a) \mid c \in \mathcal{C}_n, a \in \mathcal{A}_{n,c}\}. \tag{4.2}$$

4.2 Red de Localización

4.2.1 Arquitectura

Siguiendo trabajos previos [17, 18, 22, 88], la red de localización propuesta proyecta imágenes y conceptos semánticos a un mismo espacio vectorial multi-modal. Para esto, en primer lugar se asigna cada categoría c a un *embedding* vectorial $\boldsymbol{e}_c^{\text{cls}} \in \mathbb{R}^{d_{\text{cls}}}$, y cada atributo a a un *embedding* vectorial $\boldsymbol{e}_a^{\text{attr}} \in \mathbb{R}^{d_{\text{attr}}}$. Pares compuestos del tipo categoría-atributo $(c, a) \in \mathcal{C} \times \mathcal{A}$ se codifican utilizando *embeddings* compuestos $\boldsymbol{e}_{c,a}^{\text{comp}} \in \mathbb{R}^{d_{\text{comp}}}$, computados como:

$$\boldsymbol{e}_{c,a}^{\text{comp}} = \boldsymbol{W}_{\text{comp}} \left[\boldsymbol{e}_c^{\text{cls}}; \boldsymbol{e}_a^{\text{attr}}\right] + \boldsymbol{b}_{\text{comp}}, \tag{4.3}$$

donde $\boldsymbol{W}_{\text{comp}} \in \mathbb{R}^{d_{\text{comp}} \times (d_{\text{cls}} + d_{\text{attr}})}$ y $\boldsymbol{b}_{\text{comp}} \in \mathbb{R}^{d_{\text{comp}}}$ son los parámetros de una transformación afín, y $[\,\cdot\,;\,\cdot\,]$ denota concatenación de vectores.

Por otra parte, la imagen de entrada $\mathbf{I}_n$ se codifica utilizando una red neuronal convolucional $\phi_{\text{CNN}}(\,\cdot\,;\boldsymbol{\theta}_{\text{CNN}})$ parametrizada por $\boldsymbol{\theta}_{\text{CNN}}$, para obtener un mapa de características $\mathbf{F} = \phi_{\text{CNN}}(\mathbf{I}_n; \boldsymbol{\theta}_{\text{CNN}}) \in \mathbb{R}^{d_{\text{enc}} \times h_{\text{enc}} \times w_{\text{enc}}}$, con d_{enc} canales y dimensiones espaciales $(h_{\text{enc}}, w_{\text{enc}})$, Para implementar ϕ_{CNN}, se evalúan múltiples arquitecturas convolucionales comunes en la literatura de WSSS, como se describe en la Sección 4.5.4.

Posteriormente, se utilizan dos capas convolucionales f_{cls} y f_{comp} con *kernel* de tamaño 1×1 para proyectar $\mathbf{F}$ a los espacios vectoriales correspondientes a categorías y a compuestos, respectivamente:

$$\mathbf{F}_{\text{cls}} = f_{\text{cls}}(\mathbf{F}; \boldsymbol{\theta}_{\text{cls}}), \tag{4.4}$$

$$\mathbf{F}_{\text{comp}} = f_{\text{comp}}(\mathbf{F}; \boldsymbol{\theta}_{\text{comp}}). \tag{4.5}$$

Aquí, $\mathbf{F}_{\text{cls}} \in \mathbb{R}^{d_{\text{cls}} \times h_{\text{enc}} \times w_{\text{enc}}}$ y $\mathbf{F}_{\text{comp}} \in \mathbb{R}^{d_{\text{comp}} \times h_{\text{enc}} \times w_{\text{enc}}}$ son los mapas de características proyectados, y $\boldsymbol{\theta}_{\text{cls}}$ y $\boldsymbol{\theta}_{\text{comp}}$ son los parámetros de sus respectivas capas convolucionales. Este paso final de proyección le permite al modelo suprimir o realzar la importancia de características asociadas a información útil para cada tipo de predicción, sin competencia directa entre ambas modalidades.

Finalmente, se aplica una operación de *pooling* global sobre $\mathbf{F}_{\text{cls}}$ y $\mathbf{F}_{\text{comp}}$ para obtener los *embeddings* visuales $\boldsymbol{v}_{\text{cls}} \in \mathbb{R}^{d_{\text{cls}}}$ y $\boldsymbol{v}_{\text{comp}} \in \mathbb{R}^{d_{\text{comp}}}$, respectivamente. Siguiendo [22], esta operación se implementa utilizando una capa de *pooling* tipo WELDON [89], definida por (3.14), lo que ayuda a focalizar los gradientes del modelo en las regiones más informativas de la imagen.

Dada una pareja de *embeddings* visuales y semánticos correspondientes, se computa la probabilidad de que la imagen representada por el *embedding* visual contenga un objeto compatible con el *embedding* semántico de la siguiente manera:

$$p_c^{\text{cls}} = \sigma(\langle \boldsymbol{v}_{\text{cls}}, \boldsymbol{e}_c^{\text{cls}} \rangle), \tag{4.6}$$

$$p_{c,a}^{\text{comp}} = \sigma(\langle \boldsymbol{v}_{\text{comp}}, \boldsymbol{e}_{c,a}^{\text{comp}} \rangle), \tag{4.7}$$

donde $\langle\,\cdot\,,\,\cdot\,\rangle$ representa un producto punto, y $\sigma(\,\cdot\,)$ representa la función logística, de modo que $\sigma(x) = 1/(1 + e^{-x})$. Cabe notar que la red de localización se reduce a un clasificador

convolucional usual cuando solo se utilizan clases relevantes, dado que el producto punto con los *embeddings* correspondientes es matemáticamente equivalente a aplicar una capa *fully-connected* de clasificación sin *bias* sobre una representación visual global.

4.2.2 Entrenamiento

El entrenamiento de la red de localización propuesta se supervisa utilizando la siguiente función de pérdida:

$$\ell = \lambda_{\text{fg}}\ell_{\text{fg}} + \lambda_{\text{bg}}\ell_{\text{bg}} + \lambda_{\text{comp}}\ell_{\text{comp}}, \tag{4.8}$$

donde ℓ_{fg} y ℓ_{bg} supervisan las predicciones para categorías relevantes y de fondo, respectivamente; ℓ_{comp} supervisa predicciones para pares categoría-atributo; y λ_{fg}, λ_{bg}, y λ_{comp} son parámetros escalares que ajustan la contribución de cada componente.

Para implementar ℓ_{fg}, se adopta la siguiente variante balanceada de la entropía cruzada binaria o *binary cross-entropy* (BCE):

$$\ell_{\text{fg}} = -\frac{1}{|\mathcal{C}_{\text{fg}}|}\left[\sum_{c\in\mathcal{C}_n^{\text{fg}}} w_c^+ \log(p_c^{\text{cls}}) + \sum_{c'\in\mathcal{C}_{\text{fg}}\setminus\mathcal{C}_n^{\text{fg}}} w_{c'}^- \log(1-p_{c'}^{\text{cls}})\right] \tag{4.9}$$

donde $\setminus$ representa resta entre conjuntos; y w_c^+ y w_c^- son pesos escalares que balancean la contribución de ejemplos positivos y negativos para cada clase. Los valores de estos pesos son proporcionales al inverso de la frecuencia de ejemplos positivos y negativos presentes en la base de datos, siguiendo esquemas similares propuestos en la literatura para manejar desbalance entre clases en problemas como clasificación y detección [92, 93]. Los detalles del cómputo de estos pesos se presentan en el Anexo A.

La componente ℓ_{bg} se computa de la misma forma que ℓ_{fg}, pero considerando categorías de fondo, es decir, reemplazando $\mathcal{C}_{\text{fg}}$ por $\mathcal{C}_{\text{bg}}$ y $\mathcal{C}_n^{\text{fg}}$ por $\mathcal{C}_n^{\text{bg}}$ en (4.9). La separación de ambas componentes permite ajustar la contribución de cada tipo de categoría durante el entrenamiento.

De manera similar, ℓ_{comp} se computa como:

$$\ell_{\text{comp}} = -\frac{1}{|\mathcal{P}_n| + |\tilde{\mathcal{P}}_n^-|}\left[\sum_{(c,a)\in\mathcal{P}_n} \tilde{w}_a^+ \log(p_{c,a}^{\text{comp}}) + \sum_{(c',a')\in\tilde{\mathcal{P}}_n^-} \tilde{w}_{a'}^- \log(1-p_{c',a'}^{\text{comp}})\right]. \tag{4.10}$$

En este caso, se considera solo un subconjunto $\tilde{\mathcal{P}}_n^-$ de pares negativos, que se construye de la siguiente manera: para cada par positivo $(c,a)\in\mathcal{P}_n$, se seleccionan todos los elementos de $\{(c,\tilde{a})\,|\,\tilde{a}\in\mathcal{A}\setminus\mathcal{A}_{n,c}\}$, correspondientes a intercambiar a por un atributo negativo para c, y se seleccionan N_{comp}^- elementos aleatorios de $\{(\tilde{c},a)\,|\,\tilde{c}\in\mathcal{C}\setminus\mathcal{C}_n\}$, correspondientes a intercambiar c por una categoría negativa para la imagen. Los pesos $\tilde{w}_a^+$ y $\tilde{w}_a^-$ cumplen un rol análogo al de w_c^+ y w_c^-, pero considerando las frecuencias positivas y negativas de cada atributo a, ignorando en este caso la frecuencia de las clases. Nuevamente, los detalles de cómputo para estos pesos se incluyen en el Anexo A.

4.3 Generación de Máscaras de Segmentación

En esta sección se describe cómo generar máscaras de segmentación semántica a partir de las representaciones aprendidas por la red de localización propuesta. Siguiendo un procedimiento análogo al descrito en [12] para generar CAMs, descrito en la Sección 3.3.1, se computan mapas de activación $\boldsymbol{M}_c^{\text{cls}} \in \mathbb{R}^{h_{\text{enc}} \times w_{\text{enc}}}$ para cada clase positiva $c \in \mathcal{C}_n$ mediante una suma ponderada de los canales del último mapa de características antes de la operación de *pooling* global, $\mathbf{F}_{\text{cls}}$. En este caso, los pesos de cada canal están dados por las componentes del *embedding* semántico correspondiente, $\boldsymbol{e}_c^{\text{cls}}$, de modo que cada elemento de $\boldsymbol{M}_c^{\text{cls}}$ se define por:

$$\boldsymbol{M}_c^{\text{cls}}(i,j) = \sum_{k=1}^{d_{\text{cls}}} \boldsymbol{e}_c^{\text{cls}}(k) \cdot \mathbf{F}_{\text{cls}}(k,i,j), \tag{4.11}$$

donde se utilizan paréntesis para denotar indexación, al igual que en (3.12). Análogamente, es posible obtener mapas de activación compuestos $\boldsymbol{M}_{c,a}^{\text{comp}} \in \mathbb{R}^{h_{\text{enc}} \times w_{\text{enc}}}$ asociados a pares positivos $(c,a) \in \mathcal{P}_n$ como:

$$\boldsymbol{M}_{c,a}^{\text{comp}}(i,j) = \sum_{k=1}^{d_{\text{comp}}} \boldsymbol{e}_{c,a}^{\text{comp}}(k) \cdot \mathbf{F}_{\text{comp}}(k,i,j). \tag{4.12}$$

Todos los mapas asociados con una determinada categoría c son combinados mediante suma pixel a pixel, para obtener mapas $\tilde{\boldsymbol{M}}_c$:

$$\tilde{\boldsymbol{M}}_c(i,j) = \boldsymbol{M}_c^{\text{cls}}(i,j) + \sum_{a \in \mathcal{A}_{n,c}} \boldsymbol{M}_{c,a}^{\text{comp}}(i,j). \tag{4.13}$$

A estos mapas se les aplica un umbral en 0 para suprimir activaciones negativas, y se normalizan dividiendo por su máxima activación para obtener los mapas finales por clase $\boldsymbol{M}_c$, similar a [9, 17]:

$$\boldsymbol{M}_c(i,j) = \frac{\text{ReLU}(\tilde{\boldsymbol{M}}_c(i,j))}{\max\limits_{k,l} \text{ReLU}(\tilde{\boldsymbol{M}}_c(k,l))}. \tag{4.14}$$

Para computar el mapa de activación para la clase fondo $\langle \mathbf{bg} \rangle$, se combinan dos tipos distintos de información. En primer lugar, siguiendo trabajos anteriores [9, 17], se genera un mapa de activación $\boldsymbol{M}_{\langle \mathbf{bg} \rangle}^{-}$ utilizando información negativa dada por los mapas asociados a categorías relevantes, tal que:

$$\boldsymbol{M}_{\langle \mathbf{bg} \rangle}^{-}(i,j) = \left\{ 1 - \max_{c \in \mathcal{C}_{\text{fg}}} \boldsymbol{M}_c(i,j) \right\}^{\alpha}, \tag{4.15}$$

de modo que a pixeles con baja activación para todas las clases relevantes se les asigna una mayor probabilidad de pertenecer al fondo. El parámetro $\alpha \geqslant 1$ controla la magnitud de las activaciones de este mapa.

Adicionalmente, se computa un segundo mapa de activación $\boldsymbol{M}_{\langle \mathbf{bg} \rangle}^{+}$ utilizando información positiva, dada por los mapas predichos por el modelo para categorías de fondo. Este

mapa se computa aplicando una suma pixel a pixel sobre los mapas $\tilde{\boldsymbol{M}}_c, \forall c \in \mathcal{C}_n^{\text{bg}}$, y luego normalizando como en (4.14):

$$\tilde{\boldsymbol{M}}^+_{\langle \mathbf{bg} \rangle}(i,j) = \sum_{c \in \mathcal{C}_{\text{bg}}} \tilde{\boldsymbol{M}}_c(i,j), \tag{4.16}$$

$$\boldsymbol{M}^+_{\langle \mathbf{bg} \rangle}(i,j) = \gamma \cdot \frac{\text{ReLU}\big(\tilde{\boldsymbol{M}}^+_{\langle \mathbf{bg} \rangle}(i,j)\big)}{\max\limits_{k,l} \text{ReLU}\big(\tilde{\boldsymbol{M}}^+_{\langle \mathbf{bg} \rangle}(k,l)\big)}, \tag{4.17}$$

donde el parámetro escalar $\gamma < 1.0$ se agrega para evitar que clases de fondo dominen por sobre clases relevantes. El mapa resultante se combina con $\boldsymbol{M}^-_{\langle \mathbf{bg} \rangle}$ utilizando máximo pixel a pixel para obtener el mapa de activación final para la clase fondo, $\boldsymbol{M}_{\langle \mathbf{bg} \rangle}$:

$$\boldsymbol{M}_{\langle \mathbf{bg} \rangle}(i,j) = \max\{\boldsymbol{M}^-_{\langle \mathbf{bg} \rangle}(i,j), \boldsymbol{M}^+_{\langle \mathbf{bg} \rangle}(i,j)\}. \tag{4.18}$$

Finalmente, todos los mapas son escalados a la resolución de la imagen original, las activaciones para categorías no detectadas se fijan en 0, y se aplica *dense Conditional Random Fields* (dCRF) [46] para refinar las predicciones. Para obtener las máscaras de segmentación finales, se asigna a cada pixel la categoría correspondiente al *argmax* sobre los mapas de activación refinados.

4.4 Modelo Supervisado

Las máscaras de segmentación generadas con el método propuesto son utilizadas para entrenar un modelo de segmentación semántica en régimen supervisado. Siguiendo trabajos anteriores [7, 17, 11, 79], como modelo supervisado se utiliza DeepLab-ASPP [62] con arquitectura VGG-16. Como función de pérdida para esta etapa se utiliza *balanced seed loss*, propuesta en [7] para WSSS, definida como:

$$\ell_{seed} = -\frac{1}{\sum_{c \in \mathcal{C}_{\text{fg}}} |\mathcal{S}_c|} \sum_{c \in \mathcal{C}_{\text{fg}}} \sum_{(i,j) \in \mathcal{S}_c} \log(\boldsymbol{H}_c(i,j)) - \frac{1}{|\mathcal{S}_{\langle \mathbf{bg} \rangle}|} \sum_{(i,j) \in \mathcal{S}_{\langle \mathbf{bg} \rangle}} \log(\boldsymbol{H}_{\langle \mathbf{bg} \rangle}(i,j)), \tag{4.19}$$

donde $\mathcal{S}_c$ es el conjunto de ubicaciones espaciales (i,j) asignados a la categoría c en la máscara de supervisión artificial, y $\boldsymbol{H}_c(i,j)$ es la probabilidad predicha por el modelo para la categoría c en la ubicación (i,j). El sub-índice $\langle \mathbf{bg} \rangle$ indica la clase "fondo". Esta función de pérdida regula la contribución de pixeles etiquetados con la categoría fondo, que usualmente superan la cantidad de pixeles etiquetados con categorías relevantes por un margen significativo.

