

FACULTAD DE MATEMÁTICA, ASTRONOMÍA,
FÍSICA Y COMPUTACIÓN

UNIVERSIDAD NACIONAL DE CÓRDOBA



Zero-Shot Object Detection

TESIS

QUE PARA OBTENER EL TÍTULO DE

LICENCIADO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

AGUSTIN HORACIO URQUIZA TOLEDO

DIRECTOR: JORGE SANCHEZ

CÓRDOBA, ARGENTINA

2019

DEDICATORIA

Agradecimientos

Muchas gracias, Muchas gracias, Muchas gracias, Muchas gracias,
Muchas gracias,Muchas gracias, Muchas gracias. Son muchas
gracias?

Resumen

Pasado el año 2000 se dan dos hechos que harán que las imágenes en Internet de un gran salto. Por un lado se empiezan a popularizar las cámaras digitales y por otro lado las conexiones de Internet subieron su velocidad. Esto generó la necesidad de crear métodos veloces y eficaces que faciliten la extracción de información en este tipo de datos. Luego, a partir del año 2010 con la “Revolución” del Aprendizaje profundo, surgieron una gran cantidad de métodos para realizar esta tarea, entre ellos los Detectores. Pero esto generó la necesidad de tener una gran cantidad de imágenes anotadas, que en algunos casos no resulta viable. **Zero-shot Object Detection** intenta atacar este problema. En este artículo, abordamos este desafiante problema utilizando características visuales y descripciones semánticas, que tiene como objetivo detectar y reconocer simultáneamente instancias de conceptos novedosos. Para esto analizamos distintos trabajos y llevaremos a cabo experimentos, que aportaran una noción del estado actual de esta área.

Índice general

1. Introducción y Motivación	1
1.1. Historia	1
1.2. Detectores y ZSD	3
1.3. Estado del arte	5
1.4. Motivación	7
2. Definición del problema	9
2.1. Redes neuronales convolucionales	9
2.2. Word embedding	10
2.3. Propuestas de objetos	12
2.4. Multimodales	14
2.5. Aprendizaje por disparo cero (ZSL)	14
2.6. Detección de objeto por disparo cero (ZSD)	16
3. Diseño y Arquitectura	21
3.1. Arquitectura	21
3.2. Conjuntos de datos	23
3.3. Detalles de la implementación	26
4. Experimentos	29
4.1. Experimentación con Propuestas de objetos	29
4.2. Experimentación con CNN	30
4.3. Definición de métricas	31
4.4. Detalles de metodología de evaluación	35

5. Análisis de resultados	37
5.1. Resultados Cuantitativos	37
5.2. Resultados Cualitativos	37
6. Conclusiones y trabajo futuro	39
6.1. Aportes	39
6.2. Trabajo futuro	39
Bibliografía	41

Capítulo 1

Introducción y Motivación

1.1. Historia

La detección de objetos es una de las áreas de la visión por computadora que está creciendo muy rápidamente. Gracias al aprendizaje profundo, cada año, los nuevos algoritmos/modelos siguen superando a los anteriores. Aunque la visión por computadora recientemente tomó gran importancia (el momento decisivo ocurrió en 2012 cuando AlexNet ganó ImageNet), ciertamente no es un nuevo campo científico. Uno de los artículos más influyentes en Visión Informática fue publicado por dos neurofisiólogos, David Hubel y Torsten Wiesel, en 1959. Su publicación, titulada “Receptive fields of single neurons in the cat’s striate cortex” en español ”Campos receptivos de neuronas individuales en la corteza estriada del gato”, describieron las propiedades de respuesta central de las neuronas corticales visuales y cómo la experiencia visual de un gato moldea su arquitectura cortical. Los investigadores establecieron, a través de su experimentación Figura 1.1, que existen neuronas simples y complejas en la corteza visual primaria y que el procesamiento visual siempre comienza con estructuras simples como los bordes orientados. En la actualidad, es esencialmente el principio básico detrás del aprendizaje profundo.

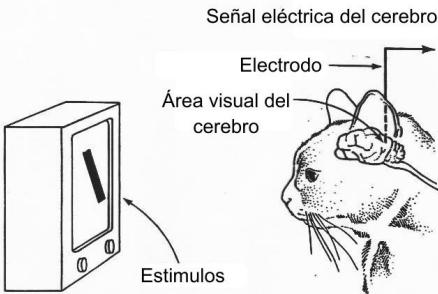


Figura 1.1: Simple explicación del experimento realizado por David Hubel y Torsten Wiesel

Otro echo importante en la historia de la visión por computadora fue, en 1959, Russell Kirsch y sus colegas desarrollaron un aparato que permitía transformar imágenes en cuadrículas de números que las máquinas de lenguaje binario podían entender. En la década de 1960 fue cuando la IA se convirtió en una disciplina académica y algunos de los investigadores, eran extremadamente optimistas sobre el futuro del campo. En este periodo, Seymour Papert, profesor del laboratorio de IA del MIT, decidió lanzar el Proyecto de Verano y resolver, en pocos meses, el problema de la visión artificial. Los estudiantes debían diseñar una plataforma que pudiera realizar, automáticamente, segmentación de fondo y extraer objetos no superpuestos de imágenes del mundo real. Claro esta que el proyecto no fue un éxito. Cincuenta años después, todavía no estamos cerca de resolver la visión por computadora. Sin embargo, ese proyecto fue, el nacimiento oficial de CV como campo científico. A este acontecimiento le siguieron una gran cantidad de investigaciones que hicieron grandes aportes al campo de la visión por computadoras. Como la tesis de doctorado de Roberts, Lawrence [12] en 1963, el paper de David Marr [6] en 1982, entre los mas reconocidos.

Pero los aportes mas influyentes a este campo empezaron a sur-

uir en los dos mil. En 2001 Paul Viola y Michael Jones presentaron el primer detector de rostros que funcionó en tiempo real. Aunque no se basa en el aprendizaje profundo, el algoritmo tenía una relación con el mismo, ya que, al procesar imágenes, aprendió qué características podría ayudar a localizar caras, inspirados en el experimento de David Hubel y Torsten Wiesel. En 2006 comenzó la competencia de Pascal VOC, que permitió evaluar el desempeño de diferentes métodos para el reconocimiento de la clase de objeto. En 2010 siguiendo los pasos de Pascal VOC, inicio el concurso de reconocimiento visual a gran escala ImageNet (ILSVRC). En 2010 y 2011, la tasa de error del ILSVRC en la clasificación de imágenes rondaba el 26 %. Pero en 2012, un equipo de la Universidad de Toronto ingresó a la competencia un modelo de red neuronal convolucional (AlexNet) y eso cambió todo. El modelo, similar en su arquitectura al LeNet-5 de Yann LeCun, logró una tasa de error del 16,4 %. En los años siguientes, las tasas de error en la clasificación de imágenes en ILSVRC cayeron a un pequeño porcentaje, como se observa en la Figura 1.2 y los ganadores, desde 2012, siempre han sido redes neuronales convolucionales.

