Instituto Tecnológico y de Estudios Superiores de Occidente

Reconocimiento de validez oficial de estudios de nivel superior según acuerdo secretarial 15018, publicado en el Diario Oficial de la Federación del 29 de noviembre de 1976.

Departamento de Electrónica, Sistemas e Informática

Maestría en Sistemas Computacionales



MODELO DE APRENDIZAJE PROFUNDO PARA LA DETECCIÓN DE OBJETOS DE INTERÉS EN EL BOSQUE DE LA PRIMAVERA

TRABAJO RECEPCIONAL que para obtener el **GRADO** de
MAESTRO EN SISTEMAS COMPUTACIONALES

Presenta: José Luis Magaña Vázquez

Asesor: Dr. José Francisco Cervantes Alvarez
Co-asesor: Mtro. Luis Eduardo Pérez Bernal

Tlaquepaque, Jalisco. 5 de junio de 2022

AGRADECIMIENTOS

Mis sinceros agradecimientos:

Al Dr. Francisco Cervantes y al Mtro. Luis Peréz, asesores durante la maestría. Por su apoyo, asesorías y guía a través del trabajo de grado, sin ellos este trabajo no hubiera sido posible.

Al Dr. Iván Villalón, coordinador de la maestría de Sistemas Computacionales, con el cuál desde la inscripción hasta el cierre de este ciclo siempre me tuvo una cordial respuesta y atención.

Al CONACYT, por la beca otorgada. El apoyo económico ha sido un factor crítico en la decisión de comenzar y llevar a buen término la maestría.

Al ITESO y todas las personas en su plantel, por la oferta académica y al apoyo en todo sentidos, que permitieron que haya realizado la maestría.

A Continental e ITESO por el convenio en apoyo económico.

DEDICATORIA

Dedico este trabajo:

A mis padres, Carmen y Walter, mis logros también son suyos, su constancia y atención son semilla de cada hoy de mi vida.

A mi hermanos Katia y Gerardo, por su cariño y confianza en mi.

Y en especial a mi esposa, amiga y compañera de vida. Tania, a tu lado cualquier día es mejor.

Índice general

Índice general **3**

Índice de figuras **7**

Índice de tablas **9**

1. INTRODUCCIÓN **12**

1.1. Antecedentes 13

1.2. Justificación 14

1.3. Problema 14

1.4. Objetivos 15

1.4.1. Objetivo general 15

1.4.2. Objetivo específico 15

1.5. Organización del documento 15

2. ESTADO DEL ARTE **16**

2.1. Introducción 17

2.2. Modelos de detección de objetos genéricos 17

2.2.1. Clasificación 18

2.2.2. Localización 18

2.3. Detectores 19

2.3.1. Detectores de dos etapas . . . 19
2.3.2. Detectores de una etapa . . . 20
2.4. Trabajos Relacionados . . . 20
2.4.1. Detección de copas de árboles individuales . . . 20
2.4.2. Clasificación automática de árboles . . . 21

3. MARCO TEÓRICO 22

3.1. Aprendiendo a partir de datos . . . 23
3.2. Introducción de redes neuronales . . . 23
3.2.1. Perceptrón . . . 24
3.2.2. Red Neuronal . . . 25
3.2.2.1. Algoritmo de aprendizaje . . . 26
3.2.3. Función de costo . . . 27
3.2.3.1. Entropía cruzada binaria . . . 27
3.2.3.2. Entropía cruzada categórica . . . 28
3.2.3.3. Smooth L1 . . . 28
3.2.4. Optimizadores . . . 28
3.2.4.1. Gradiente descendiente estocástico . . . 29
3.2.4.2. Adam . . . 29
3.2.5. Evaluación de Modelo . . . 29
3.2.5.1. Precisión . . . 29
3.2.5.2. mAP . . . 30
3.2.6. Redes neuronales convolucionales . . . 30
3.3. Detector de objetos . . . 31
3.3.1. Faster R-CNN . . . 31

3.3.2. Primera etapa: Propuesta de regiones 33
3.3.2.1. Mapa de características 33
3.3.2.2. Red de propuesta de regiones RPN 33
3.3.2.3. Generación de marcos ancla 35
3.3.2.4. Función de costo de la red RPN 38
3.3.3. Segunda etapa: Red de Detección 38
3.3.3.1. Acondicionamiento de las propuestas para la segunda red . 38
3.3.3.2. Supresión no máxima o NMS 39
3.3.3.3. Muestreo de regiones de interés 40
3.3.3.4. Fast R-CNN 40
3.3.3.5. Función de costo de la red Fast R-CNN 41

4. DESARROLLO METODOLÓGICO 42

4.1. Introducción 42
4.2. Adquisición y preprocesamiento de datos 42
4.2.1. Adquisición de los datos 42
4.2.2. Datos Obtenidos 43
4.2.3. Preparación de datos 45
4.2.3.1. Etiquetado de imágenes 45
4.2.3.2. Preprocesamiento de imágenes y etiquetas 47
4.3. Modelos 49
4.3.1. Faster R-CNN 49
4.3.1.1. Primera etapa: Generador de objetivos para RPN 49
4.3.1.2. Primera etapa: RPN 49
4.3.1.3. Segunda etapa: Supresión no máxima y muestreo de regiones de interés 51

4.3.1.4. Segunda etapa: Red de detección 51
4.3.1.5. Entrenamiento del modelo 51
4.3.2. Detección de árboles 54

5. RESULTADOS Y DISCUSIÓN **57**
5.1. Resultados del modelo 57
5.2. Discusión 58

6. CONCLUSIONES **60**
6.1. Conclusiones 61
6.2. Trabajo Futuro 61

BIBLIOGRAFÍA **63**

A. Apéndice A **66**

Índice de figuras

1.1. Zona de invasiones en BLP. Fuente [3] 13

2.1. Resultado de los ganadores en ILSVRC en la tarea de detección [8] 18

3.1. Modelo del perceptrón 24
3.2. Modelo de una red neuronal de una sola capa 25
3.3. Modelo de una red neuronal con una capa oculta 26
3.4. Problema lineal(izquierda), Problema no lineal(derecha) 26
3.5. Modelo red neuronal LeNet-5[28] 31
3.6. Modelo Faster R-CNN[11] 32
3.7. Extracción de mapa de características 33
3.8. Objeto de interés abarcando la mayor cantidad de la imagen 34
3.9. Objeto de interés abarcando una pequeña cantidad de la imagen 34
3.10. Marcos ancla 35
3.11. Red de propuesta de regiones 36
3.12. Supresión no máxima 39
3.13. Muestreo 40
3.14. Fast RCNN 41

4.1. Diagrama de metodología 42

4.2. Polígono del bosque La Primavera. Fuente Google Earth 43
4.3. Foto aérea tomada por vehículo aéreo no tripulado en bosque de la primavera. 44
4.4. Estructura de carpetas de los datos obtenidos 44
4.5. Visualización de la herramienta Supervisely 46
4.6. Redimensión de imágenes 48
4.7. Imágenes resultantes 48
4.8. Diagrama de implementación del modelo Faster R-CNN 50
4.9. Leyendas para las gráficas 52
4.10. Pérdida de clasificación en RPN durante el entrenamiento 53
4.11. Pérdida de clasificación en la detección durante el entrenamiento 53
4.12. Pérdida del modelo en la detección durante el entrenamiento 54
4.13. Precisión en la detección durante el entrenamiento 54
4.14. Detección de árboles separados 55
4.15. Detección en una zona densa de vegetación 55
4.16. Error en la detección 56
4.17. Detección en una imagen extraida de Google Maps 56

5.1. Predicción de modelos. 58

Índice de tablas

2.1. Tabla de modelos de 2 o más etapas. Bounding Box (BB), Semantic Segmentation (SS) 20

2.2. Tabla de modelos de 1 etapa. Bounding Box (BB), Pair Keypoints (PK) . 21

4.1. Redes usadas para la extracción de características 49

5.1. Resultados del entrenamiento con diferentes redes de extracción de características 57

5.2. Resultados del en imágenes de validación con diferentes redes de extracción de características 57

5.3. Parámetros y tamaño de los modelos 58

LISTA DE ACRÓNIMOS Y ABREVIACIONES

BB Bounding Box. 9, 20, 21

BLP Bosque La Primavera. 7, 13

COCO Common Objects in Context. 20, 46

DPM Region-based Convolutional Neural Network. 17

GPS Global Positioning System. 43

GPU Graphics Processing Unit. 18

HOG Histogram of Oriented Gradients. 17

ILSVRC ImageNet Large Scale Visual Recognition Challenge. 7, 17, 18, 61

IOU Intersection Over Union. 30, 36, 37

ITESO Instituto Tecnológico y de Estudios Superiores de Occidente. 13, 14

JPG Joint Photographic Group. 43

LIDAR Light Detection and Ranging. 21

mAP mean Average Presicion. 17, 18, 30

MP Mega Pixeles. 43, 48

NDVI Normalized Difference Vegetation Index. 13

NMS Non Maximum Suppression. 39, 40

PK Pair Keypoints. 9, 21

R-CNN Region-based Convolutional Neural Network. 19, 61

R-FCN Region-based Fully Convolutional Networks. 19

RGB Red Green Blue. 20, 21, 33, 43

ROI Region Of Interest. 39

RPN Region Proposal Network. 31, 33, 36–38, 40, 41

SGD Stochastic Gradient Descent. 29

SS Semantic Segmentation. 9, 20

SSD Single Shot Detector. 20

UAV Unmaned Aerial Vehicle. 13, 14, 47, 48, 61

VGG Visual Geometry Group. 33

VOC Visual Object Classes. 19, 46

YOLO You Only Look Once. 20

1. INTRODUCCIÓN

Resumen: *El Bosque la Primavera es uno de los ecosistemas más importantes en Guadalara, México. Sin embargo, en las últimas décadas, el ecosistema del bosque se ha degradado de forma preocupante debido a incendios, invasiones y al desarrollo urbano, entre otras causas. Por lo anterior, es importante monitorear de forma permanente el estado de salud del ecosistema. Sin embargo, realizar una inspección manual es un proceso lento y costoso. Por esto, con el fin de reducir costos y tiempo, en este trabajo de obtención de grado se propone el desarrollo de modelos de aprendizaje profundo para el monitoreo automático del Bosque de la Primavera.*

1.1. Antecedentes

Uno de los ecosistemas más relevantes de la zona metropolitana de Guadalajara es el Bosque La Primavera, el cual se decretó zona de protección forestal y refugio de la fauna silvestre en 1980 [1]. La superficie aproximada del bosque es de 30,500 hectáreas dentro de los municipios de Tlajomulco de Zuñiga, Tala y Zapopan en el estado de Jalisco.

En Jalisco, cerca del 77 por ciento de su territorio con vegetación presenta un grado de degradación[2]; entre las causas se encuentra la deforestación, el cambio de uso de suelo (Figura 1.1), los incendios, pérdida del hábitat natural, especies invasivas, enfermedades y plagas. Dada la problemática actual del bosque, es necesario plantearse ¿qué soluciones o acciones se pueden realizar para mitigar o disminuir las causas que degradan el estado de este ecosistema?