4.5 Metodología Experimental

4.5.1 Bases de Datos

MS-COCO

La mayoría de los experimentos reportados en la presente tesis utilizan la base de datos Microsoft Common Objects in Context (**MS-COCO**) [14]. Esta base de datos contiene 123287 imágenes en contextos cotidianos, cada una anotada con 5 descripciones en lenguaje natural,

además de máscaras de segmentación a nivel de instancia para 80 categorías de objetos distintas. Las categorías de objetos anotadas incluyen animales, comida, vehículos, muebles, y electrodomésticos, entre otros. Siguiendo trabajos previos [7, 17, 11, 79], se utiliza el subconjunto *train*2014 compuesto de 82783 imágenes para el entrenamiento de los modelos, mientras que el subconjunto *val*2014 con 40504 imágenes se reserva exclusivamente para evaluación. La estrategia utilizada para validación se describe en detalle en la Sección 4.5.2.

La elección de MS-COCO se debe principalmente a que esta es la base de datos pública de mayor tamaño que cuenta con imágenes anotadas tanto con descripciones en lenguaje natural, que son necesarias para el entrenamiento del modelo, como con máscaras de segmentación, que se utilizan para evaluar los resultados obtenidos. Esta base de datos fue utilizada previamente por el único trabajo existente que reporta resultados para el problema de segmentación semántica supervisada solo con descripciones [17], y ha sido ampliamente utilizada para evaluar modelos de WSSS en general [7, 94, 11, 79].

PASCAL VOC

Otra de las bases de datos más utilizadas por trabajos que abordan el problema de WSSS corresponde a **PASCAL VOC** [15], que cuenta con anotaciones de máscaras de segmentación a nivel de instancia para 20 categorías de objetos, correspondientes a un subconjunto de las 80 clases de MS-COCO. Debido a que esta base de datos no contiene descripciones, que son necesarias para el entrenamiento en el caso estudiado, en esta tesis PASCAL VOC se utiliza solo para evaluación, considerando las 1449 imágenes de su conjunto de validación, bajo un esquema de transferencia de aprendizaje, como se detalla en la Sección 5.4.

YouTube-Objects

La base de datos **YouTube-Objects** [95] se compone de videos recolectados de YouTube, obtenidos al buscar los nombres de 10 de las categorías semánticas de PASCAL VOC. Debido a que YouTube-Objects no cuenta con anotaciones para descripciones, se utiliza esta base de datos exclusivamente para evaluación, siguiendo un esquema de transferencia de aprendizaje, al igual que en el caso de PASCAL VOC. Para esta evaluación, se utiliza el subconjunto de cuadros anotados manualmente con máscaras a nivel de pixel facilitadas por [96], y los resultados obtenidos se reportan en la Sección 5.4.

4.5.2 Estrategia de Validación

Las máscaras de segmentación para el conjunto oficial de prueba de MS-COCO no son públicas, por lo que los trabajos anteriores de WSSS que reportan resultados en esta base de datos utilizan el conjunto de validación para evaluar los modelos propuestos. Sin embargo, esto deja MS-COCO sin un conjunto de validación propiamente tal. La mayoría de las publicaciones anteriores simplemente utiliza las máscaras de segmentación *ground truth* del conjunto de validación de PASCAL VOC para ajustar los hiperparámetros del modelo, y luego mantienen estos mismos valores durante los experimentos en MS-COCO. Sin embargo, esto no es posible en nuestro caso, debido a que PASCAL VOC no cuenta con descripciones para sus imágenes, que son necesarias para el entrenamiento del modelo propuesto.

En [17], que estudia el mismo problema que en el presente trabajo, no se define explí-

citamente un conjunto de validación. En la práctica, el ajuste de los hiperparámetros del modelo se realiza considerando las máscaras de segmentación del conjunto de entrenamiento mediante una evaluación indirecta, que corresponde a evaluar la calidad de las máscaras artificiales generadas para este conjunto. Los autores observan que el mIoU de las máscaras artificiales es un buen predictor del desempeño del modelo de segmentación final entrenado con esta supervisión. Sin embargo, esto introduce una dependencia, si bien indirecta, sobre las máscaras de segmentación *ground truth* de todo el conjunto de entrenamiento, que en principio no están disponibles en la configuración estudiada.

En la presente tesis, se adopta un esquema similar de validación indirecta basada en la calidad de las máscaras artificiales generadas, pero se restringe esta evaluación a un subconjunto aleatorio fijo de 4000 imágenes del conjunto de entrenamiento, lo que corresponde a menos del 5% del total. Para evitar confusiones con el conjunto de validación de MS-COCO, sobre el que se realiza la evaluación del modelo de segmentación final, este conjunto se denota exclusivamente como *trainval*4k durante el resto del documento.

4.5.3 Evaluación

Siguiendo trabajos anteriores [7, 17, 11, 79], la principal métrica de evaluación corresponde a la intersección sobre unión a nivel de pixel, promediada entre todas las categorías semánticas relevantes (incluyendo el fondo), denominada *mean intersection-over-union* (mIoU), según se define en la Sección 2.3.1.

4.5.4 Detalles de Implementación

Módulo de Procesamiento de Texto

Para aplicar el módulo de procesamiento de texto a la base de datos MS-COCO, en primer lugar se identifican los *synsets* correspondientes a cada una de las 80 categoría semánticas de esta base de datos, resolviendo manualmente casos en que el nombre de la clase posee múltiples posibles significados. En casos en los que las máscaras de segmentación *ground truth* asignan en la práctica una misma etiqueta semántica a objetos de distintos tipos, todos los *synsets* correspondientes se asocian a la misma clase. Por ejemplo, los objetos etiquetados con la clase "*tv*" incluyen instancias tanto de "*television_ receiver*.n.01" como de "*computer_ monitor*.n.01", por lo que ambos *synsets* se utilizan para etiquetar candidatos a objetos con esta categoría.

Las categorías que no tienen un *synset* correspondiente en WordNet ("*stop sign*" y "*potted plant*") se detectan solo en casos de menciones exactas del nombre la clase (o su plural), como en [17]. Adicionalmente, se extiende WordNet asignando algunos neologismos comunes a sus *synsets* más cercanos. Por ejemplo, "*smartphone*" se asigna a "*cellular_ telephone*.n.01", y "*macbook*" se asigna a "*laptop*.n.01".

Para obtener el conjunto de categorías de fondo $\mathcal{C}_n^{\text{bg}}$, se sigue el procedimiento descrito en la Sección 4.1.4, utilizando los *synsets* detectados en las descripciones de MS-COCO, y considerando los *synsets* asignados a las 80 categorías relevantes de esta base de datos para los filtrados iniciales. Usando un umbral de frecuencia $T = 200$, esto permite reducir más de 8k *synsets* a solo 119 categorías de objetos complementarias, que luego se filtran

manualmente para prevenir conflictos con las categorías relevantes. Por ejemplo, se descartan todos los hipónimos de "*clothing*.n.01", debido a que los artículos de ropa son generalmente etiquetados en las máscaras de *ground truth* con la misma categoría que las personas o animales que los utilizan. Se descartan también los *synsets* para almohada y mantel, ya que estos objetos no se identifican como categorías separadas de cama y mesa, respectivamente. La Tabla 4.1 muestra la lista final de 72 *synsets* utilizados para definir categorías de fondo en los experimentos realizados.

Tabla 4.1: Lista de los 72 *synsets* de WordNet utilizados para definir categorías de fondo durante los experimentos reportados. Estas categorías complementan las 80 categorías de objetos relevantes definidas en MS-COCO.

Index	Synset	Index	Synset	Index	Synset	Index	Synset	Index	Synset
81	street.n.01	96	tray.n.01	111	shower.n.01	126	painting.n.01	141	egg.n.02
82	plate.n.04	97	snow.n.01	112	lamp.n.02	127	engine.n.01	142	rice.n.01
83	tree.n.01	98	meat.n.01	113	cart.n.01	128	candle.n.01	143	goat.n.01
84	grass.n.01	99	ramp.n.01	114	tarmacadam.n.02	129	trail.n.02	144	hay.n.01
85	body_of_water.n.01	100	rock.n.01	115	curb.n.01	130	branch.n.02	145	pool.n.01
86	sky.n.01	101	shelf.n.01	116	rug.n.01	131	rack.n.05	146	ceiling.n.01
87	sidewalk.n.01	102	runway.n.04	117	basket.n.01	132	tile.n.01	147	roadway.n.01
88	floor.n.01	103	cabinet.n.01	118	home_plate.n.01	133	graffito.n.01	148	leash.n.01
89	pole.n.01	104	cheese.n.01	119	streetlight.n.01	134	curtain.n.01	149	roof.n.01
90	mirror.n.01	105	bathtub.n.01	120	step.n.04	135	potato.n.01	150	pepper.n.04
91	flower.n.01	106	pan.n.01	121	tomato.n.01	136	crossing.n.05	151	chest_of_drawers.n.01
92	track.n.09	107	shrub.n.01	122	bun.n.01	137	onion.n.01	152	soup.n.01
93	box.n.01	108	platform.n.01	123	post.n.04	138	highway.n.01		
94	railroad_track.n.01	109	countertop.n.01	124	pot.n.01	139	carriage.n.02		
95	light.n.02	110	pen.n.01	125	doorway.n.01	140	platter.n.01		

Por otra parte, debido a que el objetivo de utilizar atributos consiste en guiar la generación de mapas de localización, se restringe por defecto el conjunto de atributos $\mathcal{A}$ para etiquetar solo 40 palabras que describen distintos tipos de información visual de bajo nivel, tales como colores, materiales, o texturas. La lista de atributos utilizados se presenta en la Tabla 4.2.

Tabla 4.2: Lista de los 40 atributos visuales de bajo nivel utilizados en los experimentos reportados.

white	pink	tile/tiled	furry	leafy
black	grey/gray	wood/wooden	cloudy	glass
red	purple	grass/grassy	paved	brick
blue	gold/golden	rusty/rusted	snowy	dirt
green	beige	ceramic	muddy	stone
brown	silver	rock/rocky	fluffy	steel
yellow	checkered	metal/metallic	plastic	cement
orange	stripe/striped	concrete	sandy	leather

Arquitectura convolucional de la red de localización

Se consideran tres implementaciones distintas para el *backbone* convolucional ϕ_{CNN} de la red de localización propuesta, todos basados en arquitecturas convolucionales comunes en la literatura de segmentación semántica con supervisión débil. La mayor parte de los experimentos se realiza utilizando una arquitectura basada en ResNet-50 [33], con las modificaciones propuestas en [81], ya que se observa que esta arquitectura presenta un buen balance entre velocidad, número de parámetros, y calidad de los resultados. Siguiendo [81], para obtener

este *backbone* se elimina la operación de *pooling* global y la capa final *fully-connected* de una ResNet-50, lo que resulta en una arquitectura completamente convolucional. Adicionalmente, se reduce el *stride* de la última capa del penúltimo bloque de 2 a 1, lo que aumenta la resolución de la salida de 1/32 a 1/16 la resolución de la entrada. Excepto donde se indique explícitamente lo contrario, todos los resultados reportados fueron obtenidos con este *backbone*.

Para poder realizar una comparación más directa con trabajos anteriores (especialmente con [17]), se realizan también experimentos utilizando arquitecturas basadas en VGG-16 [31] y ResNet-38 [85], con las modificaciones descritas en [9]. En el caso de VGG-16, estas modificaciones consisten en reemplazar la primera y la segunda capa *fully-connected* por capas convolucionales con 1024 filtros de tamaño 1×1, y eliminar la tercera. Adicionalmente, se reduce el *stride* de las dos últimas capas de *downsampling* de 2 a 1, lo que resulta en salidas con 1/8 de la resolución de la entrada, y se aumenta la tasa de dilatación de las capas del último bloque convolucional de 1 a 2, para contrarrestar la reducción en el campo receptivo de las neuronas de salida debido al aumento en la resolución.

Por otra parte, en el caso de ResNet-38 simplemente se elimina la capa de *pooling* global y el clasificador *fully-connected*, y al igual que con VGG-16 se eliminan las dos últimas operaciones de *downsampling*, y se aumentan las tasas de dilatación de las capas del último y penúltimo bloque convolucional a 4 y 2, respectivamente.

Los pesos de todos los *backbones* se inicializan a partir de modelos pre-entrenados para clasificación en ImageNet [13], siguiendo la práctica estándar.

Hiperparámetros de la Red de Localización

El parámetro K de la capa de WELDON *pooling* se fija en 20% del total de pixeles del último mapa convolucional, siguiendo [22]. Los largos de todos los *embeddings* se fijan en $d_{\text{cls}} = d_{\text{attr}} = d_{\text{comp}} = 512$.

El modelo se entrena durante 15 épocas utilizando gradiente descendente estocástico con *minibatch* de tamaño 18, utilizando una tasa de aprendizaje inicial de 0.01. Siguiendo [17], se incorpora un término de penalización de la norma L^2 para los parámetros de la red con ponderación de 0.0005, la tasa de aprendizaje sigue una póliza polinomial con momentum 0.9, y los pesos de los dos primeros bloques de todas las arquitecturas convolucionales se mantienen fijos durante el entrenamiento. La tasa de aprendizaje se multiplica por un factor de 10 para las actualizaciones de los pesos de $\boldsymbol{\theta}_{\text{cls}}$, $\boldsymbol{\theta}_{\text{comp}}$, $\boldsymbol{W}_{\text{comp}}$, $\boldsymbol{b}_{\text{comp}}$, y los *embeddings* para objetos y atributos, todos los cuales se inicializan aleatoriamente. El peso para las clases relevantes en la función de pérdida se fija en $\lambda_{\text{fg}} = 1.0$, mientras que el asociado a categorías de fondo se fija en $\lambda_{\text{bg}} = 0.1$ para priorizar el entrenamiento de las clases relevantes. El peso asociado a compuestos categoría-atributo se fija en $\lambda_{\text{comp}} = 0.1$, para obtener magnitudes similares a las de la componente para categorías relevantes. Se muestrean $N^{-}_{\text{comp}} = 10$ categorías negativas por cada compuesto positivo, con el objetivo de priorizar la información de atributos en las predicciones para pares compuestos. Se aplican técnicas de aumentación artificial de datos estándar, consistentes en inversiones horizontales, escalamientos, y recortes aleatorios sobre las imágenes de entrada. Las dimensiones de los recortes de entrada se fijan en 321×321, como en [6, 8, 10].

Generación de Máscaras de Segmentación

En [9], el parámetro α de (4.15) se alterna entre 4 y 24 para obtener semillas de alta confianza para clases relevantes y el fondo, respectivamente, que son necesarias para entrenar su AffinityNet. En nuestro caso, al igual que en [17], se elige un valor intermedio fijo para obtener máscaras de segmentación completas a partir de los mapas de activación generados. Este valor se determina de forma independiente para cada arquitectura convolucional, utilizando simple búsqueda en grilla con paso 1 sobre el intervalo $[4, 24]$ [9], y evaluando la calidad de las máscaras generadas en *trainval*4k, según la metodología descrita en la Sección 4.5.2. Por simplicidad, para estos ajustes no se consideran atributos ni categorías de fondo. De esta forma, se determina un valor de 8 para ResNet-50, de 4 para VGG-16, y de 5 para ResNet-38. Los óptimos menores para las dos últimas arquitecturas pueden explicarse por el uso de convoluciones dilatadas, que tienden a extender las activaciones para categorías relevantes, aumentando naturalmente su *recall*. Por otra parte, el parámetro γ de (4.17) se determina de forma independiente siguiendo la misma estrategia de búsqueda en grilla, en este caso con paso 0.1 en un rango $[0.5, 1.1]$. Se obtiene un valor óptimo de 0.7 para las tres arquitecturas, que se utiliza en todos los experimentos reportados.

Para la etapa de dCRF se utiliza la implementación original de [46], mediante un *wrapper* público para Python[3]. Todos los parámetros se fijan en sus valores por defecto, que corresponden a $\sigma_\alpha = 80$, $\sigma_\beta = 13$, $\sigma_\gamma = 3$, $w_1 = w_2 = 1$, función de compatibilidad de Pott, y 10 iteraciones de optimización.

Modelo de Segmentación

El modelo de segmentación supervisado corresponde a DeepLab-ASPP [62], re-implementado en PyTorch siguiendo la versión original en Caffe[4], pero utilizando entradas de una sola escala, como en [7]. El modelo de segmentación supervisado se entrena durante 10 épocas con *minibatches* de tamaño 18. La tasa de aprendizaje basal se aumenta de 0.001 a 0.01, y se agregan 2k pasos de calentamiento al comienzo del entrenamiento, en el que la tasa de aprendizaje aumenta linealmente desde 0 hasta su valor basal. Esta técnica ha sido empleada en numerosos trabajos recientes [97, 98, 39], como una forma de prevenir la inestabilidad de los modelos al utilizar mayores tasas de aprendizaje, especialmente durante las primeras etapas del entrenamiento, en las que los pesos inicialmente aleatorios de la red pueden resultar en valores elevados para los gradientes. Se observa que estos cambios mejoran los resultados del modelo final, como se detalla en la Sección 5.5.3. Todos los demás hiperparámetros siguen la implementación original [62].