1.2. Detectores y ZSD

La detección de objetos es un sub-problema de la visión artificial, que estudia cómo detectar la presencia de objetos en una imagen sobre su apariencia visual. Debido a la complejidad de poder detectar todas las instancias de todos los posibles objetos en una imagen, existen diferentes tareas que tratan de disminuir la dificultad. Para poder explicar los distintos problemas, es necesario distinguir dos conjuntos. Los datos de entrenamiento, consta de las imágenes que se usarán para entrenar el modelo, con sus respectivas etiquetas, es decir, que objetos se encuentran en la imagen, localización de los objetos, descripción de la imagen, o cualquier información extra que requiera la tarea. Las imágenes de prueba, es el conjunto donde se observará o medirá la eficiencia del modelo ya entrenado. Supongamos que las eti-

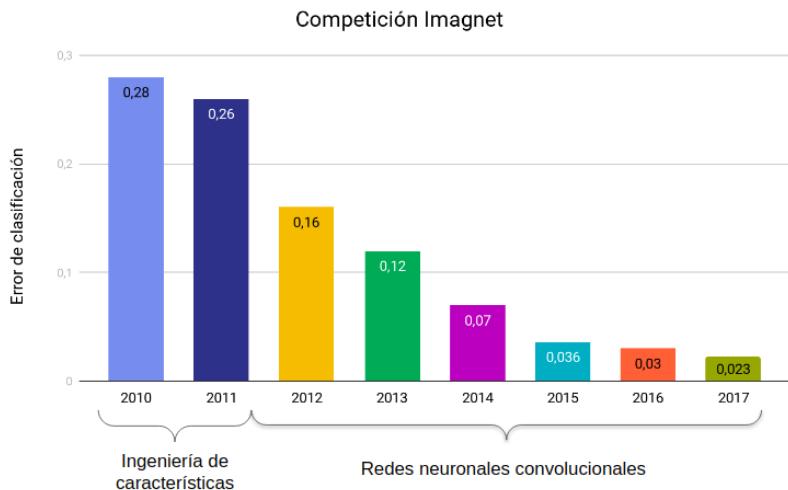


Figura 1.2: Evolución de los modelos propuestos en la competencia ILSVRC

quetas, solo cuenta con dos tipos de información, que clase de objeto es, es decir si es un perro, auto, persona, etc. y su localización en la imagen. Todas las clases que aparecen en los datos de entrenamiento llamaremos clases visibles o vistas. Toda aquella clase que no sea una clase vista la llamaremos invisible o no vista.

La **clasificación**, consta en un modelo capaz de retornar que objeto de las clases vistas se encuentra en una imagen. **Clasificación + localización**, además de poder clasificar tiene que ser capaz de ubicar el objeto en la imagen. Ambos modelos clasifican en una sola clase vista. El **reconocimiento de imagen**, predice que objetos perteneciente a las clases visibles están presente en la imagen. La **detección de objetos**, tiene que ser capaz además, de poder localizar dichos objetos. El tradicional **reconocimiento por disparo cero**, tiene que poder reconocer clases no vistas. Por último la **detección de objetos por disparo cero** o ZSD por sus siglas en inglés, debe localizar y clasificar todas las instancias de objetos en la imagen, sin depender

si es una clase vista o no. La Figura 1.3 muestra un ejemplo de las distintas tareas antes mencionadas.

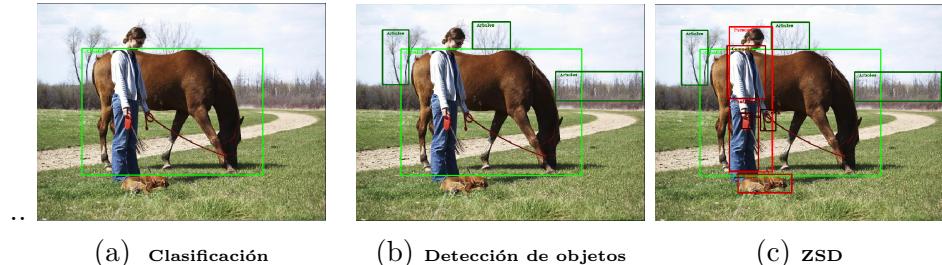


Figura 1.3: Este es jemplo de las tareas mencionadas. En la escala de los verdes se encuentran las clases vistas {Caballos, Árbol}, y en rojo las clases invisibles {Perro, Persona, Campera, Pantalón, Correa}

Existen otros problemas que no mencionamos acá, como la segmentación. Ya que en este documento trataremos la tarea de ZSD. En el capítulo Capítulo 2 formalizaremos lo aquí explicado.

1.3. Estado del arte

ZSD es un problema que tomo impulso recién en los últimos años, aunque sus estudios se remontan desde mucho antes, como ya mencionamos. Existen muchas técnicas y propuestas para poder resolver este problema, cuando se empezó a leer sobre este tema a fines del 2018 la mas utilizada era usando Multi-modales. Esta es una tecnica que se esta usando mucho como [1] quien utilizó imágenes, texto y sonido para generar representaciones discriminatorias profundas que se comparten en las tres modalidades. Del mismo modo, [18] utilizó imágenes y descripciones de texto para una mejor localización de la entidad visual basada en el lenguaje natural. La idea (para resolver el problema de ZSD) es utilizar un espacio compartido entre las re-

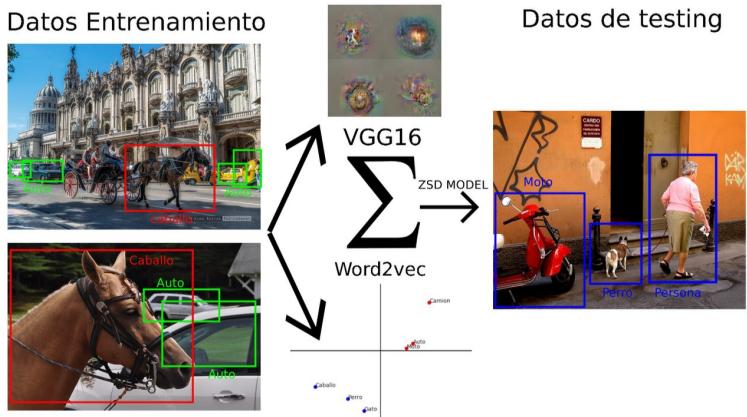


Figura 1.4: Se describe la tarea de detección de objetos por disparo cero, donde los objetos “Auto” y “Caballo” se observan durante el entrenamiento y “Persona”, “Perro” y “Moto” son clases invisibles. El enfoque localiza estas clases invisibles aprovechando las relaciones semánticas entre las clases visibles e invisibles y su aspecto visual.

presentaciones de visión y lenguaje. Para lograr esto se utiliza por un lado **Las Incrustaciones de palabras**, asignan palabras a una representación vectorial continua codificando similitud semántica entre palabras. Estos vectores de palabras funcionan bien en tareas tales como medir similitudes semánticas y sintácticas entre palabras. Entre los modelos mas famosos se encuentran Glove[10] y Word2vec[7]. Por otro lado se tiene que poder extraer los **vectores con representaciones visuales**, entre los mejores modelos se encuentran VGG [13] ResNet [14], Inception [15]. Todos estos modelos usan redes profundas para extraer dichas características. Como se muestra en la Figura 1.4 se utiliza la combinación de representación de palabras y las visuales para inferir un objetos nunca antes vistos por el modelo.

A fines del 2018 se encontro tres trabajos paralelos que apuntaban a resolver el problema de ZSD. [11] [19] [2]. Luego de leer los todos, por una decision personal y con ayuda de mi director elegimos atacar

el problema basándonos en el Paper de Bansal [2]. Así tambien sacamos muchos conceptos sobre disparo cero generalizado de [17]. En Zero-Shot Object Detection[2], enfrentan el problema de disparo cero de manera similar a la que tenemos los humanos de reconocer un objeto que nunca antes hemos visto, dada una descripción semántica, podemos imaginar el aspecto del objeto. Es decir, asociamos tanto la palabra que representa el objeto como su aspecto visual. Se utiliza dos extractores para simular estas cualidades, uno semántico y otro visual. Luego, en el momento del entrenamiento se proporcionan ejemplos visuales para solo algunas clases, pero durante la prueba se espera que el modelo reconozca instancias de clases que no se vieron, con la restricción de que las nuevas clases estén semánticamente relacionadas con las clases de entrenamiento.

1.4. Motivación

Hoy en día, hay una gran cantidad de modelos, capaces de detectar objetos en una imagen, como son las redes YOLO o Faster R-CNN. Estos, como otros no mencionados, poseen una excelente performance. Pero tienen una gran limitación, necesitan una gran cantidad de imágenes anotadas, para cada clase que se quiere detectar. Conseguir un gran numero de anotaciones, pude resultar un gran desafío, ya sea por la naturaliza del problema o por los grandes costo que esto lleva.