Figura 1.1: Zona de invasiones en BLP. Fuente [3]

En el Instituto Tecnológico y de Estudios Superiores de Occidente (ITESO), en 2017, se inició un proyecto cuyo propósito es el desarrollo de un sistema de información para de controlar las misiones de un vehículo autónomo no tripulado (UAV por sus siglas en inglés), así como la recepción y procesamiento de diferentes fuentes de información obtenidas mediante una red de sensores y las cámaras de por estos mismos vehículos. Las misiones de vuelo tienen objetivos específicos tales como: generar orto fotos, calcular un índice de vegetación normalizado (NDVI por sus siglas en inglés), monitorear la calidad del aire, examinar la salud de los árboles, auxiliar en la prevención de desastres naturales mediante la generación de datos que permitan tomar decisiones de forma oportuna.

El desarrollo del presente trabajo de obtención de grado se realiza como parte del proyecto

del ITESO en el Bosque de la Primavera, y representa una aportación en el desarrollo de sistemas de monitoreo automático para la supervisión del estado de salud de la flora en el bosque.

1.2. Justificación

El Bosque la Primavera constantemente es afectado por factores naturales y humanos, los cuales tienen efectos negativos en la vegetación, la fauna, el suelo y caminos naturales; efectos que ponen en riesgo el estado del bosque[3].

Una de las amenazas principales para el bosque son los incendios que se presentan cada año durante la época de estiaje, de manera natural o inducida. Dado que el fuego no se genera por la acción de un factor aislado, sino que se deriva de la acción conjunta de un grupo de factores, es necesario realizar un análisis integral del territorio. Para esto, son indispensables los sistemas de monitoreo territorial. El presente trabajo contribuye en el desarrollo del sistema de monitoreo mediante la construcción de un modelo de visión computacional para la detección automática de objetos que pueden ser de interés para el cuidado del bosque. Por ejemplo: árboles, caminos y vegetación muerta.

1.3. Problema

En el monitoreo forestal, los métodos de teledetección son especialmente adecuados para la identificación y análisis de objetos de interés, especialmente cuando no es posible acceder al terreno debido a su difícil acceso o por temas de seguridad.

Para el desarrollo del presente trabajo, se obtuvieron datos mediante el uso de cámaras montadas sobre un UAV que permitieron adquirir imágenes de un área del Bosque de la Primavera que se encuentra en custodia del ITESO. Así, el problema que se aborda en este trabajo de obtención de grado es, la construcción y entrenamiento de un modelo de aprendizaje profundo que permita realizar un monitoreo automático en las imágenes capturadas por el UAV, considerando que los objetos de interés para este trabajo son los árboles.

1.4. Objetivos

1.4.1. Objetivo general

Construir un modelo de aprendizaje profundo que a partir de las imágenes aéreas tomadas por un vehículo aéreo no tripulado, detecte la presencia de árboles en el Bosque da la Primavera.

1.4.2. Objetivo específico

En el trabajo de obtención de grado se han establecido los siguientes objetivos específicos:

- Identificar modelos de aprendizaje profundo adecuados para localizar y clasificar objetos de interés.
- Desarrollar un módulo de visión computacional basado en técnicas de aprendizaje profundo que permita detectar árboles en el bosque de la primavera.
- Generar un conjunto de datos etiquetados y anotaciones de los objetos de interés a partir de las imágenes aéreas.

1.5. Organización del documento

El resto del documento se encuentra organizado de la siguiente manera: en el capítulo 2 se presentan los principales trabajos relacionados al problema que se aborda; el capítulo 3 describe el marco teórico que fundamenta el desarrollo metodológico (capítulo 4); en el capítulo 5 se muestran los resultados obtenidos al probar y evaluar el modelo de aprendizaje profundo propuesto para la detección de árboles; finalmente, en el capítulo 6 se presentan las conclusiones y el trabajo futuro.

2. ESTADO DEL ARTE

Resumen: *La clasificación, localización y detección de objetos es un importante campo dentro de la visión computacional, en los últimos años las redes neuronales convolucionales han tomado relevancia en esta área al tener un buen desempeño en competencias y en las aplicaciones dedicadas en este tema; a continuación se presenta una breve recopilación de trabajos de interés dentro de las redes neuronales convolucionales.*

2.1. Introducción

La detección de objetos es uno de los problemas dentro de la visión computacional, el cual busca localizar objetos y clasificarlos a partir de imágenes o videos, en 2D o 3D. El entendimiento de la imagen por medio de la detección de objetos es una base para resolver otros problemas como la segmentación, comprensión del contexto, seguimiento de objetos, detección de eventos o actividades[4] que pueden ser usados en gran cantidad de aplicaciones como visión de robots, seguridad, conducción autónoma, vigilancia y realidad aumentada por plantear algunos[4]. Antes de ganar popularidad los modelos de aprendizaje profundo, se habían desarrollado técnicas como el detector Viola-Jones[5] en donde se logra la detección de rostros en tiempo real, siendo más rápido que cualquier otro algoritmo en su momento incorporando técnicas como imagen integral, selección de características y cascadas de detección. El detector HOG[6](Histogramas de Gradientes Orientados), el cual en el trabajo de 2005, se centra en la detección de peatones, cuya técnica mejoraba en su momento, la invariación en la escala y el contexto. Y el modelo basado en partes deformables (DPM por sus siglas en inglés)[7], en el cual el modelo aprende la forma de descomponer un objeto y la inferencia se obtiene por la composición de las diferentes objetos detectados.

2.2. Modelos de detección de objetos genéricos

Las técnicas de aprendizaje profundo aplicadas en el problema de la detección de objetos han tomado la ventaja frente a otras técnicas, teniendo un mejor desempeño en precisión; problema que antes de la irrupción del aprendizaje profundo se consideraba un problema lejos de resolverse, y que hoy se ha puesto en la mira de muchos trabajos. El objetivo de crear sistemas de detección genéricos que puedan compararse a la habilidad humana(los seres humanos pueden identificar alrededor de 3000 categorías de entrada y 30000 categorías visuales [4]).

A partir del 2012, en el desafío llamado "ImageNet Large Scale Visual Recognition Challenge" [8] (ILSVRC por sus sigla en inglés), un modelo de aprendizaje profundo, ganó en la tarea de clasificación de objetos. El modelo de redes neuronales del equipo SuperVision [9] presentó un error de clasificación de 16.4 %, cuando en años anteriores, el error de clasificación había sido de 28.2 % en 2010 y 25.8 % en 2011. En los siguientes años los modelos de aprendizaje profundo continuaron ganando el desafío disminuyendo el porcentaje de error, que ya en 2014 fue de 6.7 %.

En cuanto a la tarea de detección, esta se agregó al desafío ILSVRC a partir del año 2013 y cuya métrica de evaluación fue mean Average Presicion (mAP), que al igual que en la tarea de clasificación, los modelos con técnicas de aprendizaje profundo fueron los ganadores.

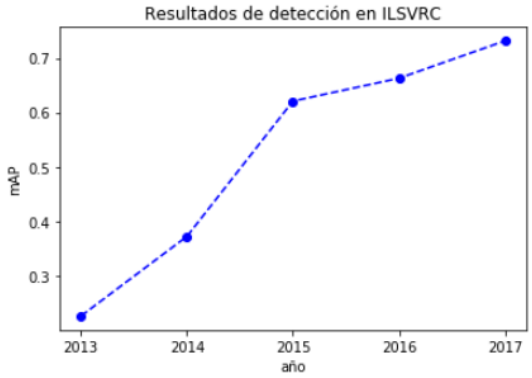


Figura 2.1: Resultado de los ganadores en ILSVRC en la tarea de detección [8]

Como se observa en le imagen 2.1, la métrica mAP de los modelos ganadores, mejoró durante los años 2013 a 2017 con valores desde 0.22 hasta 0.73. Así este tipo de modelos han ganado presencia en la publicación de trabajos en el área de la detección de objetos. Por estos resultados se optó por un modelo de detección de objetos basados en aprendizaje profundo para este trabajo de grado.

El éxito de los detectores de aprendizaje profundo se ha soportado en la gran cantidad de datos de entrenamiento y en redes con millones o más parámetros. Además de la de la disponibilidad de GPUs con gran capacidad cómputo y los grandes conjuntos de datos públicos.

El objetivo de la detección de objetos denota un problema más general de identificación/localización de todos los objetos presentes en una imagen, problema que incorpora la clasificación de los objetos presentes en la imagen y a su vez la localización del objeto, respondiendo las preguntas ¿Hay un objeto? ¿Dentro de qué clasificación se encuentra el objeto? ¿Dónde se encuentra el objeto?

2.2.1. Clasificación

El objetivo de la clasificación de objetos o la categorización es evaluar la presencia de objetos que pertenecen a una clase, a partir de un conjunto de datos como una imagen y asignando una o varias etiquetas del conjunto de clases a ese conjunto de datos que se esta evaluando.

2.2.2. Localización

La localización espacial y las dimensiones de un objeto puede ser definido de diversas formas, por ejemplo, usando una caja delimitadora que es un rectángulo que se ajusta lo

más posible al objeto, una máscara de pixeles precisa del objeto o una delimitación precisa del objeto. Para este trabajo de grado se tomarán en cuenta solamente los detectores de objetos cuyo problema se aborda usando marcos delimitadores o cajas envolventes. Esto debido a las limitaciones de no tener un conjunto de datos etiquetados y el tiempo, solo se trabajará en crear las etiquetas y las cajas delimitadoras en una herramienta en línea.

2.3. Detectores

Para la revisión de los detectores basados en aprendizaje profundo, siguiendo la disección propuesta en el trabajo de Liu et al.[4], se pueden dividir en dos categorías principales: a) Detectores de dos etapas, donde en una primera etapa proponen regiones de interés antes de pasar a la clasificación y localización. b) Detectores de una etapa, donde el modelo en una sola etapa se clasifica y localizan los objetos de interés.

2.3.1. Detectores de dos etapas

Dentro de los detectores de dos etapas, la primera etapa se encarga de proponer regiones que probablemente puedan contener una instancia de clase, para después tomar un mapa de características de esas regiones y pasarlo a una segunda etapa donde se define la localización y clase de la instancia. Trabajos como estos son Fast R-CNN[10] en el cual mejora la velocidad de trabajos anteriores al compartir el mapa de características y donde las entradas a la red neuronal son la imagen y las regiones de interés, así el modelo Faster R-CNN[11] viene a mejorar el proceso de detección al agregar la tarea de generar la regiones de interés a la red neuronal, esto, por medio de una red entre el mapa de características y la red que obtiene la ubicación y la clasificación, quitando la necesidad que otro algoritmo las calcule e introduce la generación de marcos ancla de diferentes tamaños y aspectos, para atacar el problema de la escala y la traslación; R-FCN[12] trabaja con el uso de la red de propuesta de regiones de interés, cambiando la últimas capas densas del modelo a capas convolucionales mejorando los resultados del modelo Faster R-CNN; Mask R-CNN[13] utiliza como base la red Faster R-CNN, solo agregando al final una rama extra para generar una máscara de pixeles para tener una resultado de segmentación de instancias y generando mejores resultados que Faster R-CNN en la detección; Cascade R-CNN[14] trabaja con una arquitectura, donde una unidades de redes parecidas a Fast R-CNN se conectan en cascada, la salida de la ubicación de una unidad se toma como propuesta de región para la siguiente unidad, por lo que se conectan varias unidades en forma de cascada para mejorar la predicción en cada unidad. En la siguiente tabla 2.1 se presentan los resultados en diferentes conjuntos de datos.