4.5.5 Stack Tecnológico Utilizado

Los modelos desarrollados se implementan en Python, utilizando PyTorch 1.4.0 con *backend* CUDA 10.1 y cuDNN 7.6. Los experimentos reportados se ejecutan en un servidor con tres tarjetas gráficas NVIDIA GeForce RTX 2080 Ti con 11 GB de memoria cada una, un procesador Intel Xeon W-2123 @ 3.60GHz, 64 GB de RAM, y sistema operativo Ubuntu 18.04.

La Tabla 4.3 muestra los tiempos promedio de entrenamiento bajo estas condiciones pa-

[3] `https://github.com/lucasb-eyer/pydensecrf`
[4] `https://bitbucket.org/aquariusjay/deeplab-public-ver2/`

ra las distintas redes neuronales utilizadas. La paralelización en múltiples GPUs se realiza siguiendo una estrategia de paralelización de datos, utilizando las funcionalidades de DistributedDataParallel de PyTorch[5]. Cabe destacar que todos los modelos, exceptuando la red de localización con arquitectura ResNet-38, pueden ser entrenados utilizando solo una de las tarjetas gráficas, sin necesidad de modificar los hiperparámetros del entrenamiento. La Tabla 4.3 indica también los tiempos de entrenamiento promedio obtenidos en estos casos. Adicionalmente, la etapa de generación de máscaras de segmentación artificiales sobre el conjunto de entrenamiento de MS-COCO toma en promedio 7.9 horas en el caso de la red de localización con arquitectura ResNet-50, 9.2 horas al usar VGG-16, y 14.8 horas con ResNet-38. En estos casos la base de datos se divide en tres subconjuntos iguales, que se procesan en paralelo utilizando tres réplicas de la red de localización, una en cada GPU.

Tabla 4.3: Tiempo promedio de entrenamiento en la base de datos MS-COCO para los distintos modelos que componen la metodología propuesta. Las GPUs corresponden a tarjetas NVIDIA GeForce RTX 2080 Ti.

Modelo	Tiempo promedio de entrenamiento	
	3 GPUs	1 GPU
Red de localización (ResNet-50)	8.6 horas	11.4 horas
Red de localización (VGG-16)	11.6 horas	14.8 horas
Red de localización (ResNet-38)	17.5 horas	-
DeepLab-V2 (VGG-16)	7.2 horas	9.8 horas

[5] `https://pytorch.org/tutorials/intermediate/ddp_tutorial.html`

Capítulo 5

Resultados

5.1 Evaluación de las Etiquetas de Clasificación

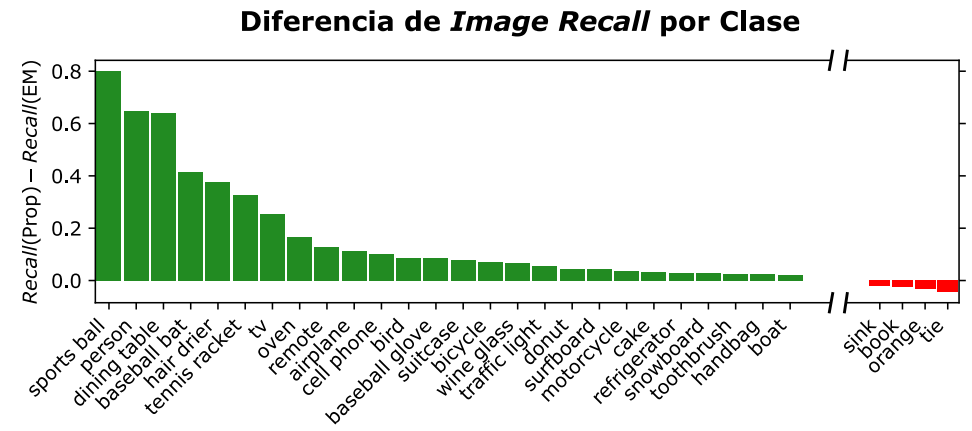
En primer lugar, se realiza una evaluación de las etiquetas de clasificación para categorías relevantes extraídas a partir de descripciones por el módulo de procesamiento de texto propuesto. En la Tabla 5.1 se resumen los resultados obtenidos, en términos de las métricas definidas en la Sección 2.3.2. Como referencia, se incluyen también los resultados obtenidos usando la estrategia de [17], que consiste en simplemente buscar menciones exactas de nombres de clases o su forma plural, lo que denominamos *exact match* (EM).

Tabla 5.1: Evaluación de las etiquetas de clasificación para categorías semánticas relevantes extraídas a partir de descripciones sobre el conjunto de entrenamiento de MS-COCO.

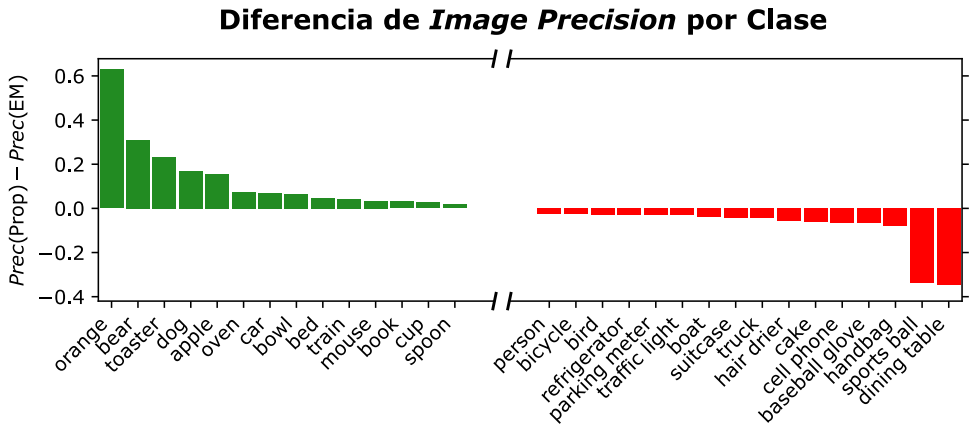
Método	*Image* IoU	*Image Precision*	*Image Recall*	*Pixel Recall*
EM	0.521	0.904	0.560	0.762
Propuesto	0.579	0.911	0.616	0.826

Se observa que la metodología propuesta permite mejorar la calidad de las etiquetas de clasificación generadas según todas las métricas utilizadas, especialmente en términos de *IoU* y *recall*. Por otra parte, se observa que para ambos métodos existe un margen de más de 20% absoluto entre los valores para *image* y *pixel recall*. Esto indica que las descripciones de las imágenes tienden a mencionar con mayor frecuencia a objetos más grandes, lo que coincide con la intuición. Las consecuencias de este sesgo se analizan en mayor profundidad en la Sección 5.3.

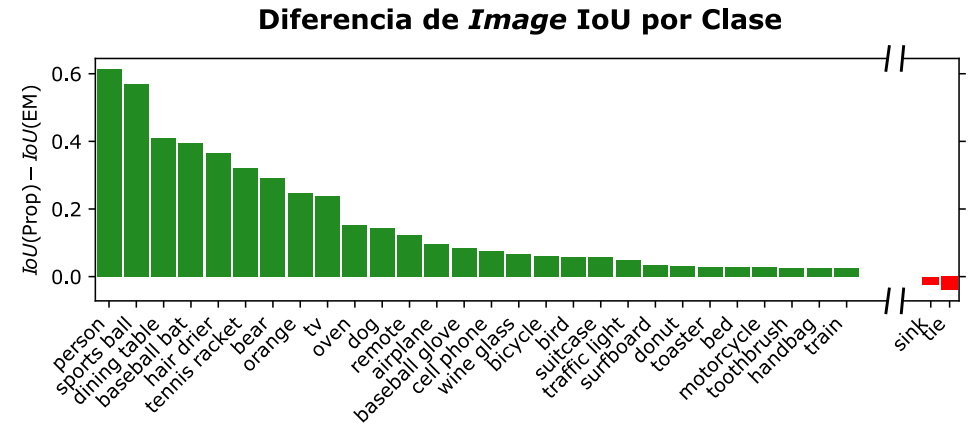
En la Fig. 5.1 se presentan en forma gráfica las diferencias entre ambos métodos en términos de *image recall*, *precision*, e IoU, divididos por categoría semántica. Para facilitar la visualización, se muestran solo las 30 categorías que presentan las diferencias de mayor valor absoluto. En primer lugar, se observa que el módulo de procesamiento de texto es particularmente útil para mejorar el *recall* de las etiquetas generadas, especialmente en el caso de categorías con gran cantidad de sinónimos e hipónimos comunes, tales como "*sports ball*"



(a)



(b)



(c)

Figura 5.1: Diferencia por clase entre las etiquetas de clasificación extraídas utilizando el método propuesto y las obtenidas con EM, en términos de (a) *image recall*, (b) *image precision*, y (c) *image* IoU.

(+80.0%) o "*person*" (+64.6%). Esto indica la importancia de modelar este tipo de relaciones para detectar categorías relevantes en descripciones de forma efectiva.

Por otra parte, existe solo un pequeño conjunto de categorías para las que el uso del módulo propuesto disminuye el *recall* con respecto a EM, y que en general corresponden a palabras que el modelo pre-entrenado de análisis sintáctico asigna incorrectamente a categorías gramaticales diferentes a sustantivos en algunos casos. Por ejemplo, el modelo puede identificar incorrectamente "*tie*", "*book*", o "*sink*" como verbos, u "*orange*" como adjetivo, en cuyo caso no se procesan como candidatos a objeto, y la etiqueta correspondiente se pierde. Estos problemas podrían aliviarse adoptando un modelo más robusto para la etapa de generación de árboles de dependencia.

En el caso de la precisión, el modelo aporta sobre todo para filtrar nombres de categorías que se utilizan con frecuencia para formar sustantivos compuestos que no pertenecen a la clase. Por ejemplo, como ocurre con "*bear*" en "*teddy bear*", "*toaster*" en "*toaster oven*", o "*dog*" en "*hot dog*". Adicionalmente, la etapa de análisis sintáctico sirve para descartar nombres de clases que se utilizan como atributos, como "*orange*" (el color), o "*apple*" (describiendo productos de esta marca).

Por otra parte, el uso del módulo propuesto puede reducir también la precisión de algunas categorías, lo que ocurre principalmente cuando el nombre de la clase posee sinónimos, hipónimos, u holónimos polisémicos, es decir, que pueden tener distintos significados dependiendo del contexto. Por ejemplo, dependiendo del contexto, "*football*" puede referirse a la pelota, en cuyo caso corresponde a un subtipo de la clase "*sports ball*", o al deporte. Si el modelo de WSD identifica erróneamente el primer significado, se reduce la precisión para esta categoría. En este caso, los problemas podrían aliviarse adoptando un mejor modelo para la etapa de *word sense disambiguation*. Dentro de estos casos se observan dos *outliers* con reducciones por sobre 30% absoluto de precisión. Sin embargo, ambos corresponden a categorías con nombres específicos y poco utilizados en descripciones, lo que resulta en una precisión cercana al 100% con EM, pero muy bajo *recall*. Ambas clases, "*sports ball*" y "*dining table*", se encuentran entre las categorías con mayor aumento de *image* IoU al utilizar el módulo propuesto.

Como se observa en el último gráfico de la Fig. 5.1, la estrategia propuesta permite aumentar el *image* IoU para casi todas las categorías de MS-COCO, la cuarta parte de estas con aumentos sobre 5% absoluto. Estos resultados demuestran la efectividad del módulo de procesamiento de texto para extraer etiquetas de clasificación para categorías relevantes a partir de descripciones, sin necesidad de datos adicionales de entrenamiento.

5.2 Análisis de las Componentes de Supervisión

En esta sección, se presenta un análisis del impacto de las distintas componentes de supervisión extraídas por el módulo de procesamiento de texto, aplicadas al problema de segmentación semántica con supervisión débil. Con este propósito, se utilizan distintas combinaciones de etiquetas visuales para entrenar la red de localización propuesta, y luego generar máscaras de segmentación semántica, siguiendo la metodología descrita en el Capítulo 4. Posteriormente, se utilizan las máscaras resultantes para entrenar el modelo de segmentación supervisado. Como referencia, se entrena además la red de localización utilizando solo etiquetas de categorías de objetos relevantes detectadas mediante EM. La Tabla 5.2 resume los resultados obtenidos, incluyendo la evaluación tanto de las máscaras de segmentación generadas para el conjunto de entrenamiento(*train*), como del modelo de segmentación final sobre el conjunto

de validación de MS-COCO (*val*).

Tabla 5.2: Efecto de las distintas componentes de supervisión sobre la calidad del modelo de segmentación semántica en la base de datos MS-COCO. Se indican los resultados tanto para las máscaras de segmentación generadas en el conjunto de entrenamiento (*train*), como los resultados del modelo supervisado sobre el conjunto de validación (*val*). Los *checkmarks* indican que la componente de supervisión correspondiente se utiliza durante el entrenamiento de la red de localización, y para generar las máscaras de segmentación de entrenamiento.

Clases Relevantes	Atributos	Clases de Fondo	*Prec.* (*train*)	*Recall* (*train*)	mIoU (*train*)	*Prec.* (*val*)	*Recall* (*val*)	mIoU (*val*)
EM	-	-	0.454	0.558	0.324	0.440	0.536	0.308
Propuesto	-	-	0.492	0.589	0.359	0.507	0.513	0.338
Propuesto	✓	-	0.491	0.611	0.368	0.505	0.524	0.344
Propuesto	-	✓	0.536	0.534	0.363	0.535	0.499	0.352
Propuesto	✓	✓	0.549	0.543	0.369	0.560	0.497	0.357

5.2.1 Categorías Relevantes

En primer lugar, se observa que la mejora en las etiquetas de clasificación para categorías relevantes tiene un impacto sustancial sobre la calidad de las máscaras de segmentación artificiales obtenidas, pasando de 0.324 a 0.359 mIoU en el conjunto de entrenamiento. Esto se traduce en una mejora similar en la calidad del modelo de segmentación final, que pasa de 0.308 a 0.338 mIoU en el conjunto de validación de MS-COCO.

Las categorías que presentan el mayor aumento de IoU en el conjunto de validación corresponden en general a las categorías cuyas etiquetas presentan el mayor aumento de *image* IoU al reemplazar EM por el módulo de procesamiento de texto. Por ejemplo, la categoría "*person*" mejora en 54.4% IoU, "*dining table*" mejora en 25.6% IoU, y "*orange*" mejora en 21.0% IoU. Estos resultados evidencian la efectividad de la metodología propuesta para extraer etiquetas de categorías semánticas relevantes para WSSS a partir de descripciones en lenguaje natural.

5.2.2 Atributos

Al introducir atributos a la supervisión del modelo, la calidad de las máscaras generadas llega a 0.368 mIoU, mejorando los resultados sobre el conjunto de validación a 0.344 mIoU. Se observa que las mejoras obtenidas se concentran principalmente en las categorías de objetos que cuentan con la mayor cantidad y variedad de atributos visuales asociados en la base de datos. Dentro de este grupo, la mayor parte presenta mejoras sobre todo en términos de *recall*, y corresponden a categorías de objetos típicamente grandes, y con múltiples partes distintivas. Por ejemplo, la categoría "*cow*" presenta una mejora de 4.2% IoU, "*train*" mejora en 3.1% IoU, y "*horse*" mejora en 2.5% IoU. Cualitativamente, se observa que en estos casos los mapas para pares compuestos ayudan principalmente a extender los mapas de activación por clase hacia regiones e instancias menos distinguibles de la misma categoría, guiados por atributos de bajo nivel.

Existen también algunas categorías que mejoran sobre todo en términos de precisión, y que corresponden usualmente a objetos pequeños y más difíciles de localizar, como por ejemplo "*frisbee*", que mejora en 4.7% IoU, o "*donut*", que mejora en 3.3% IoU. En estos casos, la información de atributos puede ayudar a corregir detecciones erróneas, cuando el modelo falla en localizar un objeto utilizando solo información de su categoría.

La Fig. 5.2 muestra un ejemplo en el que ocurren ambos tipos de efectos descritos anteriormente: mientras que el mapa de activación para la categoría "*cat*" localiza solo las partes más características de la cabeza del gato, los mapas para los compuestos ("*cat*", "*furry*") y ("*cat*", "*gray*"), que destacan mayoritariamente el pelaje del animal, permiten obtener una máscara final más completa. Por otra parte, al introducir información adicional asociada a un color ("*green*") el modelo es capaz de corregir la localización fallida para la categoría "*umbrella*", aprovechando esta información de bajo nivel.