ZSD es una habilidad que los humanos ya tienen. De hecho, podemos aprender muchas cosas con solo un “conjunto de datos mínimo”. Por ejemplo, tendemos a diferenciar variedades de la misma fruta o frutas de aspecto similar, aun si hemos visto muy pocas veces cada tipo de fruta. La situación es diferente para las máquinas. Necesitan muchas imágenes para aprender a adaptarse a la variación que se produce de forma natural en lo humanos. Esta habilidad proviene de

nuestra base de conocimientos lingüísticos existente, que proporciona una descripción de alto nivel de una clase nueva o no vista y establece una conexión entre ella y las clases que ya conocemos.

Por unos minutos dejemos llevarnos por la imaginación y supongamos que se quiere crear un programa capaz de reconocer todos los objetos en una imagen, pero objetos de cualquier índole, animales, plantas, artículos de limpieza, o cualquier cosa que se te venga a la mente. Sería casi imposible, si es que no lo es, generar un conjunto de datos que contenga una cantidad considerable de imágenes de todos los objetos posibles. Esta idea puede sonar muy descabellada, o no, pero no se puede negar su potencial y su gran cantidad de usos como en interpretaciones de escenas, seguridad, etc. A medida que ZSD continúa desarrollándose, se espera ver más aplicaciones, como mejores recomendaciones y soluciones más avanzadas que marcan automáticamente el contenido inadecuado dentro de las redes sociales, como así también un fuerte desarrollo en el campo de la robótica.

Capítulo 2

Definición del problema

2.1. Redes neuronales convolucionales

Las redes neuronales convolucionales CNN por sus siglas en ingles, es un tipo de modelo de aprendizaje profundo para procesar datos que tiene un formato de cuadrícula, como las imágenes. Está inspirado en la organización de la corteza visual de los animales, diseñada para aprender de forma automática y adaptativa, patrones en jerarquías, de bajo a alto nivel. Por lo genera una red CNN se compone de tres tipos de capas: convolución, agrupación y capas completamente conectadas, como se puede ver en Figura 2.1. Las dos primeras, realizan extracción de características, mientras que la tercera, las relaciona y genera una salida. La capa de convolución desempeña un papel clave en CNN, se compone de una pila de operaciones matemáticas, como la convolución, que es un tipo especializado de operación lineal. En las imágenes en 2D estas redes son muy utilizadas, por su alta eficiencia para tareas de visión artificial, como en la clasificación y segmentación de imágenes, entre otras aplicaciones.

Algunos ejemplos de redes CNN son, VGG16 que posee 13 capas de convolución, 5 de agrupación y una totalmente conectada. Alex-Net conocida por ganar la competencia 2012 ImageNet LSVRC-2012,

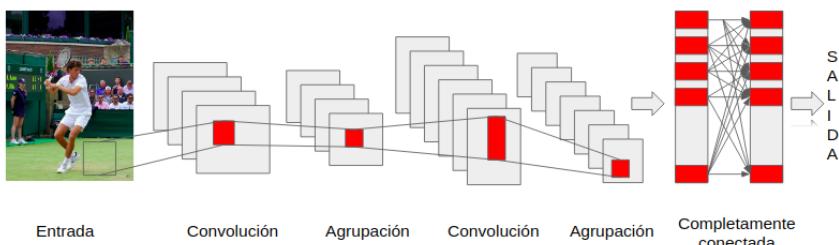


Figura 2.1: Una arquitectura simplificada de una red neuronal convolucional.

contiene 5 capas convolucionales, 3 capas de agrupación y 3 capas completamente conectadas.

Las redes CNN se utilizan para resolver distintos problemas como, la detección de objetos, Fast R-CNN [4]. La comprensión visual de escenas de calles urbanas [3], entre otros. En este trabajo utilizamos la salida de las redes CNN (la capa completamente conectada), como un vector de características visual de la imagen. Debido a que son muy eficaces reconociendo patrones, los vectores de dos imágenes que tienen un aspecto similar, también tienden a tener una semejanza.

2.2. Word embedding

Al igual que con las imágenes, que utilizamos las redes CNN, para obtener un vector que represente a la misma, es necesario un procedimiento para poder representar las palabras, con algún objeto matemático. Hay muchas formas de representar palabras, pero la más conocida es word embedding, es una técnica de aprendizaje en el campo de procesamiento del lenguaje natural (PLN). Es capaz de capturar el contexto de una palabra en un documento, calcular similitud semántica y sintáctica con otras palabras, etc.

Para entender como funcionan, consideremos las oraciones con un significado similar: “Que tengas un buen día.” y “Que tengas un gran día.”. Si construimos un vocabulario exhaustivo

$$V = \{que, tengas, un, buen, gran, dia\}.$$

Ahora, si creemos un vector codificado para cada una de estas palabras. En donde cada vector tiene el tamaño de V y son todos 0 excepto por el elemento en el índice que representa la palabra correspondiente en el vocabulario, que contiene un 1. No resulta una buena representación ya que la distancia entre *gran* y *buen* es la misma que entre *tengas* y *buen*. El objetivo es que las palabras con un contexto similar ocupen posiciones espaciales cercanas. Para lograr esto, se introduce cierta dependencia de una palabra de las otras.

Word2Vec [8] desarrollado por Tomas Mikolov en 2013. Es un modelo particularmente eficiente desde el punto de vista computacional. Este modelo se encuentra disponible de dos formas: Continuous Bag-of-Words (CBOW) o el modelo Skip-Gram. En CBOW, las representaciones distribuidas de contexto (o palabras circundantes) se combinan para predecir la palabra en el medio. En nuestro ejemplo *gran* y *buen* están rodeado de un contexto similar por lo cual resultan en vectores similares. Es varias veces más rápido de entrenar que el skip-gram, y tiene una precisión ligeramente mejor para las palabras frecuentes. Mientras que en el modelo Skip-gram, la representación distribuida de la palabra de entrada se usa para predecir el contexto. Se entrena con una tarea falsa que, dada una palabra, intentaremos predecir las palabras vecinas. En realidad, el objetivo es solo aprender los pesos de la capa oculta que son en realidad los “vectores de palabras” que estamos tratando de aprender. *Gran* se entrena para predecir el contexto *un* y *dia*, al igual que con *buen*. Funciona bien con una pequeña cantidad de datos de entrenamiento, representa bien incluso palabras o frases raras.

En este trabajo, aprovechamos la capacidad de capturar similitudes semántica que tiene word embedding, para relacionar las clases vistas con las clases no vistas. Utilizamos un modelo pre-entrenado generado a partir de word2vec para representar las palabras de las distintas clases.

2.3. Propuestas de objetos

En problemas de detección de objetos, generalmente se tiene que encontrar todos los objetos posibles en la imagen, como todos los autos, todas las bicicletas, etc. La localización de objetos se refiere a identificar la ubicación de uno o varios objetos en la imagen. Un algoritmo de localización de objetos generará las coordenadas de la ubicación de los objetos con respecto a la imagen. En visión artificial, la forma más popular de representar la ubicación de los objetos es con la ayuda de cuadros delimitadores (Bounding Boxes). Existen muchos algoritmos y redes que intenta resolver este problema como por ejemplo, **ventana deslizante**, **Edge-Boxes** [20], **Selective search** [16] ect. En ZSD las propuestas de objetos cumple un papel importante, ya que se necesita extraer todas las instancias de los objetos, pero también tiene que discriminar fondos como cielo, fondo de ciudad, veredas, ect. Es muy difícil encontrar un equilibrio ya que un algoritmo poco “permisivo” ignorara muchas instancias de objetos y por el otro extremo, se incluirá fondos y de esta manera introducir ruido en nuestro modelo.