Se observa en la tabla 2.1, que el conjunto de datos ha ido cambiando con los años, introduciendo mayor reto al pasar del conjunto de datos Pascal VOC 2017 (20 categorías)

Nombre de modelo	Año	Etapas	Backbone	Método localización	Resultados	Tiempo de inferencia (s/imagen)
Fast R-CNN	2015	2	VGG16	BB	mAP VOC 2007 65.7 % mAP VOC 2012 66 %	0.32
R-FCN	2016	2	ResNet-101	BB	mAP VOC 2007 83.6 % mAP VOC 2012 82 %	0.17
Faster R-CNN	2016	2	VGG16	BB	mAP VOC 2007 78.8 %	0.2
Cascade R-CNN	2017	>2	ResNet-101	BB	COCO AP 42.8 % COCO AP50 62.1 % COCO AP70 46.3 %	-
Mask R-CNN	2018	2	ResNext-101-FPN	BB + SS	COCO AP 37.1 % COCO AP50 60.0 % COCO AP75 39.4 %	0.2

Tabla 2.1: Tabla de modelos de 2 o más etapas. Bounding Box (BB), Semantic Segmentation (SS)

al conjunto de COCO con 80 categorías para la tarea de detección.

2.3.2. Detectores de una etapa

A pesar de que los detectores de dos etapas llevan la delantera en cuanto a resultados en los conjuntos de datos públicos que sirven como referencia para la comparación, estos son costosos computacionalmente para los dispositivos móviles o dispositivos en los que se tienen limitaciones computacionales, almacenamiento y batería. Por lo que también ha habido esfuerzos en el desarrollo de estrategias de detección unificadas en una sola etapa. Dentro de los detectores de una etapa, se refieren a aquellos en que en una misma etapa directamente predice la probabilidad de clase y localización, que no involucra la generación de propuestas de regiones. Algunos de estos trabajos son OverFeat[15], YOLO[16], SSD[17], RetinaNet[18], CornerNet[19].

2.4. Trabajos Relacionados

2.4.1. Detección de copas de árboles individuales

En el trabajo de Ben G. Wenstein et.al[20] se presenta la detección de copas de árboles en imágenes RGB usando aprendizaje profundo semi supervisado. Donde se logra un porcentaje de 82 % en la precisión media de detección de árboles usando un límite en la intersección sobre la unión de 0.5. Se utilizó un acercamiento semi-supervisado para el entrenamiento del modelo. Para un primer conjunto de datos de entrenamiento, las

Nombre de modelo	Año	Etapas	Backbone	Método localización	Resultados	Tiempo de inferencia (s/imagen)
Overfeat	2014	1	Overfeat	BB	mAP ImageNet 24.3 %	-
YOLO	2016	1	YOLO	BB	mAP VOC 2007 63.4 %	0.022
SSD	2016	1	VGG16	BB	mAP VOC 2007 68 % mAP VOC 2012 80 % COCO AP 26.8 % COCO AP50 46.5 % COCO AP75 27.8 %	0.016
RetinaNet	2018	1	ResNext-101-FPN	BB + SS	COCO AP 40.8 % COCO AP50 61.1 % COCO AP75 44.1 %	0.198
CornerNet	2019	1	Hourglass	PK	COCO AP 42.1 % COCO AP50 57.8 % COCO AP75 45.3 %	0.244

Tabla 2.2: Tabla de modelos de 1 etapa. Bounding Box (BB), Pair Keypoints (PK)

etiquetas se definieron usando algoritmos no supervisados a partir de datos de LIDAR, con lo cuales se toman los cuadros delimitadores para sobreponerlos sobre las imágenes RGB y como segundo conjunto datos se hicieron anotaciones manuales. De esta manera se logró obtener una mayor cantidad de datos de entrenamiento y una mejor precisión respecto del algoritmo no supervisado.

2.4.2. Clasificación automática de árboles

En el trabajo de Masanori Onishi y Takeshi Ise[21] se presenta la clasificación de diferentes tipos de copas de árboles en imágenes RGB, donde se clasifican 7 tipos de árboles con un 89 % de precisión. Se utiliza GoogLeNet como red para la clasificación. Para el entrenamiento se utilizaron datos obtenidos por diferentes software como ArcGIS and eCognition.

3. MARCO TEÓRICO

Resumen: *En este capítulo se presentan conceptos y elementos usados para el desarrollo del modelo de dos etapas Faster RCNN, las partes que lo conforman y algunas redes neuronales usadas para la extracción de características.*

3.1. Aprendiendo a partir de datos

Existen varios acercamientos de aprendizaje máquina, de los más exitosos ha sido el llamado aprendizaje basado en el gradiente. En aprendizaje supervisado se tienen los datos de entrada $\mathbf{x}^p$ donde ^{p} es la p-th patrón de entrada y los resultados esperados o valores verdaderos $\mathbf{d}^p$. Con los datos anteriores un modelo h puede generar predicciones $\mathbf{y}^p$. Que se puede expresar de la siguiente forma (ver descripción de la notación en **Apéndice** A):

$$\mathbf{y}^p = h\left(\mathbf{x}^p, \mathbf{W}\right) \tag{3.1}$$

Donde $\mathbf{W}$ representa la colección de parámetros ajustables en el modelo. Una vez obtenidas las predicciones a partir del modelo h se puede obtener una medida de error o discrepancia entre las predicciones $\mathbf{y}^p$ y los valores verdaderos $\mathbf{d}^p$. Esta medida de error se calcula a través de la llamada función de costo o de pérdida y se puede expresar como:

$$E^p = \ell\left(\mathbf{d}^p, h\left(\mathbf{x}^p, \mathbf{W}\right)\right) \tag{3.2}$$

o para simplificar

$$E = \ell\left(\mathbf{W}\right) \tag{3.3}$$

Una vez obtenido el error, se trata de minimizarlo actualizando o modificando recursivamente los parámetros $\mathbf{W}$ del modelo a través de ciertas reglas. A este conjunto de reglas se le llama optimizador[22]. El optimizador $\mathcal{M}$ determina los siguientes valores de $\mathbf{W}$ tomando en cuenta la historia de las iteraciones previas H_t(el error y valor de los gradientes) y un conjunto de parámetros del optimizador que regularmente se conocen como hiper parámetros ϕ.

$$\mathbf{W}_{t+1} = \mathcal{M}\left(H_t, \phi_t\right) \tag{3.4}$$

El proceso repetitivo de generar una predicción del modelo, calcular el error con la función de costo y actualizar los parámetros del modelo con el optimizador se llama normalmente entrenamiento. Se puede decir que el modelo "aprende" si decrece el error. Para evaluar el modelo existen diversas métricas que se utilizan dependiendo el caso de uso y el tipo de salida esperada.

En las siguientes secciones se presenta el tipo de modelo, optimizadores, funciones de costo y métricas utilizadas en el presente trabajo.

3.2. Introducción de redes neuronales

Las redes neuronales representan uno de los principales enfoques en el paradigma de aprendizaje supervisado para la construcción de detectores de objetos. En esta sección

se presentan los conceptos fundamentales de una red neuronal, se parte de la idea del perceptrón y se concluye con el modelo base de una red neuronal convolucional.

3.2.1. Perceptrón

El perceptrón es un modelo computacional de una neurona[23], donde una sola neurona tiene varias entradas (dentritas), un cuerpo celular y una salida (axón). Replicando esto, el modelo del perceptrón tiene varias entradas y una sola salida, el cuál consiste en un vector de "pesos" $\mathbf{W} = [w_1...w_j]$, un peso que multiplica cada entrada x_i, más un "bias" o tendencia cuyo parámetro es descrito como b. Con los parámetros dados w_i y b_i el perceptrón realiza el siguiente cálculo:

$$f(x) = 1 \qquad if\ b + \sum_{i=1}^{l} x_i w_i > 0 \tag{3.5}$$

$$f(x) = 0 \qquad de\ otra\ manera \tag{3.6}$$

Donde $f(x)$ se llama función de activación. El algoritmo del perceptrón busca obtener o aprender los pesos w tales que dados los datos de cada ejemplo de entrenamiento $x^k = \left[x_1^k...x_j^k\right]$, se obtenga la correcta clasificación binaria d^k, donde d^k será el valor esperado, 1 o 0 en este caso. A continuación, se presenta el algoritmo del perceptrón[23]:

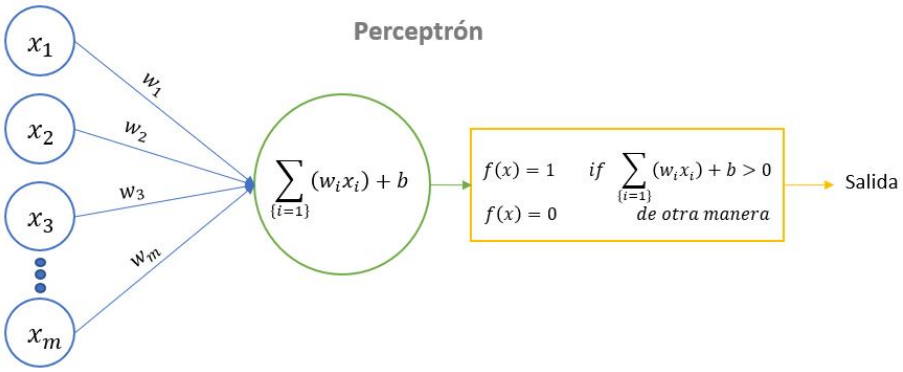


Figura 3.1: Modelo del perceptrón

1. Inicializar los parámetros w_i y b_i
2. Durante N iteraciones (definidas por el desarrollador) o hasta que los pesos no cambien realizar lo siguiente:
   a) Para cada ejemplo de entrenamiento de entrada $\mathbf{x}^k$ con respuesta esperada d^k
      - Calcular $f\left(\mathbf{x}^k\right)$

- Si $d^k - f\left(\mathbf{x}^k\right) = 0$ continua
- Si no, actualizar todos los pesos como a continuación

$$w_i = w_i + \left(d^k - f\left(\mathbf{x}^k\right)\right) x_i^k$$

Donde si existe una configuración de pesos **W** con los cuales se puedan clasificar correctamente cada elemento x^k de entrenamiento, el algoritmo lo encontrará. Esto es verdad en el modelo del perceptrón si los elementos de entrenamiento son linealmente separables[24]. Por lo que para lograr aprender patrones más complejos, se agregan más neuronas o perceptrones como se muestra a continuación.