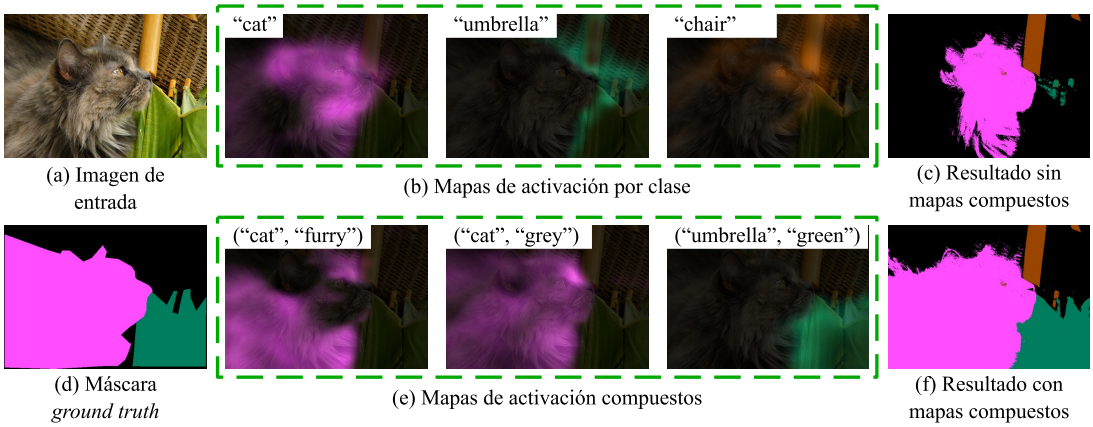


Figura 5.2: Ejemplo del efecto de los mapas de activación compuestos sobre la calidad de las máscaras de segmentación generadas. Los atributos ayudan a extender las activaciones por categoría hacia regiones e instancias menos características de la misma clase. Se muestra (a) la imagen de entrada, (b) los mapas de activación generados por la red de localización para cada categoría positiva, (c) la máscara de segmentación generada usando solo los mapas por categoría, (d) la máscara *ground truth*, (e) los mapas para pares compuestos del tipo categoría-atributo, y (f) la máscara generada usando todos los mapas de activación.

La Fig. 5.3 muestra diversos ejemplos adicionales de mapas compuestos, y sus efectos sobre las máscaras de segmentación generadas. Cualitativamente, se observa que el modelo aprende a localizar correctamente distintas partes de los objetos guiado por sus características visuales, incluso para atributos menos frecuentes, como "*furry*" o "*rusty*".

Las seis últimas filas de la Fig. 5.3 ilustran los principales modos de fallo de este tipo de mapas. En primer lugar, se observa que el modelo tiende a generar mapas nulos (sin activaciones positivas) en casos de combinaciones muy poco frecuentes, como en el caso de ("*cow*", "*ceramic*") en la fila (9), o para ("*sofa*", "*plastic*") en la fila (10). Sin embargo, estos casos no impactan negativamente las máscaras de segmentación generadas. Por otra parte, en ciertos casos el modelo puede destacar erróneamente objetos de categorías similares, guiado por sus características de bajo nivel, como se muestra en las últimas cuatro filas de la Fig. 5.3. Esto es consecuencia de priorizar la predicción de atributos por sobre la predicción

Figura 5.3: Visualización del efecto de los mapas de activación compuestos sobre las máscaras de segmentación generadas. En cada caso se muestra (a) la imagen de entrada, (b) ejemplos relevantes de mapas de activación generados por la red de localización, (c) la máscara de segmentación artificial generada utilizando solo los mapas de activación por clase, (d) la máscara obtenida utilizando todos los mapas de activación, y (e) la máscara *ground truth*.

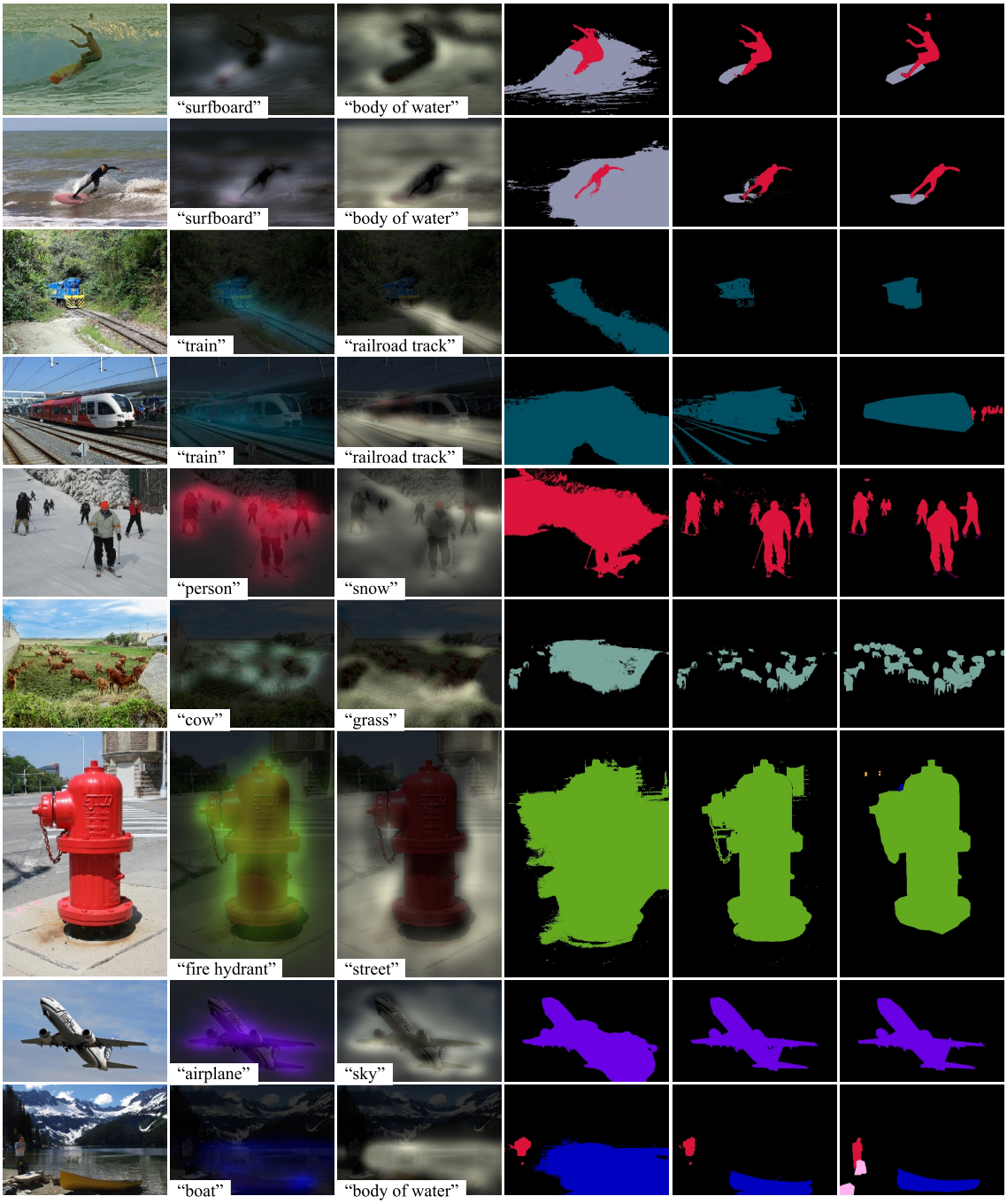
de categorías durante el entrenamiento de los *embeddings* compuestos. En casos en los que las regiones destacadas erróneamente por mapas compuestos pertenecen a categorías etiquetadas (tanto relevantes como de fondo), se observa que en general los mapas de categorías y el uso de dCRF son suficientes para corregir estos errores, como ocurre en los casos de las filas (11) y (12). Sin embargo, en casos en los que estas regiones corresponden a objetos no etiquetados, pueden introducirse pixeles indeseados a las máscaras de segmentación finales, como ocurre con el cartel en el caso de la fila (13), o con la toalla en el caso de la fila (14). En la práctica, se observa que la incorporación de mapas de activación para pares clase-atributo mejora el *recall* y mIoU de las máscaras generadas, sin impactar significativamente su precisión.

5.2.3 Categorías de Fondo

Se observa que la inclusión de mapas de activación para categorías de fondo ayuda a separar las regiones relevantes de su contexto, mejorando principalmente la precisión de las máscaras de segmentación obtenidas. Estos mapas de activación son particularmente útiles en casos en que múltiples instancias de una misma clase relevante aparecen cerca unas de otras, lo que normalmente deriva en que las activaciones de las distintas instancias se solapen entre sí, cubriendo también parte del fondo. Un efecto similar ocurre cuando las siluetas de los objetos aportan información relevante para la clasificación, como en el caso de "*fire hydrant*", lo que resulta en CAMs con altas activaciones en los bordes, que cubren también pixeles de fondo. En todos estos casos, la etapa de CRF tiende a expandir las regiones de fondo etiquetadas erróneamente como *foreground* hacia otros pixeles irrelevantes cercanos con colores similares. La incorporación de mapas de activación positivos para el fondo que compiten con los mapas para clases relevantes permite obtener predicciones que respetan de mejor manera los límites de bajo nivel de la imagen.

Adicionalmente, este tipo de mapas son útiles en casos en que ciertas categorías relevantes presentan una alta co-ocurrencia con determinadas categorías de objetos de fondo, tales como "*train*" con "*train track*", o "*surfboard*" con "*water*". En estos casos, los mapas de activación para las categorías relevantes tienden a incluir también los objetos de fondo, dado que estos aportan información útil para clasificación, pese a ser perjudiciales para segmentación. Al incluir la categoría de fondo explícitamente durante el entrenamiento, el modelo puede utilizar ejemplos en los que los objetos aparecen por separado para aprender a diferenciar ambas categorías. Esto resulta en mapas de activación superpuestos, pero distintos, que pueden ser combinados para mejorar la localización de las clases relevantes. La Fig. 5.4 muestra varios ejemplos de ambos tipos de efectos descritos anteriormente.

Pese a que la inclusión de los mapas para clases de fondo mejora principalmente la precisión de las máscaras de segmentación artificiales para el conjunto de entrenamiento, manteniendo su mIoU relativamente constante, los resultados en el conjunto de validación mejoran sustancialmente, de 0.338 a 0.352 mIoU. Esto podría deberse a que la eliminación sistemática de regiones erróneas en las máscaras de entrenamiento tiene un mayor efecto sobre la calidad del modelo final que la eliminación aleatoria de ciertas regiones relevantes. Las mejoras en IoU se concentran principalmente en categorías relevantes que presentan alta co-ocurrencia con determinadas categorías de fondo frecuentes. Por ejemplo, "*fire hydrant*" (+13.7% IoU) o "*stop sign*" (+7.9% IoU) con la clase de fondo "*street*", "*surfboard*" (+9.7% IoU) con "*water*", o "*airplane*" (+8.8% IoU) con "*sky*".



(a) Imagen de entrada | (b) Mapa de clase relevante | (c) Mapa de clase de fondo | (d) Resultado sin clase de fondo | (e) Resultado con clase de fondo | (f) *Ground truth*

Figura 5.4: Visualización del efecto de los mapas de activación para categorías de fondo sobre la calidad de las máscaras de segmentación generadas. En cada caso se muestra (a) la imagen de entrada, (b) el mapa de activación generado por la red de localización para una categoría relevante, (c) el mapa de activación generado para una categoría de fondo, (d) la máscara de segmentación artificial generada utilizando solo los mapas de activación para clases relevantes, (e) la máscara obtenida utilizando los mapas para clases relevantes y de fondo, y (f) la máscara *ground truth*.

Finalmente, combinando ambos tipos de información visual complementaria (atributos y clases de fondo), los resultados alcanzan 0.357 mIoU. En total, la utilización de los tres tipos de etiquetas generadas por el módulo de procesamiento de texto propuesto permite mejorar la calidad del modelo final de segmentación sobre el conjunto de validación de MS-COCO por un margen de 4.9% mIoU, comparado con el caso en que se utilizan solo etiquetas para categorías relevantes extraídas con EM. Esto equivale a un aumento relativo de 15.9%, lo que demuestra la efectividad del módulo propuesto para extraer información visual más completa y útil a partir de descripciones que la generada por métodos previos.

5.2.4 Comparación con Modelos de *Visual Grounding*

Los experimentos reportados en esta sección tienen por objetivo estudiar de mejor manera la importancia individual de cada uno de los módulos propuestos, además de evaluar el impacto de simplificar la supervisión de la red de localización, en términos de la calidad de los mapas de activación generados. Con este propósito, se realizan experimentos adicionales utilizando la supervisión generada por el módulo de procesamiento de texto para guiar la generación de máscaras de segmentación a partir de un modelo entrenado para localizar texto arbitrario. Concretamente, se utiliza el modelo pre-entrenado de [22], descrito en la Sección 3.4, que constituye el estado del arte en *visual grounding* basado en mapas de activación.

Dado que este modelo requiere de segmentos de texto para generar mapas de activación, se realizan experimentos preliminares utilizando los nombres de cada categoría como entrada para su rama textual. Los *embeddings* resultantes se proyectan sobre las representaciones codificadas de las imágenes correspondientes utilizando (3.17), lo que permite obtener mapas de localización por clase similares a CAMs. Estos mapas se adaptan para generar máscaras de segmentación semántica siguiendo la metodología propuesta en [17], que consiste en sustraer de cada mapa el promedio entre su máxima y mínima activación, fijar las activaciones para la clase fondo en 0, y seleccionar la categoría correspondiente al *argmax* para cada pixel. Utilizando EM para detectar categorías relevantes presentes en cada imagen, esta estrategia permite obtener 0.234 mIoU sobre el conjunto de *trainval*4k.

Posteriormente, se experimenta utilizando el procedimiento descrito en la Sección 4.3 para generar mapas de activación a partir de los *embeddings* visuales y textuales producidos por el modelo, así como para normalizar los mapas, y predecir activaciones para la clase de fondo. El parámetro α de (4.15) se ajusta en este caso siguiendo el mismo método que para la red de localización, resultando un óptimo de 4. Esta estrategia mejora el mIoU de las máscaras artificiales a 0.272 en el conjunto de *trainval*4k, que aumenta a 0.309 mIoU al incorporar una etapa de dCRF para refinar las predicciones de localización antes de tomar el *argmax*. En consecuencia, se utiliza esta configuración como base para todos los demás experimentos, en los que se evalúa la calidad de las máscaras de segmentación obtenidas al considerar diferentes entradas para la rama textual del modelo. Las máscaras generadas con cada configuración se utilizan para entrenar el modelo de segmentación DeepLab-ASPP, bajo las mismas condiciones que en los experimentos de la sección anterior. Los resultados obtenidos se resumen en la Tabla 5.3.

En primer lugar, se observa que utilizar las etiquetas para categorías relevantes predichas por el módulo de procesamiento de texto en lugar de las extraídas usando EM nuevamente

Tabla 5.3: Resultados de segmentación semántica sobre MS-COCO combinando el modelo de *visual grounding* de [22] con la supervisión extraída por el módulo de procesamiento de texto propuesto. Se indican los resultados tanto para las máscaras de segmentación generadas en el conjunto de entrenamiento (*train*), como los resultados del modelo supervisado sobre el conjunto de validación (*val*). Los *checkmarks* indican que la componente de supervisión correspondiente se utiliza durante la generación de las máscaras de entrenamiento.

Clases Relevantes	Clases de Fondo	Frases Completas	*Prec.* (*train*)	*Recall* (*train*)	mIoU (*train*)	*Prec.* (*val*)	*Recall* (*val*)	mIoU (*val*)
EM	-	-	0.441	0.563	0.304	0.436	0.539	0.307
Propuesto	-	-	0.429	0.602	0.321	0.460	0.536	0.321
Propuesto	-	✓	0.437	0.599	0.330	0.484	0.529	0.334
Propuesto	✓	✓	0.472	0.555	0.334	0.510	0.511	0.342

mejora significativamente los resultados del modelo final de segmentación sobre el conjunto de validación. En este caso, se obtiene una mejora de 0.307 a 0.321 mIoU, lo que corresponde a un aumento relativo de 4.6%. Posteriormente, se experimenta generando mapas de activación adicionales utilizando como entrada el texto original de las frases sustantivas asociadas a cada categoría relevante detectada por el módulo de procesamiento de texto. Estas frases pueden contener adjetivos y otros modificadores, además de sinónimos, hipónimos, u holónimos específicos de la categoría correspondiente. Los mapas resultantes se combinan con los mapas asociados a los nombres de las clases, siguiendo el método propuesto para incorporar mapas de activación compuestos (4.13). Al incorporar esta información, los resultados sobre el conjunto de validación mejoran a 0.334 mIoU.