En este proyecto, como veremos en el Capítulo 4 se probó con **Edge-Boxes** y **Selective search**, ya que estas generan una cantidad significativamente menor a algoritmos del estilo de ventana deslizante. Aun así, procesar todas estas propuestas es engorroso. Además, estos modelos por lo general dan como resultados muchos cuadros con una gran superposición. Esto da lugar a una técnica denominada Supresión no máxima (NMS) 2.2b. Este algoritmo necesita de un puntaje que

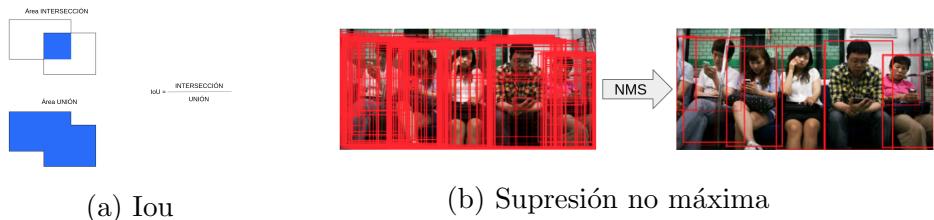


Figura 2.2: (a) Calculo de Intersección sobre Unión. (b) Salida de la propuesta de objetos y el resultado después de NMS.

indica la confianza del cuadro delimitador y un criterio para comparar entre distintos cuadros. El criterio mas común es Intersección sobre Unión (IoU), en la imagen 2.2a se muestra como se calcula sobre dos Bounding Boxes. La salida de NMS es un conjunto mas reducido de propuestas, en la cual se filtraron todas las que se consideran repetidas y retorna solo las mas representativa. A continuación se muestra el pseudocódigo de NMS.

```

1: procedure NMS(B, S, t)
2:   D =  $\emptyset$ 
3:   for  $B \neq \emptyset$  do
4:      $b_i = \text{SelPropuestaMaxPuntaje}(B, S)$ 
5:     Eliminar( $b_i$ , B)
6:     for  $b_j \in B$  do
7:       if  $\text{IOU}(b_i, b_j) > t$  then
8:         Eliminar( $b_j$ , B)
9:       end if
10:      end for
11:    end for
12:    return D
13: end procedure

```

2.4. Multimodales

Nuestra experiencia del mundo es multimodal, vemos objetos y sus entornos, escuchamos música y ruidos, sentimos la textura de los distintos materiales, olemos los aromas que no rodean y probamos el sabor de un buen plato de comida. Inconscientemente asociamos una situación a los distintos estímulos que recibimos en ese momento y los relacionamos entre si, para generar una idea de lo que está sucediendo. De esta manera, sabemos que si algo huele mal lo más probable que sepa igual y podemos relacionar una imagen del campo con el sonido de los pájaros.

La modalidad se refiere a la forma en que algo sucede o se experimenta. Un problema de investigación se caracteriza como multimodal cuando incluye datos de distinta naturaleza. Para que la inteligencia artificial avance en la comprensión del mundo que nos rodea, necesita poder interpretar y relacionar estas distintas señales. Aunque la combinación de diferentes modalidades para mejorar el rendimiento parece una tarea intuitivamente atractiva, en la práctica, es un desafío, y una ineducada combinación de distintos tipos de información, puede generar ruido y conflictos. La idea general es, partir de dos objetos matemáticos distintos uno de cada modal y poder transformar a ambos para lograr que pertenezcan a un tercer objeto que es la representación multimodal. Las imágenes suelen estar asociadas con etiquetas y explicaciones de texto. En este trabajo nos aprovechamos esto y tratamos de encontrar un espacio común entre el vector que representan a la imagen del objeto y el que representa la sintaxis del mismo.

2.5. Aprendizaje por disparo cero (ZSL)

Es un conjunto de problemas de aprendizaje automático, donde en el momento de la prueba, se observan muestras de clases que no se observaron durante el entrenamiento y se necesita predecir la categoría a

la que pertenecen. A diferencia de las configuraciones estándares en el aprendizaje automático, donde se espera que se clasifiquen correctamente las nuevas muestras en las clases que ya han observado durante el entrenamiento, en ZSL, no se han proporcionado muestras de las clases durante el entrenamiento del clasificador. Podemos diferenciar dos tipos de clases, las vistas que están presente en el entrenamiento y las invisibles o novedosas que no estuvieron. Naturalmente, se debe proporcionar algún tipo de información complementaria sobre estas clases invisibles, este tipo de dato puede ser variado. Una descripción estructurada predefinida, que al tenerla en cuenta mejora el aprendizaje. Una descripción textual, aquí las clases van acompañadas de un comentario en lenguaje natural. Esto podría incluir, por ejemplo, una descripción de wikipedia. Por ultimo, tanto las clases visibles como las invisibles están relacionadas en un espacio vectorial, donde el conocimiento de las clases vistas se puede transferir a clases invisibles.

El problema de aprendizaje por disparo cero se puede dividir en categorías según los datos presentes durante la fase de entrenamiento y la fase de prueba.

Con base en los datos disponibles en el momento de entrenar un modelo

- **Aprendizaje por disparo cero inductivo:** Se tiene acceso a datos de imágenes etiquetadas de las clases vistas.
- **Aprendizaje por disparo cero transductivo:** Además de los datos de imagen etiquetados de las clases vistas, también se tiene acceso a las imágenes no etiquetadas de las clases no vistas.

Basado en los datos disponibles en el momento de la inferencia

- **Aprendizaje por disparo cero convencional (ZSL):** En las pruebas solo se evalúan las clases no vistas.
- **Aprendizaje por disparo cero generalizado (GZSL):** En las pruebas se evalúan tanto las clases vista como las no vistas.

2.6. Detección de objeto por disparo cero (ZSD)

La Detección de objeto por disparo cero (ZSD), tiene como objetivo reconocer y localizar simultáneamente instancias de objetos que pertenecen a categorías novedosas sin ningún ejemplo de entrenamiento. Como estas categorías no están presente en entrenamiento resulta imposible, tener alguna información sobre su aspecto visual, lo cual no nos permite detectarlas ni reconocerlas. Es necesario encontrar algún dominio que tenga la capacidad de guardar la información de todas las clases, para luego relacionarlas con el aspecto visual de las categorías novedosas.

En este trabajo, proponemos un modelo de disparo cero inductivo, es decir, solo observamos imágenes de clases vistas y etiquetas que indican a que clase pertenece. Estas etiquetas son palabras del lenguaje natural sin ninguna estructura. Luego se puede inferir todas las clases o solo las invisibles, dependiendo de lo que se quiere evaluar, aprendizaje por disparo cero generalizado o convencional.

Para formalizar, denotamos las clases como $\mathcal{C} = \mathcal{S} \cup \mathcal{U}$, donde \mathcal{S} son las clases vistas para entrenamiento y \mathcal{U} las clases no vistas, utilizadas en la etapa de pruebas. Ademas se tiene que $\mathcal{S} \cap \mathcal{U} = \emptyset$. Aunque no es necesario definir el conjunto de clases de pruebas, ya que el modelo tiene que ser capaz de detectar tanto clases vista como las no vista, se hace para poder tener una evaluación cuantitativa.

Denotamos a una imagen como $\mathcal{I} \in \mathbb{D}^{\mathcal{M} \times \mathcal{N} \times 3}$. Donde $\mathbb{D} = \{0, \dots, 255\}$, \mathcal{M} es el largo de la imagen, \mathcal{N} el ancho. Esta es la forma en la que se representa cada pixel de la imagen en el formato **RGB**, donde se tiene 3 canales que caracterizan la intensidad de los colores rojo, verde y azul. Por cada imagen se provee un conjunto de cuadros delimitadores $\mathbb{B} = \{b_0, \dots, b_k \mid b_i \in N^4\}$ y sus etiquetas asociadas como $\mathbb{Y} = \{y_0, \dots, y_k \mid y_i \in \mathcal{C}\}$. Para cada cuadro delimitador b_i extraemos

una característica profunda utilizando una red neuronal convolucional denotada como $\phi(b_i) \in \mathbb{R}^{D_1}$. Denotamos las incrustaciones semánticas $w_j \in \mathbb{R}^{D_2}$ obtenido por algún modelo de **Word embedding** para cada etiqueta y_i . El conjunto de todas las imágenes de entrenamiento se indica con \mathcal{X}^s , que contiene ejemplos de todas las clases de objetos visibles. El conjunto de todas las imágenes de prueba que contienen muestras de clases de objetos invisibles se indica con \mathcal{X}^u . En particular, no hay ningún objeto de clase invisible en \mathcal{X}^s , pero \mathcal{X}^u puede contener objetos vistos.