3.2.2. Red Neuronal

Una red neuronal consiste en un conjunto de neuronas, las cuales tienen parámetros ajustables **W** y "bias". Por medio de un algoritmo la red puede aprender a clasificar correctamente las clases a partir de los datos de entrada $\mathbf{x}^k$ y los valores esperado d^k. En

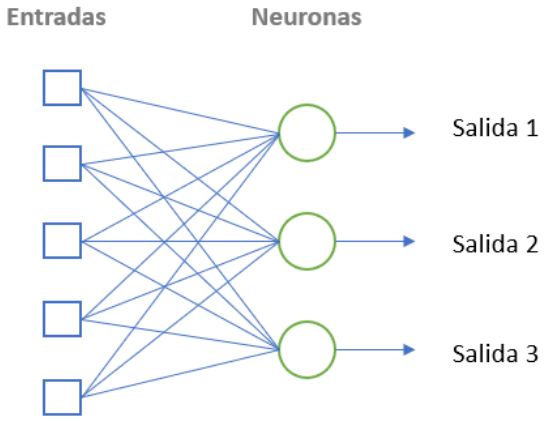


Figura 3.2: Modelo de una red neuronal de una sola capa

la Figura 3.2 se puede observar un modelo de red neuronal con un tipo de arreglo en donde cada neurona esta conectada con cada dato de entrada. Este tipo de arreglo se llama capa densa o completamente conectada. A esta capa densa se le puede agregar una segunda capa, la cual recibirá como entrada, los resultados o salidas de la primera capa como se muestra en la Figura 3.3, teniendo así un modelo de red neuronal multicapa. A las capas que se encuentran entre la entrada y la capa de salida se les suele llamar capas ocultas. Un modelo de red neuronal puede tener **n** capas y es por ello que se comienza a hablar de una profundidad de la red, entre más capas, mayor la profundidad. Al aprendizaje máquina que hace uso de este tipos de modelos se les comenzó a llamar aprendizaje profundo o "Deep Learning" en inglés.

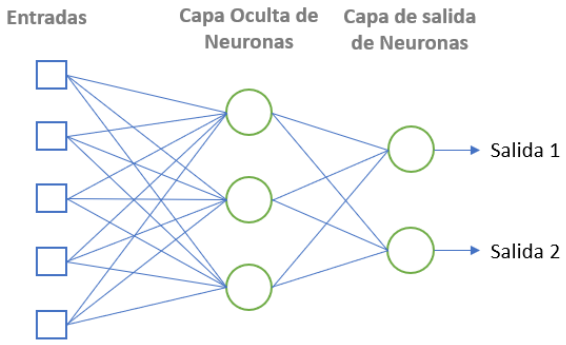


Figura 3.3: Modelo de una red neuronal con una capa oculta

Al usar más capas en los modelos de redes neuronales, el modelo puede aprender patrones más complejos, pero que al igual que el modelo del perceptrón solo puede resolver problemas cuyos elementos de entrenamiento son linealmente separables al usar la función de activación 3.5 por lo que en problemas más complejos en donde la salida deseada no puede ser calculada a partir de transformaciones lineales, se usan funciones de activación no lineales como la función sigmoide 3.7 . En la siguiente figura 3.4 se muestra un ejemplo de un problema lineal y otro no lineal.

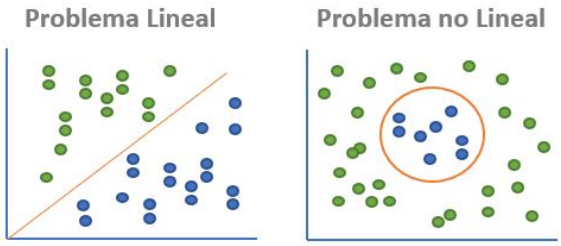


Figura 3.4: Problema lineal(izquierda), Problema no lineal(derecha)

3.2.2.1. Algoritmo de aprendizaje

Un problema que afrontar al tener tantas capas, es el cómo entrenar el modelo. En el trabajo de David E. Rumerhart et.al. (1985)[25] introdujeron el algoritmo de entrenamiento llamado "backpropagation" o retropropagación. El cual es un algoritmo que usa la regla de la cadena para calcular el gradiente del error respecto a cada peso en el modelo. Usando el gradiente del error se pueden actualizar dichos pesos con un optimizador. Una de las condiciones para usar el algoritmo es usar funciones de activación y costo que sean derivables en todos sus puntos.

Existen varias funciones de activación, una de las funciones usadas para las redes neuro-

nales es la función sigmoide:

$$f\left(z\right)=\frac{1}{1+e^{-z}} \tag{3.7}$$

La cual es derivable:

$$f'\left(z\right)=f\left(z\right)\left(1-f\left(z\right)\right) \tag{3.8}$$

Donde

$$z=\ b+\sum_{i=1}^{l}x_i w_i \tag{3.9}$$

Por lo cual se puede utilizar como función de activación y poder calcular el gradiente de la función de perdida para actualizar los pesos **W**. A dicho proceso con el cuál se calcula el gradiente a partir de la función de costo o pérdida y se actualizan los pesos **W** se le llama "backpropagation".

Con ello se resuelve el entrenamiento del modelo de red neuronal multicapa. Donde se divide en un primer paso que es la ejecución del modelo para obtener las predicciones también llamado "feedforward", un segundo paso calcular el error del modelo y utilizar el algoritmo de retropropagación, para finalmente actualizar los pesos usando un optimizador y repetir el proceso.

3.2.3. Función de costo

La función de costo, es una medida de error de las inferencias del modelo en referencia a los valores reales. Para tener esta medida de error existen diferentes tipos de funciones de costo, depende del caso de uso el usar una u otra función. Las funciones de costo utilizadas son en este caso son: entropía cruzada binaria, entropía cruzada categórica y smooth L1.

3.2.3.1. Entropía cruzada binaria

La función de costo estada dada por:

$$-\frac{1}{N}\sum_{i}^{N}f(y_i,y_i') \tag{3.10}$$

$$f(y_i,y_i')=\{y_i log(y_i')+(1-y_i)log(1-y_i')\} \tag{3.11}$$

Donde: y_i son los valores verdaderos y y_i' son los valores predichos por el modelo. Esta función es usada para calcular el error en la salida de la red de propuesta de regiones, dado que la predicción es binaria, es decir clasifica si hay o no un objeto de interés.

3.2.3.2. Entropía cruzada categórica

La función de costo estada dada por:

$$-\frac{1}{N}\sum_{i}^{N} f(y_i, y_i') \tag{3.12}$$

$$f(y_i, y_i') = \sum_{j}^{M} \left(y_{ij} log(y_{ij}')\right) \tag{3.13}$$

Donde N representa el número de muestras, M representa el número de clases, y_{ij} representa si la muestra i pertenece a la clase j y y_{ij}' es la probabilidad que la muestra pertenece a la clase j.

Esta función de costo es útil para calcular el error cuando se tienen más de dos clases ya que se calcula el error de cada clase. Que en el caso del presente trabajo de grado será la función de costo utilizada para calcular el error en la clasificación de la red de detección.

3.2.3.3. Smooth L1

La función de costo estada dada por:

$$-\frac{1}{N}\sum_{i}^{N} f(y_i, y_i') \tag{3.14}$$

La función Smooth L1 esta dada por:

$$f(y_i, y_i') = \begin{cases} \frac{1}{2}(y_i - y_i')^2 & if|(y_i - y_i')| < 1 \\ |(y_i - y_i')| - \frac{1}{2} & otherwise \end{cases} \tag{3.15}$$

Donde y_i son los valores verdaderos y y_i' son los valores predichos por el modelo. Esta función de costo, para los valores x menores a 1 la función es cuadrática y lineal para valores más grandes siendo menos sensible a estos valores. Esta función se utiliza en para calcular el error en las predicciones de las coordenadas de los cuadros delimitadores.

3.2.4. Optimizadores

Los optimizadores $\mathcal{M}$ son reglas para la actualización de los parámetros o pesos de los modelos. Para la actualización de los parámetros toma en cuenta la historia H_t de las iteraciones previas.

$$H_t = \{\mathbf{W}_s, \nabla\ell\left(\mathbf{W}_s\right), \ell\left(\mathbf{W}_s\right)\}_{s=0}^{t} \tag{3.16}$$

Estos optimizadores a su vez tienen hiperparámetros ϕ que modifican el ritmo de aprendizaje.

A continuación se muestran algunos como el gradiente descendiente estocástico(SGD por sus siglas en inglés) y Adam.

3.2.4.1. Gradiente descendiente estocástico

El optimizador SGD esta dado por:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta_t \nabla \ell \left(\mathbf{W}_t \right) \tag{3.17}$$

donde η_t es el hiperpárametro llamado razón de aprendizaje.

3.2.4.2. Adam

El optimizador Adam esta dado por:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \alpha_t \frac{m_{t+1}}{\sqrt{v_{t+1}} + \epsilon} b_{t+1} \tag{3.18}$$

donde

$$m_0 = 0, v_0 = 0$$
$$m_{t+1} = \beta_1 m_t + \left(1 - \beta_1\right) \nabla \ell \left(\mathbf{W}_t \right)$$
$$v_{t+1} = \beta_2 v_t + \left(1 - \beta_2\right) \nabla \ell \left(\mathbf{W}_t \right)^2$$
$$b_{t+1} = \frac{\sqrt{1 - \beta_2^{t+1}}}{1 - \beta_1^{t+1}}$$

3.2.5. Evaluación de Modelo

Una etapa fundamental en el proceso de aprendizaje supervisado es la evaluación que permite determinar que tan bueno es el desempeño de un modelo. En el caso particular de un detector de objetos, se pueden utilizar métricas como la precisión y mAP.

3.2.5.1. Precisión

En la evaluación del modelo una métrica es la precisión, la cual es la cantidad de veces que el modelo predijo correctamente la clasificación entre la cantidad total de elementos clasificados.

$$precision = \frac{num\ de\ predicciones\ correctas}{num\ total\ de\ predicciones}$$

Esta métrica puede quedar limitada en la interpretación cuando se trabaja con un conjunto de datos no balanceados. Se tomará está métrica durante el entrenamiento del modelo.

3.2.5.2. mAP

La métrica mAP cuando es usada en un modelo de detección de objetos se refiere a la precisión del modelo en la clasificación cuando la IOU del cuadro delimitador predicho es mayor a cierto límite α. Donde α es un valor entre 0 y 1. Por ejemplo se puede calcular la precisión del modelo cuando IOU>0.5 y notación puede ser AP_{50} o mAP@0.5. En algunas competiciones puede ser calculado tomando la precisión desde IOU>0.5 e ir aumentando en pasos de 0.05 hasta 0.95, luego se calcula el promedio de la precisión de cada límite de IOU y se obtiene la métrica cuya notación puede ser AP@[0.50:0.95] o mAP@[0.50:0.05:0.95][26].