Finalmente, se experimenta generando mapas para categorías de fondo detectadas por el módulo de procesamiento de texto, utilizando la misma estrategia que para las categorías relevantes. Las activaciones resultantes se incorporan a la predicción del mapa de fondo siguiendo la metodología descrita en la Sección 4.3, lo que permite obtener resultados finales sobre el conjunto de validación de 0.342 mIoU.

En total, la utilización de la supervisión generada por el módulo de procesamiento de texto permite mejorar la calidad del modelo final de segmentación por un margen de 3.5% mIoU en comparación con EM, lo que equivale a un aumento relativo de 11.4%, pese a que las máscaras son generadas utilizando el mismo modelo pre-entrenado. Esto demuestran la efectividad de la metodología propuesta para extraer información visual útil para WSSS a partir de descripciones, así como de la estrategia utilizada para combinar los mapas de localización asociados a distintos tipos de etiquetas.

Adicionalmente, se observa que este resultado final es de todas maneras significativamente peor que el 0.357 mIoU obtenido con la red de localización propuesta. Esto es a pesar de que el modelo de *visual grounding* utilizado posee una arquitectura convolucional ResNet-152 mucho más profunda que la ResNet-50 del modelo propuesto, y es significativamente más costoso de entrenar debido al uso de redes neuronales recurrentes y *hard mining contrastive loss*. Estos resultados evidencian la efectividad de focalizar la supervisión del modelo de localización en información visual relevante para obtener mapas de activación de alta calidad.

5.3 Comparación con Etiquetas de Clasificación *Ground Truth*

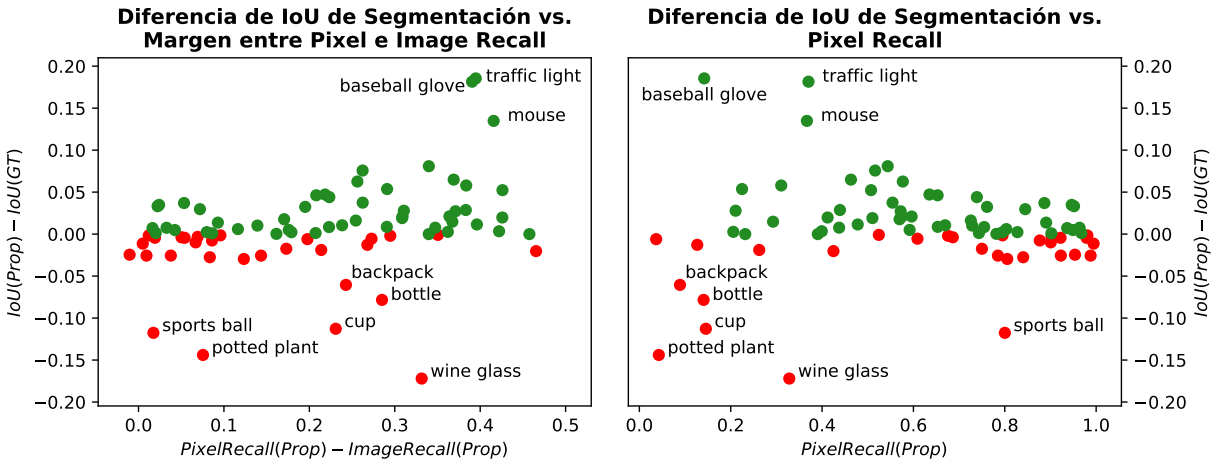
En esta sección se realiza una comparación entre las etiquetas de clasificación utilizadas normalmente en aplicaciones de WSSS, y la supervisión estructurada extraída a partir de descripciones por el módulo de procesamiento de texto propuesto, en términos de su calidad como fuente de información para entrenar modelos de segmentación semántica. Para esto, se experimenta entrenando la red de localización propuesta utilizando solo las etiquetas de clasificación *ground truth* de MS-COCO como supervisión. Posteriormente se sigue el mismo procedimiento que en los experimentos anteriores para generar máscaras de segmentación, y luego para entrenar el modelo supervisado. De esta manera, se obtiene un valor final de 0.330 mIoU sobre el conjunto de validación, tal como se muestra en la Tabla 5.4.

Tabla 5.4: Comparación entre la supervisión extraída a partir de descripciones y las etiquetas de clasificación *ground truth* como fuentes de información para la tarea de segmentación semántica con supervisión débil en MS-COCO. Se indican los resultados tanto para las máscaras de segmentación generadas en el conjunto de entrenamiento (*train*), como los resultados del modelo supervisado sobre el conjunto de validación (*val*). Los *checkmarks* indican que la componente de supervisión correspondiente se utiliza durante el entrenamiento de la red de localización, y para generar las máscaras de segmentación de entrenamiento.

Clases Relevantes	Atributos	Clases de Fondo	*Prec.* (*train*)	*Recall* (*train*)	mIoU (*train*)	*Prec.* (*val*)	*Recall* (*val*)	mIoU (*val*)
GT	-	-	0.429	0.674	0.349	0.448	0.536	0.330
EM	-	-	0.454	0.558	0.324	0.440	0.536	0.308
Propuesto	-	-	0.492	0.589	0.359	0.507	0.513	0.338
Propuesto	✓	✓	0.549	0.543	0.369	0.560	0.497	0.357

Estos resultados superan los obtenidos utilizando EM por un margen de 2.2% mIoU, pero sorprendentemente se encuentran por debajo de los obtenidos al utilizar solo las etiquetas para categorías relevantes generadas por el módulo de procesamiento de texto. Esto es a pesar de los valores relativamente bajos de *recall* que presentan las etiquetas extraídas a partir de descripciones, como se discutió en la Sección 5.1. Estos resultados podrían deberse a que las categorías no mencionadas en las descripciones tienden a corresponder a objetos más difíciles de identificar, tales como en casos de objetos pequeños o parcialmente ocluidos. En estos casos, si el modelo falla en localizar una determinada categoría, su etiqueta pasa a comportarse esencialmente como un falso positivo, introduciendo regiones erróneas a las máscaras de segmentación generadas, lo que degrada la calidad del modelo final.

Para estudiar en mayor profundidad los efectos de ambos tipos de supervisión, en la Fig. 5.5 se grafican las 80 categorías de MS-COCO como puntos en dos planos. En ambos casos, se utiliza como coordenada vertical la diferencia absoluta en IoU de segmentación al utilizar las etiquetas generadas a partir de descripciones en lugar del *ground truth*. Se observa en ambos gráficos que la mayoría de las categorías semánticas alcanzan resultados similares para ambos tipos de etiquetas, con un 80% de las clases dentro de un rango de $\pm 5\%$ IoU.



(a) (b)

Figura 5.5: Visualización de la diferencia en IoU de segmentación por clase al utilizar las etiquetas generadas por el módulo de procesamiento de texto en lugar del *ground truth*, (a) en función del margen entre *pixel* e *image recall*, y (b) en función del *pixel recall*.

Para visualizar el efecto del sesgo por tamaño en las etiquetas extraídas a partir de descripciones, en el primer gráfico se utiliza como coordenada horizontal la diferencia entre *pixel* e *image recall* de cada categoría, según se definen en la Sección 2.3.2. Un valor alto en este eje indica que existe una gran cantidad de falsos negativos para esta categoría (i.e., bajo *image recall*), pero que representan una proporción pequeña del total de pixeles etiquetados con esta clase (manteniendo un alto *pixel recall*). En otras palabras, que los falsos negativos derivados de detectar etiquetas a partir de descripciones corresponden mayoritariamente a instancias pequeñas de la clase.

Se observa que la mayoría de las categorías que mejoran su IoU al utilizar descripciones se encuentran por sobre el 20% de diferencia absoluta entre *pixel* e *image recall*. En particular, las 3 categorías que presentan el mayor aumento de IoU, "*mouse*", "*baseball glove*", y "*traffic light*", se encuentran todas en torno al 40% de margen. Esto indica que la eliminación de instancias pequeñas difíciles de identificar es un factor relevante para explicar las mejoras en los resultados de segmentación semántica al detectar categorías de objetos relevantes a partir de descripciones.

Por otra parte, para caracterizar de mejor manera los casos en los que utilizar descripciones perjudica los resultados, en el segundo gráfico se utiliza como coordenada horizontal el *pixel recall* absoluto de cada categoría. Se observa que esta métrica varía significativamente dependiendo de la categoría, lo que indica que ciertas clases de objetos son mencionadas con más frecuencia por los anotadores, incluso tomando en consideración su tamaño en la imagen. En particular, casi todas las categorías con disminuciones mayores al 5% de IoU se encuentran entre las clases con menor *pixel recall*, correspondiendo a objetos que ocupan típicamente un rol accesorio en las imágenes, y a consecuencia son muchas veces omitidos de las descripciones. Ejemplos de esto incluyen "*bottle*", "*cup*", "*wine glass*", o "*backpack*".

Una excepción a lo anterior corresponde a la categoría "*sports ball*", cuya reducción en IoU se debe principalmente a la baja precisión de las etiquetas predichas. Esto resulta de predicciones erróneas del modelo de WSD para palabras como "*football*" o "*baseball*", y también por casos en que se mencionan objetos de esta clase sin que aparezcan en la imagen. Por ejemplo, una imagen con descripción "*a tennis player preparing to hit the ball*" puede o no contener la pelota dentro del cuadro, pero el módulo de procesamiento de texto carece de contexto para resolver esta ambigüedad.

Además de los casos ya discutidos anteriormente, la categoría que presenta la mayor disminución en IoU corresponde a "*potted plant*" lo que se explica principalmente por su bajo nivel de *recall*. Esto se debe a que esta clase se menciona con muy poca frecuencia por su nombre, y no presenta un *synset* correspondiente en WordNet. Esta clase es única en el sentido de que corresponde en realidad a la unión de dos categorías semánticas distintas, "*vase*" y "*plant*", la primera de las cuales existe como categoría independiente en MS-COCO.

Una posible extensión al módulo de procesamiento de texto que permitiría manejar este tipo de casos corresponde a identificar patrones en el grafo de escena generado antes del proceso de filtrado, siguiendo un esquema similar al de [24] para identificar estructuras útiles en el árbol de dependencia. Por ejemplo, para predecir la clase "*potted plant*" podría identificarse la presencia de un objeto compatible con "*vase*" o "*pot*", y otro compatible con "*plant*", aprovechando las mismas relaciones de sinonimia, hiponimia, y holonimia dadas por WordNet, y luego verificar que exista una relación compatible entre ambos. En este caso, la categoría podría predecirse a partir de tripletas de objeto-relación-objeto como ("*pot*", "*with*", "*plant*"), o ("*flower*", "*in*", "*vase*").

Pese a las limitaciones anteriores, las diferencias entre ambos tipos de supervisión resultan en un neto positivo al utilizar etiquetas para categorías relevantes extraídas a partir de descripciones. Esto indica que la información sobre el tamaño y relevancia de las distintas categorías, que está implícita en las descripciones en lenguaje natural, resulta beneficioso para WSSS, compensando su carácter no exhaustivo. Más aún, al aprovechar la información complementaria presente en este tipo de anotaciones en forma de atributos y categorías de fondo, es posible extender este margen, llegando a 2.7% mIoU. Esto demuestra el potencial de las descripciones en lenguaje natural como fuente de información para supervisar modelos de segmentación semántica.

5.4 Experimentos de Transferencia de Aprendizaje

Para complementar los resultados obtenidos con la base de datos MS-COCO que se presentan en las secciones anteriores, se realizan experimentos adicionales utilizando las bases de datos PASCAL VOC y YouTube-Objects. Debido a que estas bases de datos no cuentan con descripciones en lenguaje natural, que son necesarias para el entrenamiento del modelo propuesto, estos experimentos se ejecutan siguiendo un esquema de transferencia de aprendizaje. Más precisamente, se utiliza el conjunto *train*2014 de MS-COCO para entrenar el método propuesto, como en secciones anteriores, y posteriormente se realiza la evaluación sobre el conjunto de validación de PASCAL VOC, y sobre el conjunto de cuadros etiquetados de YouTube-Objects. Para adaptar el modelo de segmentación a las nuevas bases de datos, se descartan todos los pesos de la capa final de segmentación que no corresponden

a ninguna de las categorías semánticas del dominio de destino, sin ejecutar ninguna etapa de re-entrenamiento adicional. No fue posible encontrar estudios anteriores que reportasen resultados para WSSS bajo este esquema de transferencia de aprendizaje, por lo que se entrenaron diversas *baselines* basadas en el modelo propuesto para servir de referencia, como se reporta en la Tabla 5.5.

Tabla 5.5: Resultados para los experimentos de transferencia de aprendizaje, en términos de mIoU sobre el conjunto de validación de PASCAL VOC, y sobre el conjunto de cuadros etiquetados de YouTube-Objects. Todos los modelos son entrenados en el conjunto de entrenamiento de MS-COCO. Los *checkmarks* indican que la componente de supervisión correspondiente se utiliza durante el entrenamiento de la red de localización, y para generar las máscaras de segmentación de entrenamiento.

Clases Relevantes	Atributos	Clases de Fondo	mIoU	
			PASCAL VOC	YouTube-Objects
GT	-	-	0.503	0.517
EM	-	-	0.448	0.528
Propuesto	-	-	0.512	0.538
Propuesto	✓	-	0.524	0.550
Propuesto	-	✓	0.533	0.584
Propuesto	✓	✓	0.548	0.597

5.4.1 PASCAL VOC

Se observa que todos los modelos alcanzan resultados significativamente mejores en PASCAL VOC que en el conjunto de validación de MS-COCO. Esto se deba a que PASCAL VOC define una menor cantidad de categorías, mientras que los objetos relevantes son en general de mayor tamaño, y más fáciles de identificar que los de MS-COCO. Como se observa en la Tabla 5.5, se obtienen mejoras similares para las distintas componentes del método propuesto a las observadas en MS-COCO, lo que refuerza las conclusiones derivadas de los experimentos reportados en secciones anteriores. Concretamente, se observa que utilizar el módulo de procesamiento de texto para identificar categorías relevantes en lugar de EM, mejora los resultados de forma sustancial, en este caso de 0.448 mIoU a 0.512 mIoU. El modelo entrenado con las etiquetas para categorías relevantes detectadas con el método propuesto también obtiene mejores resultados que modelo entrenado con las etiquetas de clasificación *ground truth*, que en este caso alcanza 0.503 mIoU. Adicionalmente, la incorporación tanto de atributos como de categorías complementarias al entrenamiento mejoran los resultados también bajo este esquema, alcanzándose 0.548 mIoU en el conjunto de validación de PASCAL VOC con la versión completa del método propuesto.

5.4.2 YouTube-Objects

En el caso de la base de datos YouTube-Objects, la utilización de categorías relevantes detectadas usando el método propuesto también mejora los resultados respecto de EM, si bien por un margen menor de 1.0% mIoU. Esto se debe a que las 10 categorías semánticas representadas en esta base de datos poseen nombres sin ambigüedades, que generalmente son

detectados correctamente con el método simple de EM. La utilización de información complementaria extraída con el módulo de procesamiento de texto mejora los resultados de forma sustancial, en este caso derivando en un aumento relativo en mIoU de 11.0%. Estos resultados son además 15.5% mejores que los obtenidos utilizando las etiquetas *ground truth*, lo que nuevamente demuestra el potencial de utilizar descripciones como fuente de supervisión para WSSS, y del modelo propuesto para aprovechar esta información.

Finalmente, los resultados obtenidos bajo este esquema de transferencia de aprendizaje muestran que el método propuesto también mejora la capacidad de generalización del modelo de segmentación final hacia nuevas bases de datos.

5.5 Experimentos Complementarios

En esta sección se presentan los resultados de experimentos complementarios que validan diversas decisiones de diseño adoptadas durante el desarrollo del método propuesto.

5.5.1 Importancia de la Selección de Atributos

En primer lugar, se evalúa la importancia de restringir el conjunto de atributos utilizados durante el entrenamiento de la red de localización a conceptos que denotan información visual de bajo nivel, tales como colores, materiales, o texturas, resumidos en la Tabla 4.2. Con este propósito, se experimenta entrenando la red de localización utilizando categorías relevantes, y distintos conjuntos de atributos visuales extraídos por el módulo de procesamiento de texto, sin el filtrado manual descrito anteriormente. En este caso se omiten las categorías de fondo, por simplicidad.

Tabla 5.6: Efecto de incorporar distintos conjuntos de atributos sin filtrar a la supervisión de la red de localización, en términos del mIoU de las máscaras generadas sobre el conjunto de entrenamiento de MS-COCO.