El objetivo es encontrar una matriz de proyección W_p , tal que

$$\psi_i = W_p \phi(b_i) \quad | \quad W_P \in \mathbb{R}^{D_2 \times D_1}, \quad \psi_i \in \mathbb{R}^{D_2}$$

Notar que ψ_i y las incrustaciones semánticas se encuentran en el mismo dominio. Como mencionamos en secciones anteriores, el espacio vectorial semántico, tiene una gran capacidad de capturar similitudes semántica. Por lo cual resulta clave encontrar una matriz que para cada cuadro delimitador se proyecte lo mas cerca posible a la incrustación semántica de su clase. En otras palabras se quiere una función $f : \mathcal{X} \rightarrow \{y_0, \dots, y_k \mid y_i \in \mathcal{C}\}$ con $\mathcal{X} = \mathcal{X}^s \cup \mathcal{X}^u$ que da el riesgo empírico regularizado mínimo \mathcal{R} de la siguiente manera:

$$\arg \min_{f \in F} \mathcal{R}(f(x, W_p))$$

donde, $x \in \mathcal{X}^s$ durante el entrenamiento. La función de mapeo utilizado en las pruebas, tiene la siguiente forma

$$f(x, W_p) = \arg \max_{y \in \mathcal{C}} \max_{b \in \mathbb{B}(x)} (F(x, y, b, W_p))$$

donde $\mathbb{B}(x)$ son las propuestas de la imagen x . Intuitivamente, se busca los cuadros delimitadores de mejor puntuación y se les asigna la categoría de objeto de puntuación máxima.

El estricto requisito de no utilizar ninguna imagen de clase invisible durante el entrenamiento en si mismo es una condición difícil. Pero

existen otras dificultades de la tarea de detección de disparo cero que están asociado al conjunto de datos de entrenamiento y prueba, es decir entre las clases vistas e invisibles, que lo dificultan mas.

- **Rareza:** los conjuntos de datos por lo general, contiene un problema de distribución, es decir, muchas clases raras obtienen menos cantidad de instancias. Este problema hace que las clases con mayor cantidad de instancias sesguen el modelo y en las pruebas las clases mas raras y las relacionadas a ella sean marcadas incorrectamente. Es evidente que una clase invisible debería estar dentro del conjunto de clases raras, pero esto implica introducir información extra que en la vida real no es posible. Esto es un problema al tiempo de comparar dos modelos que fueron entrenados con distintas clases ya que algunas separaciones resultan mejores que otras.
- **Tamaño del objeto:** algunas clases de objetos raros como tijeras, lapicera, celulares, etc., suelen tener un tamaño pequeño. Los objetos más pequeños son difíciles de detectar y reconocer. También, tienen el problema que por lo general están juntos a objetos mas grandes como una mesa o persona y se ven opacadas por estas clases.
- **Alta diversidad:** cada clase pertenece a un conjunto donde se encuentran todas las clases similares o que tienen alguna relación, que llamaremos metaclass. Cada metaclass recibe un número diferente de clases y existe una gran diversidad visual en las imágenes de cada metaclass o clase superior. Dado que estar en una misma metaclass no garantiza la similitud visual, es difícil aprender relaciones para las categorías invisibles que son bastante diferentes de las categorías vistas en la misma metaclass. Por ejemplo, “auto” tiene muchas clases similares en comparación con “cartel”. Esto permite una descripción inadecuada de la clase invisible que eventualmente afectará el rendimiento de detección de disparo cero.

- **Ruido en el espacio semántico:** cuando se utiliza los vectores de incrustación semántica no supervisados como word2vec o GloVe. Las incrustaciones resultante en general son ruidosas, ya que se generan automáticamente a partir de la minería de texto no anotado. Esto también afecta significativamente el rendimiento de la detección de disparo cero.

Existen algunas variaciones del problema de ZSD, que intentan aliviar estos problemas, siendo un poco mas permisivos. Como **Detección de metaclas de disparo cero (ZSMD)**, que dada una imagen de prueba, el objetivo es localizar cada instancia de una clase de objeto invisible y categorizarla en una de las superclases. **Etiquetado de disparo cero (ZST)**, para reconocer una o más clases invisibles en una imagen de prueba, sin identificar su ubicación. **Etiquetado de metaclas de disparo cero (ZSMT)**, para reconocer una o más metaclasses en una imagen de prueba, sin identificar su ubicación. Pero estas tareas son buenas para calcular resultados cuantitativos y no para ser utilizados en problemas reales.

Capítulo 3

Diseño y Arquitectura

3.1. Arquitectura

Como ya mencionamos anteriormente decidimos basarnos en el modelo propuesto por Ankan Bansal [2]. Su arquitectura propuesta se puede dividir en las siguientes etapas.

- **Pre-procesamiento:** Por cada imagen de entrenamiento, se extrae por cada cuadro delimitador una característica profunda utilizando una red neuronal convolucional y se asocia con el vector semantico a la clase que corresponde dicho cuadro, que se puede obtener de modelos de incrustación de palabras previamente entrenados como Glove o Word2vec. Esta etapa nos genera como salida dos listas $X = [\phi(b_0), \dots, \phi(b_k) \mid \phi(b_i) \in \mathbb{R}^{D_1}]$ $W = [w_0, \dots, w_k \mid w_i \in \mathbb{R}^{D_2}]$
- **Entrenamiento:** Utilizamos el espacio de incrustación común (\mathbb{R}^{D_2}) para calcular una medida de similitud entre las proyecciones $\phi(b_i)$ y las words embedding w_i . Luego se entrena la proyección usando una pérdida de margen máximo que impone la restricción de que el puntaje de similitud de un cuadro delimitador con su clase verdadera debe ser más alto que el de otras

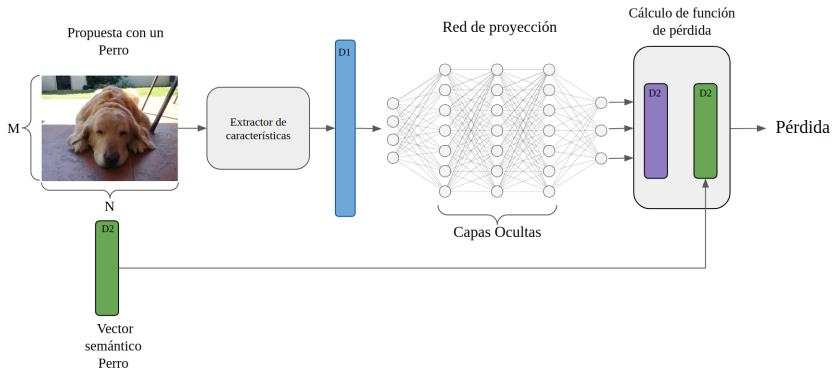


Figura 3.1: Arquitectura propuesta y la dimensión de cada paso.

clases. Para esto se utiliza una función de perdida definida como:

$$\mathcal{L}(\psi_i, w_i) = \sum_{j \in \mathcal{S}, j \neq i} \max(0, m - S_{ii} + S_{ij})$$

m es el margen máximo, y S_{ij} es la similitud entre la proyección i -esima y la incrustación j -esima. En la Figura 3.1, se puede apreciar la arquitectura completa.

También se agrega una función de pérdida de reconstrucción como sugiere Kodirov en [5]. Se utilizan las características del cuadro delimitado proyectada para reconstruir las características profundas originales y calcular la pérdida de reconstrucción como la distancia L_2 entre la característica reconstruida y la característica profunda original.

$$\mathcal{L}_r = \|\phi(b_i) - \psi_i W_p^T\|^2$$

Luego definimos λ como un coeficiente de ponderación que controla la importancia del primer y segundo término, que corresponden a las pérdidas de proyección y reconstrucción, respectivamente. Por lo cual la función de perdida total es:

$$\mathcal{L}_t = \lambda \mathcal{L} + (1 - \lambda) \mathcal{L}_r$$

- **Evaluación:** Por cada imagen del conjunto de entrenamiento, se genera un conjunto de propuestas de cuadros delimitadores. Luego, se eliminan todos lo que no tienen un puntaje de confianza mayor a un umbral. Para cada cuadro se computa la característica profunda $\phi(b_i)$ y utilizando la matriz W_p , para predecir las características semánticas. Por ultimo, se calcula la similitud con todas las características semánticas, asignando al nuevo cuadro delimitador la que tenga mayor puntaje.