3.2.6. Redes neuronales convolucionales

Hasta ahora se ha considerado las redes neuronales completamente conectadas, pero estas también pueden ser parcialmente conectadas y un caso especial son las redes neuronales convolucionales, las cuales son muy recurridas es visión computacional[27]. Las redes convolucionales combinan tres ideas para asegurar algún grado de invarianza en deslizamiento, escala y distorsión: campos receptivos locales, pesos compartidos y sub muestreo espacial o temporal. Los campos receptivos locales son llamados filtros o kernel, donde para una imagen en 2 dimensiones un filtro de dimensión 3x3, como ejemplo se podría ver de la siguiente forma:

$$\begin{matrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \\ a_3^1 & a_3^2 & a_3^3 \end{matrix}$$

De una imagen de dimensión nxm se toma una parte de la dimensión del filtro viendo la parte superior izquierda como el punto de partida, y se realiza un producto punto, luego se toma de nuevo otra parte de la imagen de 3x3 pero esta vez tomamos los valores que están hacia la derecha desplazando la ventana un paso, uno en este caso y se realiza la misma operación con el filtro. Se repite la misma operación de desplazamiento y producto punto sobre toda la imagen, donde los valores resultantes dan otra imagen después de haber pasado por el filtro. Dicho filtro puede contener valores que filtren líneas verticales, horizontales, esquinas o puntos finales. Por lo que la imagen puede pasar por varios

filtros obteniendo por cada filtro un mapa de característica y al final varios mapas con diferentes características. Esta capa de operaciones con filtros y desplazamiento se llama capa convolucional. Con este tipo de capa se busca que la red pueda aprender a partir de las características obtenidas por los filtros en lugar de los datos crudo de los pixeles.

Otra capa en una red convolucional es llamada "pooling" o de agrupación, que es un sub muestreo de la imagen de entrada para reducir sus dimensiones, la operación que realiza puede ser que a partir de una ventana por ejemplo de dimensión de 2x2 de la imagen de entrada y obtener el mayor valor que pasara a conformar la imagen reducida de salida, al finalizar el barrido sobre la imagen. Con este tipo de capa se busca reducir la sensibilidad de la red al desplazamiento y las distorsiones de la imagen entrada. En la Figura 3.5

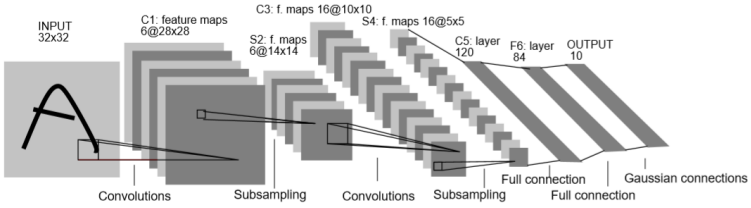


Figura 3.5: Modelo red neuronal LeNet-5[28]

se presenta un ejemplo de una red convolucional llamada LeNet-5 en donde se puede observar 3 capas convolucionales C1, C3 y C5, dos capas de agrupación S2 y S4 y una capa completamente conectada.

3.3. Detector de objetos

A continuación se describe el modelo Faster R-CNN, el cuál fue elegido para el presente trabajo.

3.3.1. Faster R-CNN

En el articulo Faster R-CNN se describe una arquitectura de modelo de un detector de dos etapas: En una primera etapa, se introdujo como mejora al modelo Fast R-CNN, una red que propone las regiones de interés(RPN por sus siglas en inglés) a partir de una imagen en donde se encuentra el o los objetos de interés, para posteriormente en la segunda etapa predecir la clasificación y localización de los objetos de interés basado en las regiones de interés propuestas por la primera red RPN, esta segunda red es llamada Fast R-CNN. Así el modelo de la red RPN y Fast R-CNN es llamado Faster R-CNN.

Ahora se ahondará en los detalles de las ideas e implementación del modelo. Las imágenes

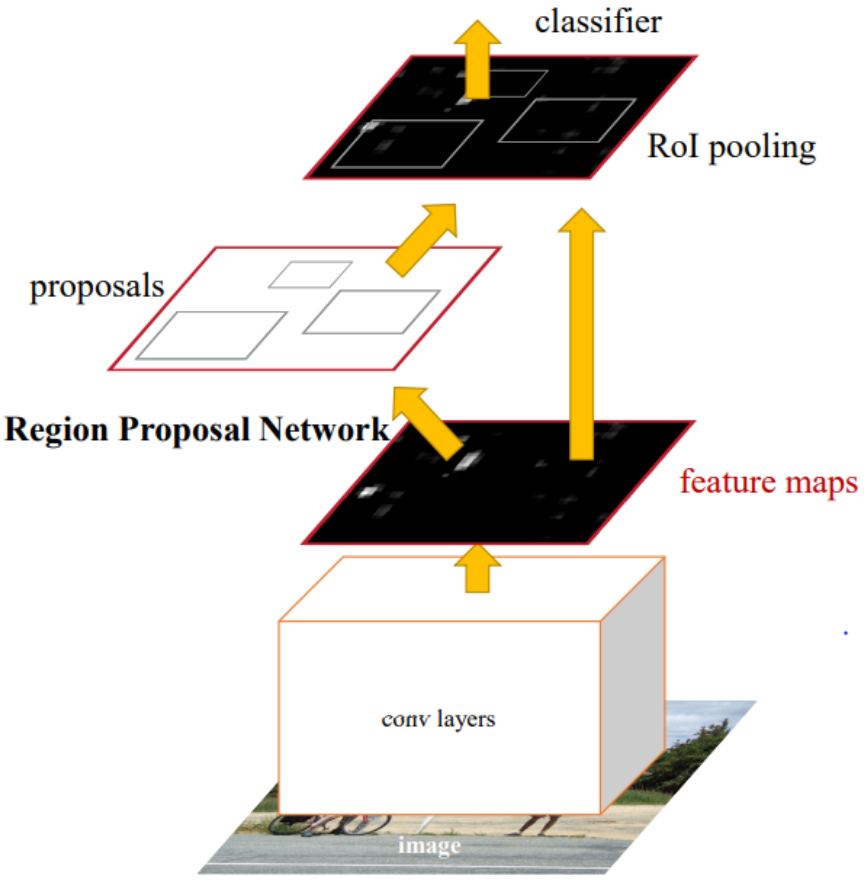


Figura 3.6: Modelo Faster R-CNN[11]

de entrada se supondrán de tamaño 800x800 pixeles y 3 canales(RGB).

3.3.2. Primera etapa: Propuesta de regiones

En esta primera etapa se genera el mapa de características de las imágenes con el cual se predicen diferentes regiones o marcos con la clasificación acerca de si se encuentra algún objeto de interés o no. Una realizadas las propuestas se comparan contra marcos ancla que se generan a partir de las ubicaciones y clasificaciones reales de los objetos a detectar.

3.3.2.1. Mapa de características

Primero comenzamos con la red Region Proposal Network (RPN). Como se observa en la figura 3.6 se tiene primero una generación de mapas de características a partir de de la imagen de entrada. En el artículo, este mapa de características se obtiene usando una red VGG16 sin las últimas capas densas como se ve a continuación en la figura 3.7. La salida

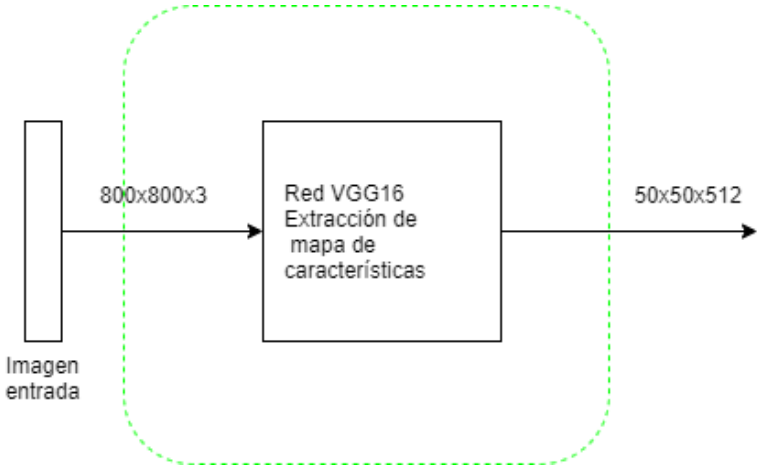


Figura 3.7: Extracción de mapa de características

del mapa de características es de 50x50x512, lo que significa que se tienen 512 mapas, cada mapa de 50x50 lo podemos ver como un submuestro de alguna característica de la imagen original de tamaño 800x800 a 50x50, donde cada pixel en este caso sería un arreglo de dimension 1x1x512.

3.3.2.2. Red de propuesta de regiones RPN

Una vez obtenido el mapa de características se pasan a una red neuronal convolucional cuya salida se divide en dos ramas, en una rama se predice si en esa región hay o no un

objeto de interés y en otra la predicción de la ubicación de dicha región. La red predice 9 marcos por cada pixel del mapa de 50x50 La rama que propone las ubicaciones realiza 9 predicciones de marcos de diferentes tamaños y aspectos por cada pixel del mapa de 50x50, cada marco se representa con 4 valores, lo que resulta una salida de 50x50x36 y la rama que predice el objeto de interés se representa con 2 valores por lo que la salida de esta rama es de 50x50x18.

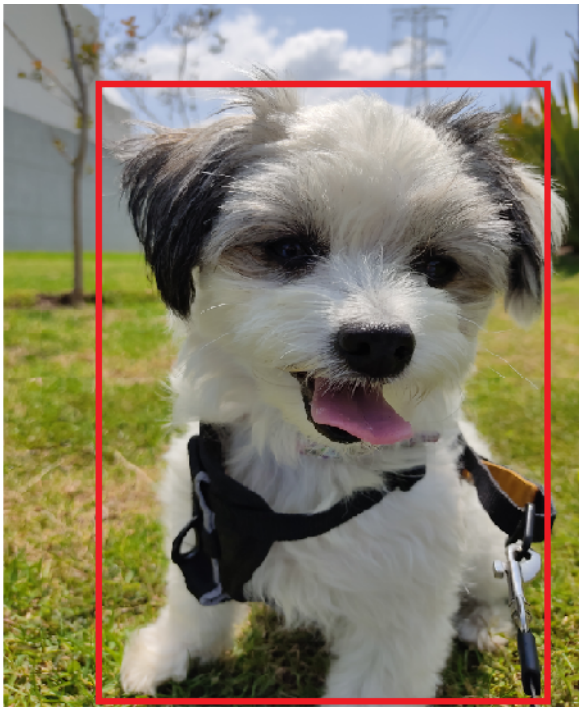


Figura 3.8: Objeto de interés abarcando la mayor cantidad de la imagen

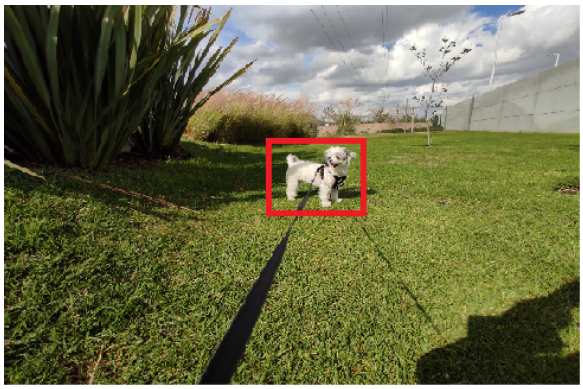


Figura 3.9: Objeto de interés abarcando una pequeña cantidad de la imagen

Un problema al que se enfrenta un detector de objetos es la escala, ejemplificando, si se tuviera una imagen donde el objeto de interés fuese un perro, se esta de acuerdo que lo que se esperaría de un detector es que independientemente del tamaño (un perro en la imagen ya sea que apareciera en todo el espacio de la imagen o si el perro apareciera en una pequeña región), el perro fuera detectado como se muestra en la figuras 3.8 y 3.9.