Tipo de atributos:	Ninguno	Solo de bajo nivel	Todos los adjetivos	Todos los verbos	Todos los sustantivos
mIoU (*train*):	0.359	0.368	0.355	0.356	0.357

La Tabla 5.6 muestra los resultados obtenidos en términos del mIoU de las máscaras de segmentación artificiales generadas sobre el conjunto de entrenamiento, con distintos tipos de atributos divididos solo según su categoría gramatical. Se observa que en todos estos casos, la incorporación de atributos resulta perjudicial para el desempeño del modelo. Esto se debe a que estos atributos describen mayoritariamente información distribuida, tales como cantidades o acciones, que el modelo es incapaz de localizar correctamente, degradando la calidad de las máscaras generadas.

5.5.2 Importancia de la Función de Pérdida Balanceada

Se estudia también el efecto de los pesos introducidos para balancear el aporte de ejemplos positivos y negativos para cada clase en las funciones de pérdida de la red de localización.

Para esto, se experimenta re-entrenando el modelo propuesto utilizando entropía cruzada binaria usual, equivalente a fijar $w_c^+ = w_c^- = 1$ en (4.9) y $\tilde{w}_a^+ = \tilde{w}_a^- = 1$ en (4.10). Los resultados obtenidos se presentan en la Tabla 5.7, observándose una caída de 0.6% mIoU al eliminar estos términos de la función de pérdida, que se concentra sobre todo dentro de un pequeño conjunto de categorías poco frecuentes. Cualitativamente, se observa también que estos pesos son necesarios para que el modelo aprenda a localizar correctamente pares compuestos con atributos poco frecuentes, si bien esto tiene poco impacto sobre el desempeño del modelo de segmentación final.

Tabla 5.7: Efecto de los pesos que balancean la contribución de ejemplos positivos y negativos en la función de pérdida de la red de localización sobre la calidad de segmentación en el conjunto de validación de MS-COCO. Los *checkmarks* indican que la componente de supervisión correspondiente se utiliza durante el entrenamiento de la red de localización, y para generar las máscaras de segmentación de entrenamiento.

Función de Pérdida	Clases Relevantes	Atributos	Clases de Fondo	*Prec.* (*val*)	*Recall* (*val*)	mIoU (*val*)
BCE estándar	Propuesto	✓	✓	0.586	0.466	0.351
BCE balanceada	Propuesto	✓	✓	0.560	0.497	0.357

5.5.3 Importancia de la Tasa de Aprendizaje del Modelo de Segmentación

Para todos los experimentos reportados anteriormente se utiliza una tasa de aprendizaje base de 0.01 para el entrenamiento del modelo supervisado, con 2000 iteraciones de calentamiento inicial, durante el cual la tasa de aprendizaje aumenta linealmente desde 0 hasta su valor nominal base, similar a [97]. Para evaluar la importancia de esta modificación, se re-entrena el modelo supervisado utilizando las mismas máscaras de segmentación artificiales, pero fijando la tasa de aprendizaje inicial en 0.001, y sin etapa de calentamiento, siguiendo la implementación original [62]. Los resultados obtenidos se resumen en la Tabla 5.8.

Tabla 5.8: Efecto de la tasa de aprendizaje y etapa de calentamiento sobre el entrenamiento del modelo de segmentación supervisado, evaluado en el conjunto de validación de MS-COCO. Los *checkmarks* indican que la componente de supervisión correspondiente se utiliza durante el entrenamiento de la red de localización, y para generar las máscaras de segmentación de entrenamiento.

Clases Relevantes	Atributos	Clases de Fondo	Tasa de Aprendizaje	Calentamiento	*Prec.* (*val*)	*Recall* (*val*)	mIoU (*val*)
Propuesto	✓	✓	0.001	-	0.526	0.470	0.326
Propuesto	✓	✓	0.001	2k pasos	0.525	0.468	0.328
Propuesto	✓	✓	0.01	2k pasos	0.560	0.497	0.357

Se observa que los valores modificados permiten mejorar de forma significativa el desempeño del modelo final con respecto a la configuración original. La etapa de calentamiento por

sí misma tiene un efecto marginal sobre los resultados, pero su inclusión evita que el modelo diverja durante las etapas iniciales del entrenamiento, en las que los pesos aleatorios de la capa de clasificación resultan en valores altos para la función de pérdida. Estos resultados demuestran que esta modificación constituye una forma simple y de bajo costo para mejorar la calidad del modelo entrenado con la supervisión artificial generada.

5.6 Comparación con el Estado del Arte

En esta sección se realiza una comparación entre los resultados obtenidos utilizando la metodología propuesta, y los reportados por otros estudios previos en segmentación semántica con supervisión débil propuestos en la literatura. La Tabla 5.9 resume los resultados de todos los métodos encontrados durante la revisión bibliográfica realizada que reportan resultados sobre la base de datos MS-COCO. Siguiendo la práctica estándar, la comparación se realiza en términos del mIoU sobre el conjunto de validación *val*2014. Para facilitar la comparación, en el caso del modelo propuesto se reportan los resultados obtenidos utilizando 3 arquitecturas convolucionales distintas para la implementación de la red de localización, correspondientes a VGG-16, ResNet-50, y ResNet-38. Cabe destacar que en los tres casos la arquitectura convolucional del modelo de segmentación supervisado se mantiene fija, correspondiendo a VGG-16, al igual que en los demás trabajos citados.

Para obtener una evaluación más completa del método propuesto, se realizan múltiples entrenamientos considerando diferentes inicializaciones aleatorias de los parámetros de las redes neuronales utilizadas, y se reportan tanto el valor de mIoU promedio como la desviación estándar obtenidos con cada arquitectura en la Tabla 5.9. En el caso de la red de localización basada en ResNet-50, se realizan 10 repeticiones con inicializaciones aleatorias de los parámetros, mientras que para las arquitecturas VGG-16 y ResNet-38 se realizan 5 repeticiones para cada una, debido al mayor costo computacional de entrenar estos modelos. Debido a que los estudios anteriores reportan resultados para un único entrenamiento de los modelos respectivos, se incluye solo este valor en las comparaciones de la Tabla 5.9. Los resultados obtenidos indican que el método propuesto es robusto frente a distintas inicializaciones aleatorias de sus parámetros, presentando desviaciones estándar en torno a 0.15% mIoU para las tres arquitecturas convolucionales estudiadas.

Entre los resultados de la Tabla 5.9, los más directamente comparables a los del modelo propuesto corresponden a los reportados para TAM-Net [17], ya que este es el único método anterior que aborda el problema de WSSS usando solo descripciones en lenguaje natural. En primer lugar, se observa que el desempeño del modelo propuesto basado en la arquitectura VGG-16 supera el reportado para la versión de TAM-Net basada en la misma arquitectura convolucional, por un margen de 5.3% mIoU. Esto es incluso sin utilizar Deep Seeded Region Growing (DSRG) [7], que es una técnica que permite refinar progresivamente las máscaras de segmentación artificiales mediante un costoso esquema de re-entrenamientos iterativos del modelo de segmentación supervisado.

Más importante aún, la versión del modelo propuesto basado en ResNet-38 supera ampliamente el estado del arte para este problema por un margen de 7.8% mIoU, mejorando de 28.5% a 36.3% mIoU en el conjunto de validación de MS-COCO. Adicionalmente, la versión basada en la significativamente más liviana ResNet-50 alcanza un mIoU muy cercano al ob-

Tabla 5.9: Comparación de métodos del estado del arte en segmentación semántica con supervisión débil, en términos de mIoU (%) sobre el conjunto de validación de MS-COCO. El mejor resultado se muestra en negrita.

Método	Arquitectura	Supervisión a Nivel de Imagen	Supervisión Adicional	mIoU [%]
DSRG [7]	VGG-16	etiquetas	saliency[a]	26.0
SGAN [79]	VGG-16	etiquetas	saliency	33.6
BFBP [99]	VGG-16	etiquetas	-	20.4
SEC [5]	VGG-16	etiquetas	-	22.4
WAILS [94]	VGG-16	etiquetas	-	22.5
IAL [11]	VGG-16	etiquetas	-	27.7
TAM-Net [17]	VGG-16	descripciones	-	21.6
TAM-Net [17] + DSRG [7]	VGG-16	descripciones	saliency[a]	26.9
TAM-Net [17]	ResNet-38	descripciones	-	28.5
TAM-Net [17] + DSRG [7]	ResNet-38	descripciones	saliency[a]	27.7
Propuesto	VGG-16	descripciones	-	32.19 ± 0.17
Propuesto	ResNet-50	descripciones	-	35.65 ± 0.13
Propuesto	ResNet-38	descripciones	-	$\mathbf{36.28 \pm 0.14}$

[a]Los *saliency maps* se utilizan solo para generar semillas para el fondo.

tenido con ResNet-38, permitiendo un buen balance entre tamaño del modelo y calidad de segmentación. Estos resultados demuestran la efectividad de la metodología propuesta para entrenar modelos de segmentación semántica a partir de descripciones en lenguaje natural.

Para realizar una evaluación más completa del modelo propuesto, se incluyen además otros métodos del estado del arte para segmentación semántica basados en otros tipos de supervisión débil. En particular, se observa que la versión del modelo propuesto basada en VGG-16 supera a todos los trabajos previos con la misma arquitectura que utilizan únicamente etiquetas de clasificación, validando aún más la efectividad de la metodología desarrollada. Estos resultados demuestran además el potencial de utilizar descripciones en lenguaje natural como fuentes de supervisión para modelos de WSSS.

Finalmente, se observa que el método propuesto obtiene incluso resultados comparables a los reportados para SGAN [79]. Esto es a pesar de que este modelo utiliza extensivamente una supervisión mucho más fuerte, que consiste en mapas de prominencia o *saliency maps* agnósticos de clase, generados por un modelo entrenado con supervisión a nivel de pixel. Pese a que el método propuesto depende solo de información a nivel de imagen, es capaz de alcanzar un 95.8% del mIoU reportado para SGAN utilizando la misma arquitectura, y lo supera ampliamente al utilizar la versión de la red de localización basada en ResNet-50.

Para completar el análisis de los resultados obtenidos, en la Tabla 5.10 se presenta el detalle

de la IoU obtenida para cada una de las 80 categorías de MS-COCO y el fondo, utilizando la mejor versión del modelo propuesto, basada en ResNet-38, además de los obtenidos con VGG-16, para facilitar la comparación con otros trabajos anteriores. En ambos casos se presentan tanto el valor promedio como la desviación estándar para los 5 entrenamientos realizados. Lamentablemente, en [17] no se reportan los resultados por categoría, por lo que no es posible hacer una comparación a este nivel. A modo de referencia, se incluyen los valores reportados en [7] para BFBP [99], SEC [5], y DSRG [7], si bien estos métodos utilizan etiquetas de clasificación como supervisión.

Tabla 5.10: Resultados de segmentación por categoría, en términos de IoU (%) sobre el conjunto de validación de MS-COCO, obtenidos con el modelo propuesto y con métodos previos publicados en la literatura.

Categoría	BFBP	SEC	DSRG	Propuesto (VGG16)	Propuesto (ResNet38)	Categoría	BFBP	SEC	DSRG	Propuesto (VGG16)	Propuesto (ResNet38)
background	68.8	74.3	80.6	78.6 ± 0.1	80.4 ± 0.1	wine glass	17.5	22.3	24.0	2.7 ± 0.6	16.9 ± 3.9
person	27.5	43.6	-	60.9 ± 0.3	61.2 ± 0.4	cup	5.6	17.9	20.4	9.3 ± 0.9	11.6 ± 0.5
bicycle	18.2	24.2	30.4	36.9 ± 1.1	37.6 ± 0.6	fork	0.5	1.8	0.0	11.8 ± 1.2	11.4 ± 0.8
car	7.2	15.9	22.1	29.0 ± 0.8	33.2 ± 0.5	knife	1.0	1.4	5.0	8.1 ± 0.3	9.6 ± 0.2
motorcycle	40.5	52.1	54.2	59.1 ± 0.3	59.6 ± 0.1	spoon	0.6	0.6	0.5	1.9 ± 0.3	1.5 ± 0.4
airplane	32.0	36.6	45.2	50.6 ± 1.2	56.0 ± 0.7	bowl	13.3	12.5	18.8	17.0 ± 0.5	20.9 ± 0.8
bus	39.2	37.7	38.7	58.9 ± 1.2	64.2 ± 0.1	banana	44.9	43.6	46.4	58.0 ± 0.7	58.6 ± 0.3
train	26.5	30.1	33.2	35.4 ± 0.9	51.9 ± 0.4	apple	18.9	23.6	24.3	43.1 ± 0.9	43.6 ± 0.2
truck	17.5	24.1	25.9	32.7 ± 1.3	39.5 ± 0.3	sandwich	21.4	22.8	24.5	34.2 ± 0.5	36.8 ± 0.4
boat	16.5	17.3	20.6	31.7 ± 0.7	35.6 ± 0.3	orange	35.0	44.3	41.2	55.6 ± 0.5	57.6 ± 0.6
traffic light	3.9	16.7	16.2	33.8 ± 0.9	37.2 ± 0.5	broccoli	27.0	36.8	35.7	53.5 ± 0.5	53.6 ± 0.2
fire hydrant	33.1	55.9	60.4	64.4 ± 0.4	65.3 ± 0.3	carrot	16.0	6.7	15.3	30.1 ± 0.6	30.9 ± 0.4
stop sign	28.4	48.4	51.0	72.9 ± 0.7	72.1 ± 1.4	hot dog	22.5	31.2	24.9	37.1 ± 1.7	44.1 ± 0.1
parking meter	25.5	25.2	26.3	50.4 ± 0.7	55.4 ± 0.8	pizza	57.8	50.9	56.2	53.0 ± 1.3	63.7 ± 0.5
bench	12.4	16.4	22.3	29.5 ± 0.4	32.8 ± 0.2	donut	36.2	32.8	34.2	26.0 ± 1.8	45.6 ± 1.7
bird	31.1	34.7	41.5	41.2 ± 1.0	46.9 ± 0.4	cake	17.0	12.0	6.9	34.8 ± 0.4	39.6 ± 0.6
cat	52.8	57.2	62.2	64.6 ± 0.9	67.1 ± 0.5	chair	8.2	7.8	9.7	14.2 ± 0.3	15.4 ± 0.7
dog	44.1	45.2	55.6	46.1 ± 1.8	57.4 ± 0.2	couch	13.9	5.6	17.7	26.5 ± 0.3	28.7 ± 0.3
horse	34.2	34.4	42.3	50.8 ± 0.5	53.4 ± 0.2	potted plant	7.4	6.2	14.3	0.2 ± 0.1	0.7 ± 0.1
sheep	38.0	40.3	47.1	52.5 ± 0.9	55.7 ± 0.4	bed	29.8	23.4	32.4	42.8 ± 0.5	45.4 ± 0.3
cow	42.1	41.4	49.3	50.1 ± 1.0	55.2 ± 0.2	dining table	2.0	0.0	3.8	21.4 ± 0.7	24.0 ± 0.8
elephant	65.2	62.9	67.1	69.0 ± 0.5	71.6 ± 0.3	toilet	30.1	38.5	43.6	48.8 ± 1.1	49.8 ± 1.9
bear	57.0	59.1	62.6	59.4 ± 1.6	64.8 ± 0.3	tv	14.8	19.2	25.3	33.5 ± 1.4	39.2 ± 0.2
zebra	65.0	59.8	63.2	75.0 ± 1.0	74.3 ± 0.3	laptop	19.9	20.1	21.1	43.0 ± 0.7	46.6 ± 0.1
giraffe	55.6	48.8	54.3	61.5 ± 1.3	59.6 ± 0.7	mouse	0.4	3.5	0.9	15.0 ± 1.3	15.5 ± 0.8
backpack	3.2	0.3	0.2	4.6 ± 0.7	6.3 ± 0.8	remote	9.9	17.5	20.6	18.7 ± 1.8	31.3 ± 1.2
umbrella	28.1	26.0	35.3	44.8 ± 1.1	50.8 ± 0.5	keyboard	19.9	12.5	12.3	42.1 ± 0.9	46.7 ± 0.6
handbag	1.1	0.5	0.7	2.6 ± 0.3	4.8 ± 0.4	cellphone	26.1	32.1	33.0	42.2 ± 0.8	44.0 ± 0.7
tie	5.5	6.5	7.0	10.9 ± 0.9	11.2 ± 1.1	microwave	9.8	8.2	11.2	24.4 ± 1.6	27.5 ± 1.4
suitcase	21.3	16.7	23.4	35.2 ± 0.8	40.0 ± 0.5	oven	16.4	13.7	12.4	32.0 ± 0.6	33.5 ± 0.4
frisbee	5.6	12.3	13.0	27.7 ± 3.5	45.4 ± 0.7	toaster	0.0	0.0	0.0	0.0 ± 0.0	0.0 ± 0.1
skis	1.0	1.6	1.5	2.3 ± 0.1	5.4 ± 0.4	sink	9.5	10.8	17.8	27.4 ± 0.8	25.9 ± 1.0
snowboard	2.8	5.3	16.3	8.3 ± 1.3	15.8 ± 1.3	refrigerator	13.2	4.0	15.5	37.5 ± 0.4	41.4 ± 0.5
sports ball	1.9	7.9	9.8	1.0 ± 0.1	0.8 ± 0.1	book	7.5	0.4	12.3	18.9 ± 0.8	22.9 ± 0.6
kite	10.3	9.1	17.4	22.4 ± 0.8	28.0 ± 0.8	clock	16.5	17.8	20.7	26.4 ± 0.7	28.2 ± 0.7
baseball bat	1.7	1.0	4.8	3.9 ± 0.6	6.8 ± 1.1	vase	13.4	18.4	23.9	16.7 ± 0.7	23.8 ± 0.6
baseball glove	0.5	0.6	1.2	4.2 ± 1.9	24.6 ± 1.0	scissors	12.2	16.5	17.3	24.9 ± 1.1	25.2 ± 0.7
skateboard	6.6	7.1	14.4	17.4 ± 0.7	21.2 ± 0.3	teddy bear	41.0	47.0	46.3	48.7 ± 1.1	56.0 ± 0.3
surfboard	3.3	7.7	13.5	11.5 ± 1.4	23.4 ± 4.0	hair dryer	0.0	0.0	0.0	0.0 ± 0.0	0.0 ± 0.0
tennis racket	5.5	9.1	6.8	6.9 ± 0.5	13.7 ± 2.0	toothbrush	2.0	2.8	4.5	9.5 ± 1.8	16.0 ± 2.6
bottle	9.6	13.2	22.3	17.8 ± 0.5	18.8 ± 0.7	**mean IoU**	**20.4**	**22.4**	**26.0**	**32.2 ± 0.2**	**36.3 ± 0.1**