Es común que en la detección de objetos incluyan una clase de fondo para aprender un detector robusto que pueda discriminar eficazmente entre objetos de primer plano y objetos de fondo. En ZSD, esto no es un problema trivial, ya que no sabemos si un cuadro de fondo incluye elementos de fondo como cielo, tierra, bosque, etc. o una instancia de una clase de objeto invisible. En muchos trabajamos se proponen distintas técnicas para abordar este problema, pero no presentan mejoras en evaluaciones cuantitativas. Es por esto que no se incluye una arquitectura que discrimine cuadros de fondos.

Las propuestas de cuadro delimitadores son claves a la hora de evaluar un modelo, como mencionamos en secciones anteriores. Muchas arquitecturas incluyen una red de propuestas regionales, RPN por sus siglas en inglés. De esta manera entran un modelo completo que también aprende a generar propuestas. Por lo que se pudo investigar, en nuestro caso resulta más conveniente utilizar un modelo pre-entrenado como Edge-Boxes o Selective search.

3.2. Conjuntos de datos

COCO es un conjunto de datos de detección, segmentación y subtítulos de objetos a gran escala. COCO tiene varias características: Segmentación de objetos, Reconocimiento en contexto, Segmentación de material de superpixeles, 330 mil imágenes (> 200 mil etiquetadas), 1,5 millones de instancias de objetos y 80 categorías de objetos.

La gran cantidad de instancias de objetos y de categorías, resulta en un conjunto ideal para entrenar y evaluar modelos de ZSD. Ademas la mayoría de la imágenes consta de una gran cantidad de objetos que generan un contexto y no de uno solo centralizado, como son por ejemplo las imágenes del conjunto Visual Genome Dataset. Utilizamos las imágenes de entrenamiento del conjunto COCO 2014 e imágenes del conjunto de validación para realizar pruebas.

Como COCO no provee una separación de los datos para evaluar modelos de ZSD, es necesario crear una forma de dividirlos. Notar que resulta de suma importancia dividir las clases, de tal manera que para todo objeto del conjunto prueba se encuentre otro similar en entrenamiento. Ademas, no se puede encontrar ningún objeto de prueba en los datos de entrenamiento. Dicho esto, se propuso una manera de separar las imágenes. COCO también tiene agrupada las clases por “Clases superiores”

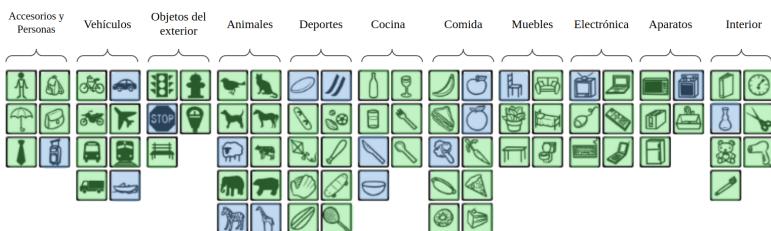


Figura 3.2: Divisional de las clases para entrenamiento (verde) y pruebas (azul).

Por cada “Clase superior”, elegimos de forma aleatoria un 70 % de clases para entrenamiento y un 30 % para pruebas. Es decir 47 y 18 clases respectivamente. En la Figura 3.2 se puede ver como quedan divididas. Por ultimo se eliminaron todas las imágenes de entrenamiento que contengan al menos una instancia de las clases de prueba. Esto resulta en 42564 imágenes con 261258 instancias de entrenamiento y

3008 con 10878 instancias de prueba. Bansal [2], divide el conjunto de datos, de manera similar, utiliza la misma cantidad de clases para pruebas y entrenamiento. Pero para crear las metaclases utiliza las incrustaciones de vectores de palabras para todas las clases y las agrupa en grupos de 10 usando la similitud de coseno entre los vectores de palabras como métrica. Por los problemas mencionados en la Sección 2.6, también utilizamos esta separación, de esta manera logramos una comparación de modelos mas justa.

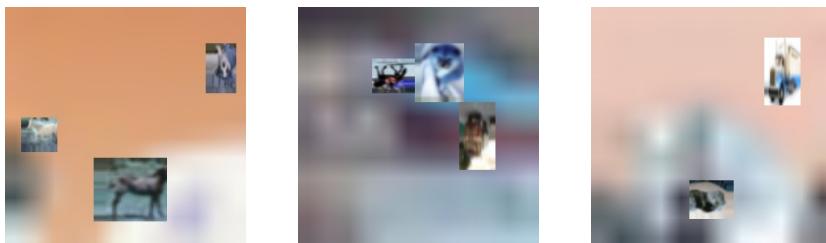


Figura 3.3: Ejemplos de imágenes del conjunto de datos CIFAR-ZSD.

COCO puede resultar pesado lo cual implica mucho tiempo de cómputos, para solucionar esto se creo un conjunto de datos sintéticos basado en CIFAR-100 datasets, el cual denominamos CIFAR-ZSD. Consta de imágenes localizadas, rotadas y re-escalada aleatoriamente con un fondo de otra imagen. Con esto se intenta simular imágenes reales en la cual un objeto puede aparecer con distintos aspectos y escalas. Este conjunto esta dividido para que ninguna instancia de prueba aparezca en el conjunto de entrenamiento.

Aunque resulta muy útil para probar modelos, no es bueno para reportar métricas reales. Pero en combinación con COCO, que si lo es, ambos cooperan para enfrentar el problema de ZSD de una forma mas practica.

3.3. Detalles de la implementación

El paper de Bansal [2], carece de una implementación de acceso publico, por lo cual se realizo una implementación propia, basándose en los detalles que se pueden extraer del documento publicado. Fue necesario algunas presunciones, pero también, nos otorgo flexibilidades a la hora de codificar. Se busco obtener los resultados mas cerca a los reportados, pero siendo lo mas fiel a la información disponible. Para esto se decidió utilizar Python 3 con el framework Keras ejecutandose sobre TensorFlow. Si se quiere acceder a mas detalles, el código se encuentra en <https://github.com/agustinhurquiza/Tesis>.

Primero se realiza un preprocesamiento a todas las imagines del conjunto de datos de entrenamiento, que consiste en generar propuestas de objetos utilizando **Edge Boxes** o **Selective Search**. En el Capítulo 4 se muestra los resultados de usar cada algoritmo. Luego por cada propuesta se calcula la intersección sobre unión con todos los cuadros delimitadores verdaderos. Si el $\text{IoU} > 0,5$ con algún cuadro verdadero, se guarda la propuesta y se la asocia con la clase del cuadro delimitador verdadero. Este preprocesamiento aumenta significativamente la cantidad de instancias para la etapa de entrenamiento. Resulta en un total de 1.436.835 instancias para COCO y 137.204 para CIFAR-ZSD. El siguiente paso consiste en generar el vector de características visuales y semánticas de cada cuadro delimitador. Como las CNN, utilizan un tamaño de entrada fijo, es necesario rescalar todos los cuadros. Para **VGG16** utilizamos 224×224 y 299×299 en **Inception ResNet V2**, que son las dimensiones por defecto. El tamaño del vector de salida de cada red es de 512 en VGG y 1536 ResNet. Para ambas usamos pesos preentrenado en **Imagenet**. Este paso es diferente a como se hace en el paper de Bansal [2], ya que en este trabajo el modelo se entrena de extremo a extremo, es decir los pesos de la CNN se ajustan en la etapa de entrenamiento. Para obtener las características semánticas, utilizamos **Word2Vec** previamente entrenado de Google. Que Incluye vectores para un vocabulario de 3

millones de palabras y frases que se entreno en aproximadamente 100 mil millones de palabras de un conjunto de datos de Google News. La longitud del vector resultante es de 300. Luego se guardan dos archivos por cada imagen, en uno se encuentran los vectores visuales de cada clase y en el otro los semánticos.