Para lidiar con este problema lo que han hecho los autores es hacer que las predicciones se realicen en 3 escalas de marcos delimitadores, (128, 256 y 512 pixeles), pero todavía esto no resuelve la duda de por qué se predicen 9 regiones y la respuesta se desprende del hecho que un cuadrado de 128x128 podría no encajar muy bien en muchos casos, por lo que también introducen una variable más, el aspecto de la región a predecir, que en este caso lo introducen en radios de las dimensiones del marco, en el que plantean 3 formas de las regiones, un rectángulo horizontal, un cuadrado y un rectángulo vertical definidos en radios (2:1, 1:1, 1:2). Por lo que, por cada escala se tienen 3 tipos de formas de las regiones a predecir, obteniendo así 3x3, es decir 9 regiones a predecir. Esto nos da que las predicciones de si es una región de interés serán realmente 50x50x2x9 = 45000 valores en un arreglo de la forma (50, 50, 18) que después será modificado para obtener la forma (22500, 2). Y las coordenadas de la región 50x50x4x9 = 90000 de valores en un arreglo de la forma (50, 50, 36), que después será ordenado para obtener la forma (22500, 4). Y de esta forma se trata de atacar el problema de la escala y la forma del objeto a predecir. En la figura 3.10 se puede observar un ejemplo de los aspectos de los marcos que se esperan predecir y en la figura 3.11 se muestra el modelo de red convolucional que realiza las predicciones a partir del mapa de características.

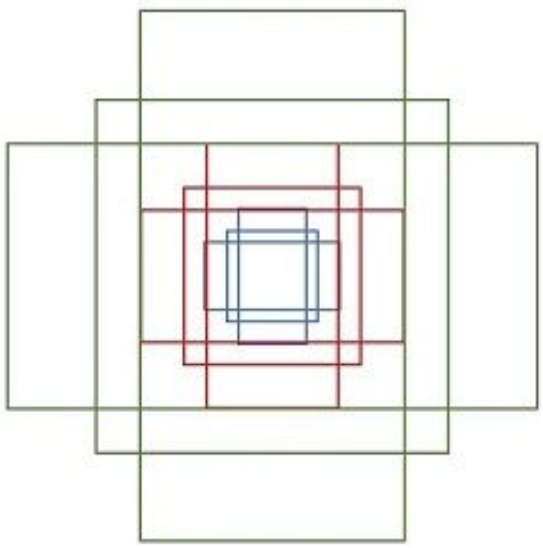


Figura 3.10: Marcos ancla

3.3.2.3. Generación de marcos ancla

Hasta ahora se ha visto parte de la primera etapa del detector, notando que las salidas de la primera etapa son 2 (un arreglo de (22500, 2) para identificar si es una región de interés y otro arreglo de (22500, 4) para la localización de la región) con un total de 22500 regiones predichas. Hay que recordar que el modelo es supervisado y se tienen que generar

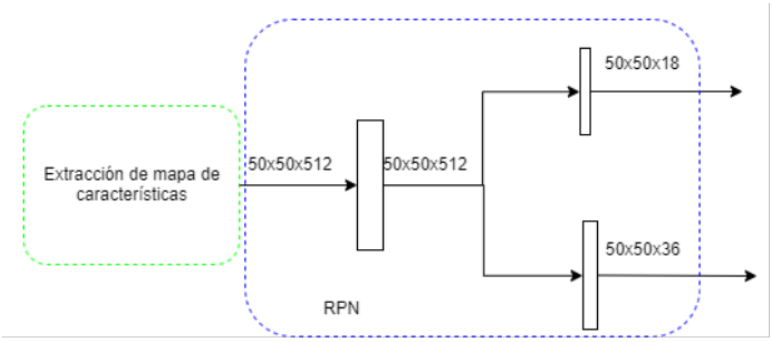


Figura 3.11: Red de propuesta de regiones

o tener los valores esperados para poder entrenar al modelo. A continuación se presenta como generar estos valores esperados.

Para ello se trae de vuelta los marcos delimitadores con 3 diferentes tamaños y 3 diferentes aspectos de tamaño en cuanto a la longitud de sus lados, estos marcos son los objetivos a ser predichos por la primera etapa del RPN. Por lo tanto necesitamos generar estos marcos que servirán durante el entrenamiento y que en el trabajo de Faster R-CNN son llamados marcos ancla.

La forma de generarlos es la siguiente: 1) Generar los puntos de ancla que serán los centros de las marcos. 1.1) Los puntos ancla son distribuidos en la imagen de entrada de 800x800 cada 16 pixeles. Estos nos da $800//16 = 50$ puntos por lado lo que da un total de 50x50 de puntos ancla = 2500 de puntos ancla. 1.2) Sobre los puntos de ancla se generan 9 marcos delimitadores (dada la combinación de 3 tamaños diferente y los 3 aspectos de tamaño). Por lo que se obtendrían 2500x9 marcos ancla = 22500 marcos ancla, que es la cantidad que también tenemos en las salidas de del RPN. 1.3) Para cada marco delimitador se crea una etiqueta numérica donde se definirá su clasificación y que se inicializará en -1.

2) Una vez generados los marcos delimitadores hay que asignarles un valor en la etiqueta para clasificar si ese marco tiene o no un objeto de interés. 2.1) Descartar todos los marcos ancla cuyas coordenadas no estén dentro de la imagen de 800x800 y se deja la etiqueta -1. En este caso la cantidad de marcos ancla válidos se reduce a 8940, los cuales se tienen que clasificar. 2.2) Para llevar a cabo esta tarea se toman las marcos delimitadores objetivo que contienen al objeto de interés real (codificados en 4 coordenadas, hasta ahora no se había detallado el formato de las coordenadas, en este caso serán coordenadas cartesianas [x1,y1,x2,y2]), y la etiqueta de a qué clase pertenece(la clase está representada por un número).Y para cada marco ancla se calcula la intersección sobre la unión (IOU) con cada marco delimitador objetivo. Es decir si la imagen contiene 4 objetos de interés que se quieren predecir, se tendrán que calcular 8940x4 IOUs, por cada marco ancla se tendrán los 4 valores de IOU respectivo a cada marco delimitador objetivo. 2.3) Una vez obtenidos los valores, se toman los siguientes criterios para asignarles la clasificación si contienen o no una región de interés. 2.3.1) Se obtienen los IOUs más altos por cada marco delimitador objetivo, es decir se obtendrían en este ejemplo 4 valores, que representaría los máximos

IOUs calculados. Todos aquellos marcos ancla que tengan un IOU igual al valor máximo de alguno de los 4 valores se le asignará la etiqueta 1, clasificándolos como marcos ancla con una región de interés. 2.3.2) Para cualquier marco ancla que tenga un IOU $>= 0.7$ respecto de cualquiera de los marcos delimitador objetivo se le asignará la etiqueta 1, clasificándolos como marcos ancla con una región de interés 2.3.3) Para cualquier marco ancla que tenga IOU < 0.3 en todos los IOUs con respecto de los marcos delimitadores objetivo, se le asignará la etiqueta 0, denotando que no es una región de interés. De esta manera se obtienen 22500 marcos ancla clasificados con los valores -1: marcos ancla inválidos que no se pueden usar para el entrenamientos, 0: Son marcos ancla válidos pero no son una región de interés, 1: Son marcos ancla válidos y contienen una región de interés.

3) Muestreo de marcos ancla válidos. 3.1) Como describen los autores de Fast R-CNN la mayoría de marcos ancla válidos tendrán la etiqueta 0 y otros pocos tendrán la etiqueta 1. Dado que para que la red aprenda si es una región de interés o no, se tiene que entrenar con ejemplos negativos (marcos delimitadores que no contienen una región de interés) y con ejemplos positivos (marcos delimitadores que contienen una región de interés), debido al desbalance entre los marcos ancla con etiqueta 0 y los marcos ancla con la etiqueta 1, se sigue la siguiente estrategia, tomar una muestra de 256 marcos ancla con 50 % de etiquetas con valor 1 y 50 % de etiquetas con valor 0, siendo que habrá en su mayoría etiquetas con valor 0, si se tuviera que la cantidad de marcos ancla con etiqueta 1 fuesen menos de 128 (50 % de 256), se toman todos los marcos ancla con etiqueta 1 y se toman los marcos ancla con etiqueta 0 que sean necesarios para completar los 256. Por último a todos los marcos ancla válidos que no hayan sido seleccionados en el muestreo, su etiqueta de clasificación se pondrá como -1. Más adelante se verá el uso de colocar el valor -1 al calcular la parte de la función de perdida que se enfoca en la clasificación si es o no una región de interés.

4) Asignación de la ubicación de los marcos ancla. Hasta ahora no se ha descrito la interpretación de la salida de la rama de regresión(rama que predice la ubicación del marco delimitador objetivo). Al inicio se hace la suposición que la salida de esta rama daba los valores en el formato de coordenadas cartesianas [x1, y1, x2, y2] pero lo cierto que lo que nos da como resultado, es una parametrización de la siguiente forma $[tx, ty, tw, th]$ donde:

$$t_x = (x - x_a)/w_a$$
$$t_y = (y - y_a)/h_a$$
$$t_w = log(w/w_a)$$
$$t_h = log(h/h_a)$$

x, y, h, w son los parámetros del marco delimitador predicho por el RPN, x_a, y_a, h_a, w_a son los parámetros de los marcos ancla y corresponden al centro en el eje x, centro en el eje y, la altura y ancho del marco respectivamente.

Por lo que para el entrenamiento se debe realizar la parametrización de los marcos ancla de la siguiente forma $[t_x^*, t_y^*, t_w^*, t_h^*]$:

$$t_x^* = (x^* - x_a)/w_a$$

$$t_y^* = (y^* - y_a)/h_a$$
$$t_w^* = log(w^*/w_a)$$
$$t_h^* = log(h^*/h_a)$$

donde x^*, y^*, h^*, w^* son los parámetros del marco delimitador objetivo más cercano al marco ancla que se esté calculando.