Se observa que el modelo propuesto supera los trabajos anteriores basados en etiquetas de clasificación en casi todas las clases semánticas, presentando un buen desempeño para diversos tipos de objeto. El modelo presenta resultados particularmente buenos para clases de objetos complejos, que suelen ocupar un rol protagónico tanto en las imágenes como en sus descripciones. Esto incluye categorías como "*person*"; animales como "*zebra*", "*elephant*", "*cat*", o "*bear*"; y vehículos como "*bus*", "*motorcycle*", "*airplane*", o "*train*". Todas estas categorías presentan una IoU por sobre el 50% en el caso del modelo basado en ResNet-38.

Por el contrario, las categorías que presentan los mayores problemas corresponden generalmente a objetos pequeños, que aparecen normalmente en escenas con muchos elementos, y que se omiten frecuentemente de las descripciones, como es el caso de "*spoon*", "*knife*", "*cup*", "*backpack*", o "*handbag*".

5.7 Resultados Cualitativos

Finalmente, en la Fig. 5.6 se presentan algunos resultados cualitativos para el conjunto de validación de MS-COCO, obtenidos utilizando el modelo de segmentación supervisado entrenado siguiendo la metodología propuesta. Se incluyen las predicciones generadas por la mejor versión del modelo, basada en ResNet-38, y también las generadas por la versión basada en VGG-16, para facilitar la comparación con trabajos anteriores.

Los resultados dan cuenta de que el modelo es capaz de segmentar correctamente objetos de diversas categorías, y en un amplio rango de escalas, pese a haber sido entrenado utilizando únicamente anotaciones a nivel de imagen. Pese a que ambas versiones permiten obtener buenos resultados en casos similares, se observa que el modelo entrenado con las máscaras generadas a partir de la red de localización basada en ResNet-38 tiende en general a producir máscaras más completas y precisas.

Las últimas dos filas de cada columna ilustran algunos de los tipos más comunes de error del sistema. Los casos que presentan mayor dificultad para el modelo consisten en general en escenas con múltiples objetos pequeños, sobre todo de categorías poco llamativas. Este tipo de objetos suele estar ausente de las descripciones, por lo que tampoco aparecen en las máscaras de segmentación artificiales. Consecuentemente, el modelo de segmentación supervisado tiende a replicar estos mismos sesgos erróneos.

Algunos ejemplos de este efecto corresponden a las botellas, vasos, y cubiertos de la penúltima imagen de la primera columna, las bolsas de la última imagen de la misma columna, y el guante y la pelota de la penúltima fila de la segunda columna. El último ejemplo de la segunda columna muestra otro tipo de error común, en el que el modelo es incapaz de identificar la categoría de ciertos objetos por falta de contexto, como ocurre en este caso con las sábanas de la cama, que podrían corresponder, por ejemplo, a un mantel o a cortinas.

Figura 5.6: Ejemplos de resultados de segmentación obtenidos con el método propuesto sobre el conjunto de validación de MS-COCO, usando la red de localización basada en VGG-16 y en ResNet-38. Las últimas dos filas de ambas columnas muestran casos típicos de error.

Capítulo 6

Conclusiones y Trabajo Futuro

En el presente trabajo de tesis se desarrolló una metodología completa para abordar el problema de segmentación semántica utilizando únicamente imágenes anotadas con descripciones en lenguaje natural. La componente principal de la metodología propuesta consiste en un módulo de procesamiento de texto, que utiliza la estructura sintáctica de las descripciones para identificar menciones de objetos y sus atributos, y luego explota las relaciones definidas por una base de conocimiento para asignar etiquetas semánticas útiles a cada objeto. Esto permite extraer una supervisión visual estructurada a partir de descripciones sin necesidad de anotaciones adicionales, por lo que este módulo puede extenderse de forma inmediata a otras aplicaciones y categorías de objetos.

Se presentó además una nueva red de localización, que puede ser entrenada utilizando la información extraída para generar mapas de activación asociados tanto a etiquetas de clasificación, como a pares compuestos del tipo categoría-atributo. Finalmente, se describió un método para generar máscaras de segmentación artificiales, aprovechando todos los tipos de mapas de localización generados por el modelo. Estas máscaras pueden ser posteriormente utilizadas para entrenar un modelo de segmentación en régimen supervisado.

Se presentaron también múltiples resultados experimentales que demuestran la efectividad de la metodología propuesta. En primer lugar, se evaluó la calidad de las etiquetas de clasificación generadas por el módulo de procesamiento de texto, observándose mejoras significativas respecto de la estrategia de búsqueda exacta utilizada por métodos anteriores, sobre todo en términos de *recall* e intersección sobre unión de las predicciones. Adicionalmente, se demostró que esta mejora en las etiquetas de clasificación para categorías relevantes se traduce en una mejora sustancial en la calidad de las máscaras de segmentación generadas por el modelo de localización, lo que deriva en un aumento de 3.0% mIoU en el conjunto de validación de MS-COCO para el modelo supervisado. Estos resultados comprueban la efectividad de la metodología propuesta para extraer etiquetas de clasificación más completas a partir de descripciones, así como la importancia de esta mejora sobre la calidad del modelo de segmentación final.

Además de las etiquetas para categorías relevantes, se estudia el impacto de otros dos tipos de elementos visuales detectados en descripciones por el módulo propuesto: atributos,

y categorías de fondo. Por una parte, se comprueba que la información sobre atributos sirve principalmente para mejorar el *recall* de las máscaras de segmentación, al permitir identificar regiones relevantes, pero menos útiles para clasificación, a partir de sus características visuales de bajo nivel. Adicionalmente, los mapas asociados a categorías complementarias sirven principalmente para mejorar la precisión de las máscaras artificiales, al introducir información positiva para las regiones de fondo de la imagen, que facilitan la separación entre los objetos relevantes y su contexto. Ambos tipos de información complementan los mapas de activación para etiquetas de clasificación usuales, mejorando el desempeño del modelo final de segmentación sobre el conjunto de validación de MS-COCO en 1.9% mIoU.

Se realizan también experimentos adicionales empleando un modelo del estado del arte para *visual grounding* [22], entrenado para localizar texto arbitrario en imágenes. Este modelo se utiliza para generar mapas de activación a partir de distintas componentes de supervisión extraídas por el módulo de procesamiento de texto, que luego se combinan siguiendo la metodología propuesta para generar máscaras de segmentación y entrenar el mismo modelo supervisado. Aprovechando todas las etiquetas visuales generadas, se obtienen mejoras de 3.5% mIoU para el modelo de segmentación final, comparado con el caso en el que se utilizan solo etiquetas para categorías relevantes detectadas con búsqueda exacta, como en trabajos anteriores. Estos experimentos permiten comprobar la efectividad de la supervisión extraída, así como de la metodología de combinación propuesta, para mejorar la calidad de los mapas de segmentación artificiales, incluso ignorando su efecto durante el entrenamiento del modelo. Adicionalmente, se verifica que los resultados finales obtenidos con el modelo de *visual grounding* están 1.5% mIoU por debajo de los obtenidos con la red de localización propuesta, pese a que el modelo pre-entrenado utiliza una arquitectura convolucional mucho más profunda. Esto corrobora la importancia de focalizar el entrenamiento del modelo en información visual relevante para obtener mapas de activación precisos para WSSS.

Para evaluar las diferencias entre la supervisión extraída a partir de descripciones y las etiquetas de clasificación usuales, se experimenta entrenando la red de localización propuesta utilizando las etiquetas *ground truth* de MS-COCO. Los resultados obtenidos dan cuenta de que las etiquetas de clasificación para categorías relevantes extraídas por el módulo de procesamiento de texto permiten mejorar la calidad del modelo de segmentación final. Esto se debe principalmente a que las descripciones tienden a mencionar con mayor frecuencia objetos relevantes y fáciles de identificar, lo que descarta una gran cantidad de instancias pequeñas o incompletas, que en la práctica actúan como falsos positivos en el caso de las etiquetas *ground truth*, perjudicando la calidad de las máscaras generadas. Adicionalmente, la incorporación de información complementaria sobre atributos y clases de fondo permite extender aún más estas diferencias, superando la versión entrenada con etiquetas de clasificación *ground truth* por un margen de 2.7% mIoU en el conjunto de validación de MS-COCO. Estos resultados demuestran la efectividad de las descripciones en lenguaje natural como fuente de supervisión alternativa para entrenar modelos de segmentación semántica.

Finalmente, se presenta una comparación exhaustiva de la metodología propuesta respecto de trabajos previos para segmentación semántica utilizando distintos tipos de supervisión débil. En particular, se comprueba que la versión del modelo propuesto basada en VGG-16 alcanza 0.322 mIoU sobre el conjunto de validación de MS-COCO, superando todos los trabajos previos para el problema de WSSS utilizando solo información a nivel de imagen

basados en la misma arquitectura. Adicionalmente, la mejor versión del método propuesto alcanza 0.363 mIoU, superando el estado del arte para esta base de datos por 7.8% mIoU, lo que corresponde a una mejora relativa de 27.4%. De esta manera, se concluye que el trabajo realizado cumple con el objetivo general propuesto.

Adicionalmente, los resultados de esta tesis han sido aceptados para publicación en un artículo que será publicado en la revista (WoS) IEEE Access [100].

Pese a los avances que representa la metodología desarrollada en la presente tesis, existen aún importantes limitaciones y posibilidades de expansión para los métodos propuestos. A continuación se describen algunas de las líneas de trabajo futuro más prometedoras.

En primer lugar, se observa en los experimentos realizados que el desempeño del modelo propuesto está en parte limitado por el carácter incompleto de las descripciones en lenguaje natural. Si una determinada categoría de objeto no se menciona en las descripciones, estará ausente de las máscaras de segmentación generadas, disminuyendo el *recall* para esta clase en el modelo de segmentación final. Adicionalmente, existen casos en que ciertas categorías pueden aparecer mencionadas en las descripciones, pero estar solo implícitas en la imagen, reduciendo la precisión de los resultados.

Una posible forma de abordar estos problemas sería modelar el entrenamiento de la red de localización como un problema de clasificación con etiquetas ruidosas. Numerosos estudios que abordan esta problemática han sido publicados recientemente en la literatura [101, 102, 103], permitiendo en algunos casos refinar iterativamente las etiquetas visuales, inicialmente incompletas o corruptas, utilizando las predicciones del mismo modelo de clasificación. Sin embargo, la mayoría de estos trabajos se enfocan en el caso multi-clase, por lo que requieren de modificaciones adicionales para ser aplicados en el contexto multi-etiqueta, o para incorporar predicciones de atributos.

Otra posible línea de investigación consiste en extender la metodología propuesta al problema de segmentación de instancia, que tiene por objetivo generar máscaras de segmentación individuales para cada objeto relevante en la imagen. Las descripciones en lenguaje natural presentan ventajas comparativas importantes respecto de las etiquetas de clasificación utilizadas por los pocos estudios existentes que abordan este problema usando supervisión a nivel de imagen [104, 105, 81]. En particular, las descripciones contienen información de atributos, que pueden servir para generar mapas de activación separados para distintas instancias de la misma clase cuando estas poseen características visuales distintas, como se comprobó en los experimentos presentados en esta tesis. Adicionalmente, la presencia de múltiples hipónimos de la misma clase en una descripción podría utilizarse de forma similar para generar mapas con mayor granularidad. Por ejemplo, en vez de generar solo un mapa para la categoría "*person*", un modelo entrenado con descripciones podría generar mapas separados para "*man*", "*woman*", o "*child*", facilitando el paso a máscaras a nivel de instancia. Más aún, las descripciones contienen adjetivos cuantificadores, que pueden utilizarse para guiar la separación de los mapas por clase en máscaras por instancias.

Bibliografía

[1] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple Does It: Weakly supervised instance and semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1665–1674, Honolulu, HI, USA, jul 2017. IEEE.

[2] Jifeng Dai, Kaiming He, and Jian Sun. BoxSup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1635–1643, Santiago, Chile, dec 2015. IEEE.

[3] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. ScribbleSup: Scribble-supervised convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3159–3167, Las Vegas, NV, USA, jun 2016. IEEE.

[4] Amy L Bearman, Olga Russakovsky, Vittorio Ferrari, and Fei-Fei Li. What's the point: Semantic segmentation with point supervision. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 549–565, Amsterdam, The Netherlands, 2016.

[5] Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 695–711, Amsterdam, The Netherlands, 2016.

[6] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 6488–6496, Honolulu, HI, USA, jul 2017. IEEE.

[7] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. Weakly-supervised semantic segmentation network with deep seeded region growing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 7014–7023, Salt Lake City, UT, USA, 2018.

[8] Yunchao Wei, Huaxin Xiao, Honghui Shi, Zequn Jie, Jiashi Feng, and Thomas S. Huang. Revisiting dilated convolution: A simple approach for weakly- and semi-supervised semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 7268–7277, Salt Lake City, UT, USA, 2018.

[9] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4981–4990, Salt Lake City, UT, USA, 2018.

[10] Jungbeom Lee, Eunji Kim, Sungmin Lee, Jangho Lee, and Sungroh Yoon. FickleNet: Weakly and semi-supervised semantic image segmentation using stochastic inference. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5267–5276, Long Beach, CA, USA, 2019.

[11] Xiang Wang, Sifei Liu, Huimin Ma, and Ming-Hsuan Yang. Weakly-supervised semantic segmentation by iterative affinity learning. *Int. J. Comput. Vis.*, 2020.

[12] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2921–2929, Las Vegas, NV, USA, jun 2016. IEEE.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 248–255, Miami, FL, USA, jun 2009. IEEE.

[14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 740–755, Zurich, Switzerland, 2014.

[15] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The Pascal visual object classes (VOC) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, jun 2010.

[16] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, pages 2556–2565, Melbourne, Australia, 2018. Association for Computational Linguistics.

[17] Johann Sawatzky, Debayan Banerjee, and Juergen Gall. Harvesting information from captions for weakly supervised semantic segmentation. In *Proc. ICCV Workshops*, pages 4481–4490, Seoul, South Korea, oct 2019. IEEE.

[18] Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. Weakly-supervised visual grounding of phrases with linguistic structures. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5253–5262, Honolulu, HI, USA, jul 2017. IEEE.

[19] Chaorui Deng, Qi Wu, Qingyao Wu, Fuyuan Hu, Fan Lyu, and Mingkui Tan. Visual grounding via accumulated attention. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 7746–7755, Salt Lake City, UT, USA, 2018.

[20] Kan Chen, Jiyang Gao, and Ram Nevatia. Knowledge aided consistency for weakly supervised phrase grounding. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4042–4050, Salt Lake City, UT, USA, 2018.

[21] Fang Zhao, Jianshu Li, Jian Zhao, and Jiashi Feng. Weakly supervised phrase localization with multi-scale anchored transformer network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5696–5705, Salt Lake City, UT, USA, 2018.