Por ultimo, para el entrenamiento, se creo una red con una sola capa oculta, una capa de entrada del tamaño del vector de características visuales y una de salida de la dimensión de las características semánticas. Lo cual resulta en 153.900 parámetros entrenables con **VGG16** y 461.100 con **Inception ResNet V2**. Como optimizador se utilizo Adam, sin ninguna activación, un taza de aprendizaje de 10e-3 y un tamaño del lote de 64 muestras. Para la función de perdida se uso un lambda de 10e-3 y un margen máximo de 1.

Capítulo 4

Experimentos

4.1. Experimentación con Propuestas de objetos

Como se menciono anteriormente, el numero de propuestas es un parámetro clave. Algunas métricas son muy sensible a la cantidad de propuestas, afectando los resultados finales. Esto surgió, cuando se obtuvieron las primeras métricas, los valores estaban muy lejos de los esperados, y a medida que se aumentaba la cantidad de propuesta, los resultados empeoraban. Por este motivo se probaron dos algoritmos (**Edge Boxes** y **Selective Search**) con algunas combinaciones de sus parámetros. Con el objetivo de obtener una cantidad de propues-

Algoritmo	Edge Boxes				Selective Search	
	-				Single	Fast
Número de propuestas	100	500	1000	5000	≈ 5000	≈ 1000
Tiempo promedio (s)	0,11	0,11	0,12	0,12	5,48	1,41
Propuestas Totales	4.415.244	22.050.071	43.802.935	161.809.194	350.535.591	95.643.172
Propuestas con IOU > 0,5	86.233	133.942	155.584	194.891	221.551	203.563

Cuadro 4.1: Resultados de correr los distintos algoritmos de propuestas de regiones en los datos de entrenamiento. El numero de propuestas verdaderas es 261.258.

tas que se superponga con el mayor numero de objetos sin afectar las métricas.

Para no sesgar al experimento con los datos de prueba, se definió la metodología de la siguiente manera. Por cada imagen de entrenamiento se corrió el generador de propuestas, se calculó el tiempo y la cantidad de cuadros verdaderos que tenían un IoU > 0,5, con algún cuadro verdadero. El tiempo es un parámetro importante ya que algunos algoritmos son muy lentos y resulta imposible usarlos. Como se puede observar en el Cuadro 4.1, **Selective Search** obtiene una mayor cantidad de superposición, pero con un número exageradamente grande de propuestas. La mejor opción es usar **Edge-boxes**. En cuanto número de propuestas totales resulta más conveniente entre 100 y 500 propuestas como máximo, ya que al aumentar este número no se generan mejoras en superposición pero si aumenta el número de propuestas. Si tenemos en cuenta el tiempo, resulta mejor **Edge-boxes**, ya que demora una fracción de lo que tarda **Selective Search**.

4.2. Experimentación con CNN

Se decidió analizar la CNN ya que el modelo final es muy dependiente de esta red y su capacidad de extraer características visuales. Lo que se quiere aquí es que la red sea capaz de asociar las características visuales de objetos similares, y diferenciar los elementos de distinta naturaleza. En otras palabras, el espacio resultante tiene que distribuirse de tal manera que por ejemplo las imágenes de los animales estén muy cerca y a su vez alejado de vehículos o electrodomésticos, pero también tiene que mantener una separación entre los distintos animales como perro y gato. Bansal en su trabajo, propone utilizar **Inception ResNet V2**, pero esta red resulta muy pesada en cuanto a tiempo de ejecución y memoria. Por este motivo se decidió intentar con **VGG16**, que reduce el número de parámetros en las ca-

pas convolucionales y mejorar el tiempo de ejecución, ademas es una de la mas utilizada.

El experimento consistió en comparo miles de recuadros de 3 clases de entrenamiento, caballo, perro y camión. Por cada cuadro se genero el vector de caracteristicas visuales. Luego se comparo utilizando la similitud coseno, entre todas las caracteristicas de caballo vs caballo, caballo vs camión y caballo vs perro. Se graficaron (Figura 4.1) las frecuencias de los resultados para cada CNN. Con esto se intenta observar como se distribuyen en el espacio visual, las distintas clases. Como se esperaba la similitud entre entre animales es mas grande que con un vehiculo. Pero, se observo que para **Inception ResNet V2** existe una mayor separación entre clases, aunque sus similitudes están mas dispersas. **VGG16**, parece tener una menor dispersión, pero la similitud coseno entre distintas clases tiene valores muy cercanos. Esto puede afectar de manera negativa ya que camión y caballo no poseen una gran diferencia y el modelo podría interpretarlo como clases similares.

4.3. Definición de métricas

Entre los diferentes conjuntos de datos anotados utilizados por los desafíos de detección de objetos y la comunidad científica, la métrica más común utilizada para medir la precisión de las detecciones es el **Mean Average Precision (mAP)**, seguida por **Recall**. Un problema que tienen la métricas en detección de objetos, es la falta de una implementación estándar para calcularlas. Ademas, aquellas implementaciones publicas, están muy encapsuladas al código y resulta muy difícil adaptarlo, para medir rendimientos de modelos propios. Como ya se menciono, el código de Bansal [2] no esta disponible, por este motivo fue necesario encontrar alguna implementación de estas métricas. Se encontraron varias y luego de hacer cambios para utili-

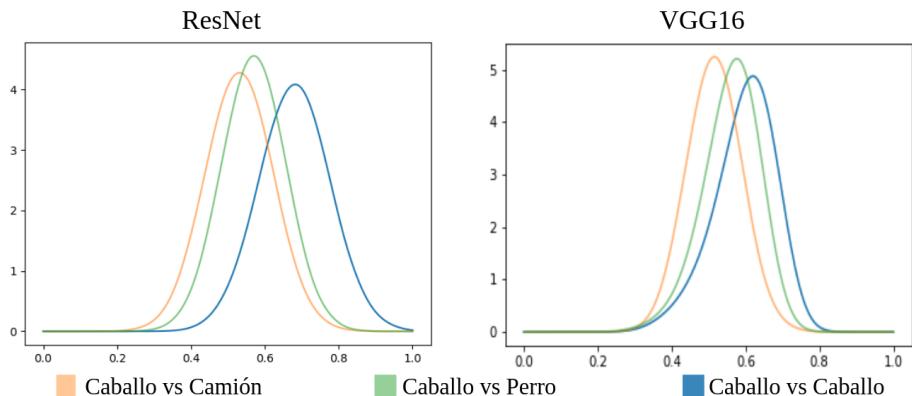


Figura 4.1: Frecuencia de la similitud coseno de los vectores de características visuales, entre la misma y distintas clases, para las CNN **Inception ResNet V2** y **VGG16**.

zarlos, los resultados variaban mucho de un código a otro. Fue así que se encontró el trabajo de Padilla Rafael [9], que explica lo planteado:

“La falta de consenso en diferentes trabajos e implementaciones de AP es un problema al que se enfrentan las comunidades académicas y científicas. Las implementaciones métricas escritas en diferentes lenguajes y plataformas computacionales generalmente se distribuyen con los conjuntos de datos correspondientes que comparten una descripción determinada del cuadro delimitador. De hecho, estos proyectos ayudan a la comunidad con las herramientas de evaluación, pero exigen trabajo adicional para adaptarse a otros conjuntos de datos y formatos de cuadro delimitador.”

Ademas Padilla, propone una definición y un código para estandarizar las métricas, de esta manera se pueden comprar distintos modelos, de una forma “justa”. Por estos motivos decidimos utilizar este trabajo, aunque los resultados de nuestros modelos, no sean exactos

a los reportados por Bansal [2].