3.3.2.4. Función de costo de la red RPN

Las función de costo se define como:

$$L_{cls}(p_i, t_i) = \frac{1}{N_{cls}} \sum_{i}^{Ncls} L_{cls}(p_i, p_i^*) \tag{3.19}$$

$$L_{reg}(p_i, t_i) = \frac{1}{N_{reg}} \sum_{i}^{Nreg} p_i^* L_{reg}(t_i, t_i^*) \tag{3.20}$$

La función L_{cls} se refiere a la función 3.11 y la función L_{reg} se refiere a la función 3.14. Donde i es el índice del marco ancla dentro de los 256 marcos tomados para el entrenamiento y p_i es la probabilidad del marco i predicho por la red RPN de ser un objeto. La etiqueta p_i^* del marco ancla como se ha descrito en los pasos anteriores, es 1 si hay un objeto de interés o 0 en caso contrario. t_i es un arreglo que representa las 4 coordenadas parametrizadas del marco predicho, y t_i^* se refiere a las coordenadas parametrizadas del marco ancla respectivo. La función de costo en la clasificación L_{cls} es la función de costo logarítmica sobre dos clases (Objeto o no objeto en este caso). Para la función de costo de la regresión L_{reg} es una función llamada "smooth L_1) y el término $p_i^* L_{reg}$ significa que la función de costo de la regresión solo se activa si se trata de un marco ancla positivo (etiqueta = 1). N_{cls} y N_{reg} son el número de muestras que se usan en el entrenamiento, en este caso $N_{cls} = 256$ y N_{reg} 2400.

3.3.3. Segunda etapa: Red de Detección

En la segunda etapa se toma un submuestreo de las regiones propuestas en la primera etapa, se acondicionan para que una segunda red haga la predicción del marco delimitador y la clasificación de los objetos de interés.

3.3.3.1. Acondicionamiento de las propuestas para la segunda red

Aquí partimos de las predicciones hechas por la RPN, donde obtenemos 2 arreglos de salida, uno (22500, 2) que nos dice si hay un objeto o no y otro (22500, 4) que predice

la ubicación del objeto. Dado que el número de predicciones es muy grande y la mayoría son casos negativos, es decir marcos que no contienen al objeto de interés, se realiza un procesamiento de estos datos antes de pasar a la siguiente red llamada Fast R-CNN, se una muestra de 128 marcos predichos, estos marcos son llamados regiones de interés o ROI. En las siguientes secciones se muestra el procedimiento para obtener esta muestra.

3.3.3.2. Supresión no máxima o NMS

Primero que nada, en la salida de la RPN, la mayoría de las predicciones son marcos delimitadores que se superponen entre sí, entonces el primer paso es disminuir la cantidad de estos marcos con los menos posibles descartando los marcos que se superponen y tomando aquellos que representen a los marcos descartados y que tengan la mayor probabilidad de que exista un objeto. A esta sección o capa del modelo se le llama de supresión no máxima o NMS figura 3.12.

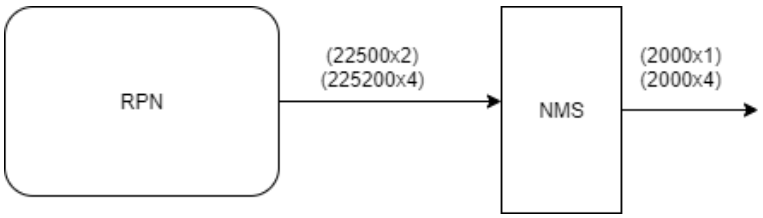


Figura 3.12: Supresión no máxima

1) Se toman las coordenadas parametrizadas de los marcos predichos y se convierten a coordenadas [x1, y1, x2, y2] correspondientes al tamaño real de la imagen de entrada.

2) En donde se tengan coordenadas negativas se fijan en 0 los valores. Y para las coordenadas que sean mayor de 800, se fija al valor de 800. De esta forma las coordenadas se encontraran en el rango de las dimensiones de la imagen real.

3) Se descartan aquellos marcos predichos que sean de tamaño menor a 16 en este caso, dado que cada pixel en el mapa de características representa 16 pixeles de la imagen real y en otra capa del modelo se necesita extraer secciones, más adelante se verá mejor esta decisión de descarte de marcos con límite mínimo de tamaño.

4) Una vez teniendo los marcos redimensionados y con los límites adecuados se ordenan de acuerdo a la calificación de si existe un objeto o no, de mayor a menor, donde el rango de la calificación es [0, 1] donde 1 es el valor más alto y significa que existe un objeto de interés.

5) Una vez ordenados se toman los primeros 12000 marcos con sus calificaciones en este caso (este es un parámetro que se puede modificar en el modelo).

6) A dichos marcos se toma el de mayor calificación, se revisa cuáles otros marcos predichos

tienen un traslape mayor a 0.7 con este marco de mayor calificación y se descartan los que cumplan esta condición. Se pasa con el siguiente marco en orden de clasificación y se procede a hacer lo mismo hasta que se termina de recorrer todos los marco predichos. Una vez realizado este procedimiento, muchos marco habrán sido descartados y solo quedarán aquellos con mayor representación de ciertas áreas.

7) Una vez teniendo los marcos predichos sin traslape se seleccionan solo los primeros 2000 en el entrenamientos.

3.3.3.3. Muestreo de regiones de interés

Una vez obtenidos los 2000 marcos sin traslape después de pasar por la operación de NMS, se toman aleatoriamente 128, de las que el 25 % serán muestras positivas y el 75 % restante serán muestras negativas, figura 3.13.

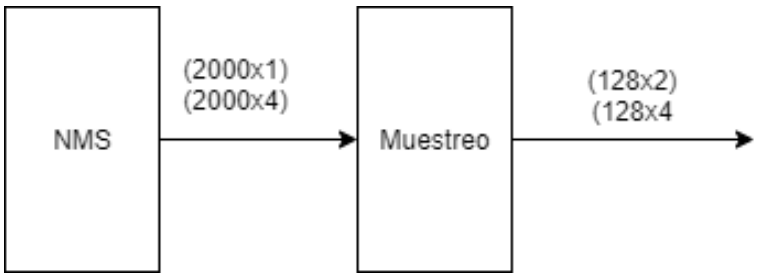


Figura 3.13: Muestreo

3.3.3.4. Fast R-CNN

Las 128 muestras se pasan al través de la red Fast R-CNN. Esta parte del modelo refina las coordenadas del marco predicho y clasifica las muestras ahora sí dentro de las clases objetivo, en lugar de solo definir que existe un objeto o no.

La red Fast R-CNN consta al inicio de una capa de "MaxPool" adaptativa la cual su función es que las predicciones de las coordenadas de las 128 muestras serán de diferente tamaño y para poder introducirlas a la red, primero se necesita fijarlas a cierta dimensión para que sean iguales en las siguientes capas.

Las siguientes capas son parecidas a las capas en la red RPN, que constan de una capa convolucional que se deprende en dos ramas, una rama predice las coordenadas de la ubicación y la otra rama predice la clasificación del objeto, figura 3.14.

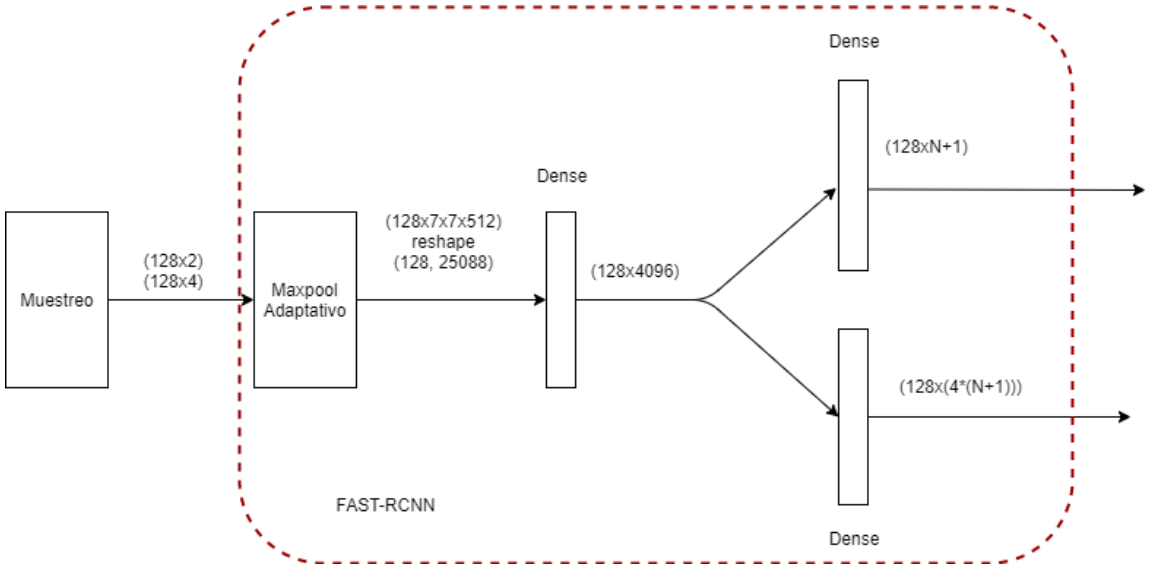


Figura 3.14: Fast RCNN

3.3.3.5. Función de costo de la red Fast R-CNN

La función de costo de la red Fast R-CNN para la detección es igual a la de la red RPN 3.19 y 3.20, solo que esta toma los datos de salida de la red Fast R-CNN y las 128 datos de entrenamiento generados en esta etapa.

4. DESARROLLO METODOLÓGICO

Resumen: *A continuación se presentan las actividades desarrolladas en el trabajo de grado para lograr implementar un modelo de aprendizaje profundo supervisado, el código de la implementación puede ser encontrado en* `https://github.com/JFranciscoCervantes/mvjluis-faster_rcnn`

4.1. Introducción

En este capitulo se presenta la adquisición de la imágenes, preprocesamiento de los datos, y el entrenamiento del modelo Faster R-CNN.

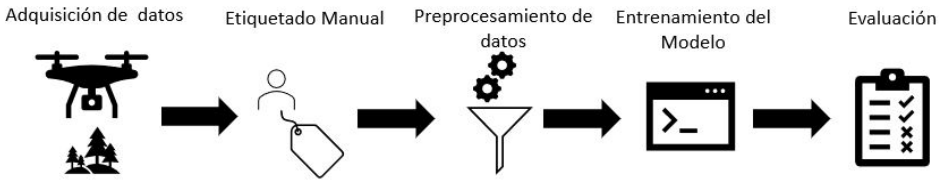


Figura 4.1: Diagrama de metodología

4.2. Adquisición y preprocesamiento de datos

4.2.1. Adquisición de los datos

El proceso para obtener los datos ha sido por medio de un vuelo de un vehículo aéreo no tripulado, en un terreno en el bosque de la primavera (Figura 4.2) se generó una ruta de

vuelo en el cual se van tomando imágenes cada segundo, se tienen imágenes de 20MP en formato de colores RGB y un formato de archivo ".JPG". Se tiene una cantidad de 496 fotos de las cuales se necesita etiquetar los elementos de interés, así como redimensionarlos, un ejemplo de estas imágenes se presenta en la Figura 4.3. El recorrido del vehículo aéreo no tripulado es a una velocidad de 5m/s y una altura de 80m.