[22] Martin Engilberge, Louis Chevallier, Patrick Pérez, and Matthieu Cord. Finding beans in burgers: Deep semantic-visual embedding with localization. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3984–3993, Salt Lake City, UT, USA, 2018.

[23] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1219–1228, Salt Lake City, UT, USA, 2018.

[24] Sebastian Schuster, Ranjay Krishna, Angel Chang, Li Fei-Fei, and Christopher D. Manning. Generating semantically precise scene graphs from textual descriptions for improved image retrieval. In *Proc. Workshop Vis. Lang.*, pages 70–80, Lisbon, Portugal, 2015. Association for Computational Linguistics.

[25] George A. Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, nov 1995.

[26] Convolution 2D. `https://embarc.org/embarc_mli/doc/build/html/MLI_kernels/convolution_2d.html`. Accessed: 2020-12-10.

[27] Max-Pooling. `https://computersciencewiki.org/index.php/File:MaxpoolSample2.png`. Accessed: 2020-12-11.

[28] Sumit Saha. A Comprehensive Guide to Convolutional Neural Networks. `https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53`, 2018. Accessed: 2020-12-11.

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet large scale visual recognition challenge. *Int. J. Comput. Vis.*, 115(3):211–252, 2015.

[30] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. ImageNet classification with deep convolutional neural networks. In F Pereira, C J C Burges, L Bottou, and K Q Weinberger, editors, *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, pages 1097–1105. Lake Tahoe, NV, USA, 2012.

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *Proc. Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, sep 2014.

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, volume 07-12-June, pages 1–9, Boston, MA, USA, 2015.

[33] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 770–778, Las Vegas, NV, USA, 2016.

[34] Recurrent neural network unfold. `https://commons.wikimedia.org/wiki/File:Recurrent_neural_network_unfold.svg`. Accessed: 2020-12-11.

[35] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[36] Christopher Olah. Understanding LSTM Networks. `https://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. Accessed: 2020-12-09.

[37] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, pages 1724–1734, Doha, Qatar, 2014.

[38] Tao Lei, Yu Zhang, Sida I Wang, Hui Dai, and Yoav Artzi. Simple recurrent units for highly parallelizable recurrence. In *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, pages 4470–4481, Brussels, Belgium, 2018.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, pages 5998–6008, Long Beach, CA, USA, 2017.

[40] Jay Alammar. The Illustrated Transformer. `https://jalammar.github.io/illustrated-transformer/`, 2018. Accessed: 2020-12-11.

[41] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, pages 3111–3119, Lake Tahoe, NV, USA, 2013.

[42] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, pages 1532–1543, Doha, Qatar, 2014.

[43] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. NAACL-HLT*, pages 2227–2237, New Orleans, LA, USA, 2018.

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proc. NAACL-HLT*, pages 4171–4186, Minneapolis, MN, USA, 2019. Association for Computational Linguistics.

[45] John D Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 282–289, Williamstown, MA, USA, 2001.

[46] Philipp Krähenbühl and Vladlen Koltun. Efficient inference in fully connected CRFs with gaussian edge potentials. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, pages 109–117, Granada, Spain, 2011.

[47] Justin Johnson, Ranjay Krishna, Michael Stark, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Li Fei-Fei. Image retrieval using scene graphs. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3668–3678, Boston, MA, USA, jun 2015. IEEE.

[48] Hao Wu, Jiayuan Mao, Yufeng Zhang, Yuning Jiang, Lei Li, Weiwei Sun, and Wei-Ying Ma. Unified visual-semantic embeddings: Bridging vision and language with structured meaning representations. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 6609–6618, Long Beach, CA, USA, 2019.

[49] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. SPICE: Semantic propositional image caption evaluation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 382–398, Amsterdam, The Netherlands, 2016.

[50] Ning Xu, An-An Liu, Jing Liu, Weizhi Nie, and Yuting Su. Scene graph captioner: Image captioning based on structural visual representation. *J. Vis. Commun. Image Represent.*, 58:477–485, 2019.

[51] Danfei Xu, Yuke Zhu, Christopher B. Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3097–3106, Honolulu, HI, USA, jul 2017. IEEE.

[52] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph R-CNN for scene graph generation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 690–706, Munich, Germany, sep 2018.

[53] Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In *Proc. Annu. Meeting Assoc. Comput. Linguistics (ACL)*, volume 1, pages 423–430, Sapporo, Japan, 2003. Association for Computational Linguistics.

[54] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proc. Int. Conf. Lang. Resour. Eval. (LREC)*, pages 4585–4592, Reykjavik, Iceland, 2014.

[55] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, volume 2017-Janua, pages 6517–6525, Honolulu, HI, USA, dec 2017.

[56] Bjorn Barz and Joachim Denzler. Hierarchy-based image embeddings for semantic image retrieval. In *Proc. Winter Conf. Appl. Comput. Vis. (WACV)*, pages 638–647, Waikoloa Village, HI, USA, jan 2019. IEEE.

[57] Hang Zhao, Xavier Puig, Bolei Zhou, Sanja Fidler, and Antonio Torralba. Open vocabulary scene parsing. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 2021–2029,

Venice, Italy, oct 2017. IEEE.

[58] David Golub, Ahmed El-Kishky, and Roberto Martin-Martin. Leveraging pretrained image classifiers for language-based segmentation. In *Proc. Winter Conf. Appl. Comput. Vis. (WACV)*, pages 1999–2008, Snowmass Village, CO, USA, mar 2020. IEEE.

[59] Loïc Vial, Benjamin Lecouteux, and Didier Schwab. Sense vocabulary compression through the semantic knowledge of WordNet for neural word sense disambiguation. In *Proc. Global Wordnet Conf.*, Wroclaw, Poland, 2019.

[60] Luyao Huang, Chi Sun, Xipeng Qiu, and Xuanjing Huang. GlossBERT: BERT for word sense disambiguation with gloss knowledge. In *Proc. Conf. Empirical Methods Natural Lang. Process. Int. Joint Conf. Natural Lang. Process. (EMNLP-IJCNLP)*, pages 3507–3512, Hong Kong, China, 2019. Association for Computational Linguistics.

[61] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3431–3440, Boston, MA, USA, jun 2015. IEEE.

[62] 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. *IEEE Trans. Pattern Anal. Mach. Intell*, 40(4):834–848, apr 2018.

[63] Tsung Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, volume 2017-Janua, pages 936–944, Honolulu, HI, USA, jul 2017. IEEE.

[64] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell*, 39(12):2481–2495, dec 2017.

[65] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1520–1528, Santiago, Chile, dec 2015. IEEE.

[66] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 173–190, Glasgow, UK, aug 2020. Springer, Cham.

[67] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. MICCAI*, pages 234–241. Munich, Germany, 2015.

[68] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *Proc. Int. Conf. 3D Vis. (3DV)*, pages 565–571, Stanford, CA, USA, oct 2016. IEEE.

[69] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected CRFs. In *Proc. Int. Conf. Learn. Representations (ICLR)*, San Diego, CA, USA, dec 2015.

[70] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1529–1537, Santiago, Chile, dec 2015. IEEE.

[71] Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Ian Reid. Efficient piecewise training of deep structured models for semantic segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3194–3203, Las Vegas, NV, USA, jun 2016. IEEE.

[72] Ximin Cui, Ke Zheng, Lianru Gao, Bing Zhang, Dong Yang, and Jinchang Ren. Multiscale spatial-spectral convolutional network with image-based framework for hyperspectral imagery classification. *Remote Sens.*, 11(19):2220, sep 2019.

[73] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv preprint, jun 2017. arXiv:1706.05587.

[74] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 833–851, Munich, Germany, sep 2018.

[75] Yunchao Wei, Xiaodan Liang, Yunpeng Chen, Xiaohui Shen, Ming-Ming Cheng, Jiashi Feng, Yao Zhao, and Shuicheng Yan. STC: A simple to complex framework for weakly-supervised semantic segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(11):2314–2320, nov 2017.

[76] Seong Joon Oh, Rodrigo Benenson, Anna Khoreva, Zeynep Akata, Mario Fritz, and Bernt Schiele. Exploiting saliency for object segmentation from image level labels. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5038–5047, Honolulu, HI, USA, jul 2017. IEEE.

[77] Arslan Chaudhry, Puneet Kumar Dokania, and Philip H S Torr. Discovering class-specific pixels for weakly-supervised semantic segmentation. In *Proc. Brit. Mach. Vis. Conf. (BMVC)*, 2017.

[78] Xiang Wang, Shaodi You, Xi Li, and Huimin Ma. Weakly-supervised semantic segmentation by iteratively mining common object features. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1354–1362, Salt Lake City, UT, USA, 2018.

[79] Qi Yao and Xiaojin Gong. Saliency guided self-attention network for weakly and semi-supervised semantic segmentation. *IEEE Access*, 8:14413–14423, 2020.

[80] Tianyi Zhang, Guosheng Lin, Jianfei Cai, Tong Shen, Chunhua Shen, and Alex C. Kot. Decoupled spatial neural attention for weakly supervised semantic segmentation. *IEEE Trans. Multim.*, 21(11):2930–2941, nov 2019.

[81] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 2209–2218, Long Beach, CA, USA, 2019.

[82] Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Tell me where to look: guided attention inference network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 9215–9223, Salt Lake City, UT, USA, 2018.

[83] Qibin Hou, PengTao Jiang, Yunchao Wei, and Ming-Ming Cheng. Self-erasing network for integral object attention. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, pages 549–559, Montreal, Canada, 2018.

[84] László Lovász. Random walks on graphs: A survey. *Combinatorics*, pages 1–46, 1993.

[85] Zifeng Wu, Chunhua Shen, and Anton van den Hengel. Wider or deeper: Revisiting the ResNet model for visual recognition. *Pattern Recognit.*, 90:119–133, 2019.

[86] Mingzhe Wang, Mahmoud Azab, Noriyuki Kojima, Rada Mihalcea, and Jia Deng. Structured matching for phrase localization. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 696–711, Amsterdam, The Netherlands, 2016.

[87] Zhenxing Niu, Mo Zhou, Le Wang, Xinbo Gao, and Gang Hua. Hierarchical multimodal LSTM for dense visual-semantic embedding. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 1899–1907, Venice, Italy, oct 2017. IEEE.

[88] Hassan Akbari, Svebor Karaman, Surabhi Bhargava, Brian Chen, Carl Vondrick, and Shih-Fu Chang. Multi-level multimodal common semantic space for image-phrase grounding. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 12476–12486, Long Beach, CA, USA, 2019.

[89] Thibaut Durand, Nicolas Thome, and Matthieu Cord. WELDON: Weakly supervised learning of deep convolutional neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 4743–4752, Las Vegas, NV, USA, jun 2016. IEEE.

[90] Keren Ye, Mingda Zhang, Adriana Kovashka, Wei Li, Danfeng Qin, and Jesse Berent. Cap2Det: Learning to amplify weak caption supervision for object detection. In *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, pages 9685–9694, Seoul, South Korea, oct 2019. IEEE.

[91] Achiya Jerbi, Roei Herzig, Jonathan Berant, Gal Chechik, and Amir Globerson. Learning object detection from captions via textual scene attributes. arXiv preprint, sep 2020. arXiv:2009.14558.

[92] Xinyu Zhou, Cong Yao, He Wen, Yuzhi Wang, Shuchang Zhou, Weiran He, and Jiajun Liang. EAST: An efficient and accurate scene text detector. In *Proc. IEEE Conf. Com-*

put. Vis. Pattern Recognit. (CVPR), volume 2017-Janua, pages 2642–2651, Honolulu, HI, USA, jul 2017. IEEE.

[93] Chen Huang, Yining Li, Chen Change Loy, and Xiaoou Tang. Learning deep representation for imbalanced classification. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 5375–5384, Las Vegas, NV, USA, jun 2016. IEEE.

[94] Hongming Zhou, Kang Song, Xianglei Zhang, Wenyong Gui, and Qiusuo Qian. WAILS: Watershed algorithm with image-level supervision for weakly supervised semantic segmentation. *IEEE Access*, 7:42745–42756, 2019.

[95] Alessandro Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3282–3289. IEEE, jun 2012.

[96] Suyog Dutt Jain and Kristen Grauman. Supervoxel-consistent foreground propagation in video. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 656–671. Springer, Cham, 2014.

[97] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 558–567, Long Beach, CA, USA, 2019.

[98] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch SGD: training ImageNet in 1 Hour. arXiv preprint, jun 2017. arXiv:1706.02677.

[99] Fatemehsadat Saleh, Mohammad Sadegh Aliakbarian, Mathieu Salzmann, Lars Petersson, Stephen Gould, and Jose M. Alvarez. Built-in foreground/background prior for weakly-supervised semantic segmentation. In *Proc. Eur. Conf. Comput. Vis. (ECCV)*, pages 413–432, Amsterdam, The Netherlands, 2016.

[100] Daniel R. Vilar and Claudio A. Perez. Extracting structured supervision from captions for weakly supervised semantic segmentation. *IEEE Access*. Accepted 2021-04-23.

[101] Lu Jiang, Zhengyuan Zhou, Thomas Leung, Li-Jia Li, and Li Fei-Fei. MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 2309–2318, Stockholm, Sweden, 2018.

[102] Bo Han, Quanming Yao, Xingrui Yu, Gang Niu, Miao Xu, Weihua Hu, Ivor Tsang, and Masashi Sugiyama. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *Proc. Annu. Conf. Neural Inf. Process. Syst. (NIPS)*, volume 31, pages 8527–8537, Montreal, Canada, 2018.

[103] Pengfei Chen, Benben Liao, Guangyong Chen, and Shengyu Zhang. Understanding and utilizing deep neural networks trained with noisy labels. In *Proc. Int. Conf. Mach. Learn. (ICML)*, pages 1062–1070, Long Beach, CA, USA, 2019.

[104] Yanzhao Zhou, Yi Zhu, Qixiang Ye, Qiang Qiu, and Jianbin Jiao. Weakly supervised

instance segmentation using class peak response. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 3791–3800, Salt Lake City, UT, USA, 2018.

[105] Yunhang Shen, Rongrong Ji, Yan Wang, Yongjian Wu, and Liujuan Cao. Cyclic guidance for weakly supervised joint detection and segmentation. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 697–707, Long Beach, CA, USA, 2019.

Anexo A

Cómputo de Pesos de la Función de Pérdida

En esta sección se formaliza el método utilizado para computar los pesos de las funciones de pérdidas con las que se entrena la red de localización propuesta, según se describen en la Sección 4.2.2. En primer lugar, los pesos w_c^+ y w_c^- de (4.9) se obtienen como:

$$w_c^+ = \frac{(N_c^+ + N_c^-)}{2 \cdot N_c^+} \quad \text{y} \quad w_c^- = \frac{(N_c^+ + N_c^-)}{2 \cdot N_c^-}, \tag{A.1}$$

donde N_c^+ y N_c^- son, respectivamente, el total de ejemplos positivos y negativos para la clase c en la base de datos, i.e.,

$$N_c^+ = \sum_{n=1}^{N} \mathbf{1}_{\mathcal{C}_n}(c) \quad \text{y} \quad N_c^- = N - N_c^+. \tag{A.2}$$

Aquí, $\mathbf{1}_{\mathcal{B}}$ representa la función indicatriz de un conjunto $\mathcal{B}$, de modo que $\mathbf{1}_{\mathcal{B}}(x)$ vale 1 si $x \in B$, y 0 en caso contrario. Estos valores también aseguran que todas las categorías relevantes contribuyen el mismo peso acumulado durante el entrenamiento, como ocurre en el caso usual dado por $w_c^+ = w_c^- = 1$.

Por otra parte, los pesos $\tilde{w}_a^+$ y $\tilde{w}_a^-$ de (4.10) se computan de forma análoga a w_c^+ y w_c^-, respectivamente, pero considerando el total de conjuntos $\mathcal{A}_{n,c}$ no vacíos que incluyen o no al atributo a en la base de datos, en lugar del número de imágenes. Es decir,

$$\tilde{w}_a^+ = \frac{(\tilde{N}_a^+ + \tilde{N}_a^-)}{2 \cdot \tilde{N}_a^+} \quad \text{y} \quad \tilde{w}_a^- = \frac{(\tilde{N}_a^+ + \tilde{N}_a^-)}{2 \cdot \tilde{N}_a^-}, \tag{A.3}$$

donde $\tilde{N}_c^+$ y $\tilde{N}_c^-$ están dados por:

$$\tilde{N}_a^+ = \sum_{n=1}^{N} \sum_{c \in \mathcal{C}_n} \mathbf{1}_{\mathcal{A}_{n,c}}(a) \quad \text{y} \quad \tilde{N}_a^- = \sum_{n=1}^{N} \left| \{ c \mid c \in \mathcal{C}_n, \mathcal{A}_{n,c} \neq \varnothing \} \right| - \tilde{N}_a^+. \tag{A.4}$$