Ahora definamos las métricas, basándonos en el trabajo [9]. Primero es necesario, estandarizar cuando un cuadro es:

- Falso negativo (**FN**): No se obtuvo ninguna detección en absoluto, o para un cuadro delimitador verdadero el $\text{IoU} > 0,5$ y no se predijo correctamente la clase
- Falso positivo (**FP**): Para un cuadro delimitador verdadero, se predijo correctamente la clase pero el $\text{IoU} < 0,5$, o es una predicción duplicada, es decir, ya se marcó otra con mayor IOU como **TP**.
- Verdadero positivo (**TP**): Para un cuadro delimitador verdadero, se obtuvo una propuesta con un $\text{IoU} > 0,5$ y se predijo correctamente la clase.
- Verdadero negativo (**TN**): Esto solo tiene sentido si, se quisiera medir propuestas que no tenían un $\text{IoU} > 0,5$ con todos los cuadros verdaderos, y además se predijo como clase de fondo. Pero en este trabajo no es utilizada.

La **Recall**, también conocida como sensibilidad, mide la probabilidad de que los objetos verdaderos (los que se encuentran en la imagen) se detecten correctamente, viene dado por:

$$\text{Recall} = \frac{\text{TP}}{\text{FN} + \text{TP}}$$

El trabajo de Bansal, define Recall de la siguiente manera:

“Un cuadro delimitador predicho se marca como verdadero positivo solo si tiene una superposición de IoU mayor que un cierto umbral t con un cuadro delimitador de verdad del terreno y no se ha asignado ningún otro cuadro delimitador de mayor confianza al mismo cuadro de verdad del terreno. De lo contrario, se marca como falso positivo.”

Según lo que se puede interpretar, utiliza los falsos positivos para calcular la **recall** en vez de usar Falso negativo. Sin poner en tela de juicio, si esto esta bien o mal, es claro que de esta forma solo se tiene en cuenta, los objetos que tuvieron al menos una propuesta con un $\text{IoU} > 0,5$ y el resto, quedan fuera del calculo de esta métrica. Esto genera una diferencia enorme en los resultados y dificulta la tarea de comparar con otros modelos, es por esto que en este trabajo reportamos ambas. Bansal ademas, calcula una variación denominada K@Recall, donde solo se tienen en cuenta las K mejores propuestas basándose en la confianza de la predicción y el resto son descartadas.

$$\textit{Precision} = \frac{TP}{FP + TP} \quad (4.1)$$

AP, es una métrica popular para evaluar la precisión de los detectores de objetos mediante la estimación del área bajo la curva (AUC) de la relación **precisión** 4.1 x **recall**. La curva de **precisión** x **recall** puede verse como una compensación entre ambas métricas para diferentes valores de confianza asociados a los cuadros delimitadores generados por un detector. Si la confianza de un detector es tal que su FP es bajo, la precisión será alta. Sin embargo, en este caso, se pueden pasar por alto muchos aspectos positivos, lo que produce un FN alto y, por lo tanto, una recall baja. Por el contrario, si uno acepta más positivos, el recuerdo aumentará, pero el FP también puede aumentar, disminuyendo la precisión. Sin embargo, un buen detector de objetos debe encontrar todos los objetos reales mientras identifica solo los objetos relevantes . Por lo tanto, un detector de objetos en particular puede considerarse bueno si su precisión permanece alta a medida que aumenta su recuperación, lo que significa que si el umbral de confianza varía, la precisión y la recall seguirán siendo altas. Por lo tanto, un área alta bajo la curva (AUC) tiende a indicar tanto una alta precisión como una alta recuperación. **mAP** para la detección de objetos es el promedio del AP calculado para todas las clases. Por lo general también se indica sobre que IoU se calcula, por ejemplo,

mAP@0.5, o un conjunto de umbrales como mAP@[x, y]. El trabajo de Bansal reporta **mAP**, pero no indica sobre que IoU se calcula, a si que se asume que utilizo un valor de 0,5. Muchos trabajos que utilizan COCO, reportan mAP@[.5, .95]. Esta métrica resulta muy útil si se quiere comparar rendimientos entre distintos trabajos.

4.4. Detalles de metodología de evaluación

El principal experimento consto en replicar los resultados de Bansal [2]. No se realizo exactamente sus experimentos, ya que esto no aportaría nada nuevo. Así que, se decidió analizar sus resultados y solo replicar los que consideramos indispensable y que seria un buen punto de partida. Por ejemplo, los experimentos con clases de fondo, no obtuvieron buenos resultados, en comparación con los que no la utiliza. Es por esto que no consideramos realizarlos. El principal objetivo fue obtener un modelo que obtenga resultados lo mas similar posible a los reportados. Después de varias iteraciones, no se pudo lograr, y el principal motivo es la falta de una implementación para calcular las métricas. Se utiliza el código desarrollado por [9], pero sin ninguna garantía de que las métricas se calculen de la misma manera.

Ahora definamos la metodología de evaluación. El primer paso consiste generar propuestas para cada imagen, luego cada cuadro propuesto es reescalado al tamaño de la capa de entrada que tiene la CNN, y se le extrae el vector de características visuales. Después, se utiliza el modelo entrenado para inferir el vector de características semánticas, y se calcula la similitud coseno con los vectores semánticos de todas las clases o solo las invisibles, dependiendo si se quiere evaluar ZSDG o ZSD. Aquella clase que obtenga el mayor puntaje es asignada a la propuesta. También, se guarda la puntuación como la confianza de predicción. Por ultimo, se agrupan todas las propuestas que se tengan asignada la misma clase y se corre un algoritmo de su-

presión no máxima. NMS elimina las predicciones repetidas y retorna las mejores propuestas de cada grupo. Al final obtenemos como resultado un conjunto de propuestas, sus clases y su respectivo puntaje. Estos datos se guardan en un archivo y luego se corre la implementación de Padilla, para obtener los resultados de las métricas.

Capítulo 5

Análisis de resultados

5.1. Resultados Cuantitativos

5.2. Resultados Cualitativos

Capítulo 6

Conclusiones y trabajo futuro

6.1. Aportes

6.2. Trabajo futuro

Bibliografía

- [1] Yusuf Aytar, Carl Vondrick, and Antonio Torralba. See, hear, and read: Deep aligned representations. *arXiv preprint arXiv:1706.00932*, 2017.
- [2] Ankan Bansal, Karan Sikka, Gaurav Sharma, Rama Chellappa, and Ajay Divakaran. Zero-shot object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 384–400, 2018.
- [3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.
- [4] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.
- [5] Elyor Kodirov, Tao Xiang, and Shaogang Gong. Semantic autoencoder for zero-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3174–3183, 2017.
- [6] David Marr. Vision: A computational investigation into the human representation and processing of visual information. New York, NY: W.H. Freeman and Company, 1982.

- [7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- [8] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [9] Rafael Padilla, Sergio L Netto, and Eduardo AB da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242. IEEE, 2020.
- [10] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.
- [11] Shafin Rahman, Salman Khan, and Fatih Porikli. Zero-shot object detection: Learning to simultaneously recognize and localize novel concepts. In *Asian Conference on Computer Vision*, pages 547–563. Springer, 2018.
- [12] Lawrence G Roberts. *Machine perception of three-dimensional solids*. PhD thesis, Massachusetts Institute of Technology, 1963.
- [13] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.
- [14] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

- [15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.
- [16] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International journal of computer vision*, 104(2):154–171, 2013.
- [17] Yongqin Xian, Christoph Lampert, Bernt Schiele, and Zeynep Akata. Zero-shot learning - a comprehensive evaluation of the good, the bad and the ugly. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP, 07 2017.
- [18] Yuting Zhang, Luyao Yuan, Yijie Guo, Zhiyuan He, I-An Huang, and Honglak Lee. Discriminative bimodal networks for visual localization and detection with natural language queries. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 557–566, 2017.
- [19] Pengkai Zhu, Hanxiao Wang, and Venkatesh Saligrama. Zero shot detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.
- [20] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *European conference on computer vision*, pages 391–405. Springer, 2014.