Figura 4.2: Polígono del bosque La Primavera. Fuente Google Earth

4.2.2. Datos Obtenidos

Del vuelo mencionado anteriormente se obtiene las imágenes y ciertos logs en la siguiente estructura de carpetas:

En la carpeta BLP_[FECHA] contiene toda la información obtenida del vuelo. En la carpeta "DroneLogs" contiene unos archivos *.rlog, *.tlog y *.xml. Los archivos tlog y rlog son conocidos como lo registros de telemetría que son recolectados por una estación en tierra, la diferencia entre un archivo .tlog y el archivo .rlog, es que el archivo .rlog contienen la información del archivo .tlog e incluye más información para su posterior revisión acerca del vuelo. El archivo .xml es un complemento que contiene la información de vuelo solo en el formato diferente a .tlog. En la carpeta "Fotos" se tienen las mismas fotos mostradas en "Geotdated" solo que, en los metadatos, "Geodated" contienen información extra del GPS (Latitud, Longtud, Altura).



Figura 4.3: Foto aérea tomada por vehículo aéreo no tripulado en bosque de la primavera.

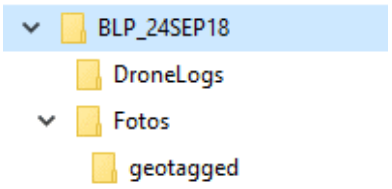


Figura 4.4: Estructura de carpetas de los datos obtenidos

4.2.3. Preparación de datos

En la preparación de las imágenes de los árboles, se toma como base las imágenes obtenidas por el vehículo aéreo no tripulado. Existen varias formas para etiquetar los datos de las imágenes como cajas delimitadoras, segmentación semántica, segmentación por pixeles, mapas de bits, o a partir de polígonos. Para este proyecto se escogieron las cajas delimitadoras, para generar cuadros que delimiten los objetos deseados.

4.2.3.1. Etiquetado de imágenes

Esta actividad es una de las que tienen mayor consumo de tiempo al iniciar un proyecto de aprendizaje máquina, debido a que el etiquetado al inicio se hace de forma manual y la cantidad necesaria en este tipo de proyectos normalmente son miles de datos para poder entrenar un modelo.

Para el presente trabajo, los elementos a etiquetar son:

- Arboles a los cuales se les asigna la etiqueta **tree**
- Ejemplos negativos a los cuales se les asigna la etiqueta **bg**

Los ejemplos negativos servirán en el propósito de que el modelo aprenda que no es un árbol.

Existen varias herramientas para el etiquetado e incluso hay empresas que prestan servicios completos desde la colección de dato s, hasta la entrega del conjunto de datos ya etiquetados y listos para usar en los modelos y en algunos casos todo el proyecto completo. De esta forma la mayoría de las herramientas con una variedad robusta de comandos e interfaz aceptable, son privadas y se necesita realizar pagos para poder utilizar dichas herramientas. De igual forma existen herramientas de código abierto que contienen funcionalidades básicas. A continuación, se presentan algunas de ellas:

Privadas:

- Labelbox
- RedLabel
- LionBridge

Uso libre:

- LabelImg

- Supervisely

Para este trabajo, se utilizó Supervisely dado que contiene un buen balance entre herramientas de uso libre y privadas, no tiene costo y el trabajo se realiza en línea, por lo que no se necesita instalar ninguna paquetería. Se pueden crear proyectos, subir imágenes, crear cuadros delimitadores, polígonos, cuboides, segmentación de mapas de bits.



Figura 4.5: Visualización de la herramienta Supervisely

Hay diferentes formas de guardar las anotaciones, entre las bases de datos más usadas como son:

Common Objects in Context (COCO)[29], el formato usado es .json en donde se pueden guardar diferentes tipos de anotaciones, detección de objetos, detección de puntos clave, segmentación y títulos de imagen. Los bloques de información básicos son: “info”, “licenses”, “categories”, “images” y “annotations”.

Visual Object Classes (VOC)[30], el formato usado es “.xml” y a diferencia de las anotaciones en COCO en que el archivo .json contiene toda la información, en Pascal VOC, existe un documento .xml por cada imagen.

En la herramienta de supervisely, la forma en que se entregan las anotaciones es un híbrido entre el formato Pascal y COCO, dado que entrega un archivo .json por cada imagen, y en cada archivo .json con la información de las anotaciones de la siguiente manera:

```
{
    "description": "csv",
    "tags": [
        {
            "name": "Tree",
```

```
            "value": null,
            "labelerLogin": "mvjluis",
            "createdAt": "2020-05-13T16:03:40.293Z",
            "updatedAt": "2020-05-13T16:03:40.293Z"
        }
    ],
    "size": {
        "height": 3864,
        "width": 5152
    },
    "objects": [
        {
            "description": "",
            "geometryType": "rectangle",
            "labelerLogin": "mvjluis",
            "createdAt": "2020-05-13T15:51:05.568Z",
            "updatedAt": "2020-05-13T15:59:12.830Z",
            "tags": [
                {
                    "name": "Tree",
                    "value": null,
                    "labelerLogin": "mvjluis",
                    "createdAt": "2020-05-13T15:52:17.952Z",
                    "updatedAt": "2020-05-13T15:52:17.952Z"
                }
            ],
            "classTitle": "tree",
            "points": {
                "exterior": [
                    [
                        1910, \\xmin
                        1785   \\ymin
                    ],
                    [
                        2114, \\xmax
                        1963   \\ymax
                    ]
                ],
                "interior": []
            }
        },
```


Donde los campos importantes son: “classTitle” y la ubicación del cuadro delimitador descrito por los puntos “exterior”. Una vez generados los cuadros delimitadores en la herramienta, se procede a exportar los archivos .json.

4.2.3.2. Preprocesamiento de imágenes y etiquetas

En esta sección se va a tratar el proceso desde que se tienen las imágenes iniciales del UAV con los archivos .json obtenidos de la herramienta supervisely, hasta modificar las imágenes en tamaños más adecuados para los modelos y la adecuación de las etiquetas a un archivo simplificado .txt. Para el preprocesamiento se utilizo el lenguaje de programación Python, versión >3.6.

Las imágenes obtenidas por el UAV, son de un tamaño de 20 MP (5152x3864 pixeles). Estas imágenes se van a reducir en un factor de 1200 para tener un tamaño de 4800x3600 pixeles. Debido a la redimensión las etiquetas también se adecuan a las nuevas dimensiones.

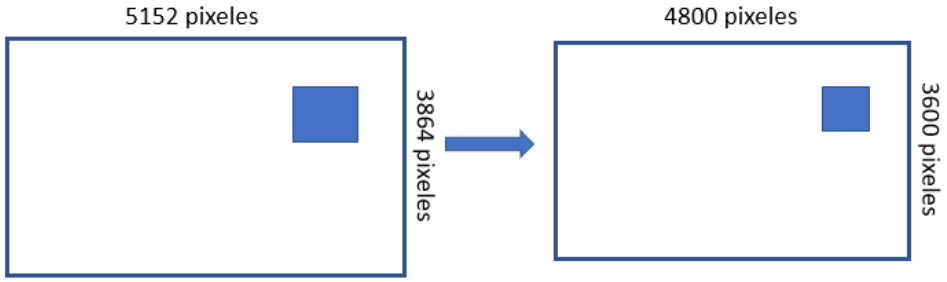


Figura 4.6: Redimensión de imágenes

Una vez redimensionada la imagen y las etiquetas como se muestra en la figura 4.6, se cortaron en un tamaño de 1200x1200 pixeles y la información de las etiquetas se modifican respecto a las nuevas imágenes cortadas.

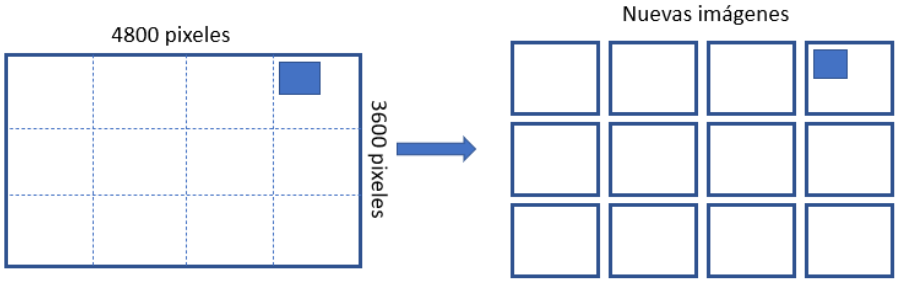


Figura 4.7: Imágenes resultantes

Y donde al final las etiquetas se cambiaran del formato .json a un formato .txt, en este nuevo formato, por cada línea se define una caja delimitadora de la siguiente forma:

Ubicación de la imagen, ancho, alto, x1, y1, x2, y2

Donde x1, y1 y x2, y2; son las coordenadas de los vértices de los cuadros delimitadores. A continuación se muestra un ejemplo.

```
C:\dataset\proc_imagenes\IMG_7248_0_0.jpg,1200,1200,111,164,488,441,bg
C:\dataset\proc_imagenes\IMG_7248_0_0.jpg,1200,1200,479,22,919,299,tree
```

4.3. Modelos

Para el trabajo de grado se utilizó el modelo Faster R-CNN con una implementación en lenguaje de programación python 3.x, librería tensorflow 2.x.

4.3.1. Faster R-CNN

El modelo Faster R-CNN tiene varias implementaciones; como base se utilizó la desarrollada en `https://github.com/you359/Keras-FasterRCNN` [31], la cual hace uso de la biblioteca de tensorflow 1.x. En el desarrollo de este trabajo se realizaron los cambios necesarios para poder trabajar en la biblioteca de tensorflow 2.x. En la figura 4.8 se muestra un diagrama simplificado de la implementación.

En el diagrama del modelo Faster R-CNN es un detector de dos etapas. La primera etapa consiste en la extracción del mapa de características y la propuesta de regiones. La segunda etapa consiste en que a partir de las regiones propuestas se detecten los objetos de interés.

Una diferencia entre el modelo en la publicación de Faster R-CNN[11] al de la implementación es que la capa de entrada del submuestreo del mapa de características de la segunda etapa, en la publicación de realiza un "MaxPool" adaptable mientras que en la implementación se realiza una interpolación para conseguir el tamaño adecuado.

4.3.1.1. Primera etapa: Generador de objetivos para RPN

Antes de la ejecución de modelo se crea un generador de objetivos como se explica en la sección 3.3.2.3. En la implementación del código se tiene un modulo llamado data_generators.py el cual toma cada imagen, la redimensiona a un valor definido, en este caso a 600x600 pixeles, redimensiona los cuadros delimitadores, genera los marcos ancla y selecciona aquellos que se utilizaran para calcular la función de costo de la primera etapa.

4.3.1.2. Primera etapa: RPN

La red de extracción de características es un elemento relevante del modelo. Durante la experimentación se usaron 4 diferentes redes para la extracción de características:

Extractor de características	Número de parámetros
VGG16	14714688
Resnet50	8589187

Tabla 4.1: Redes usadas para la extracción de características

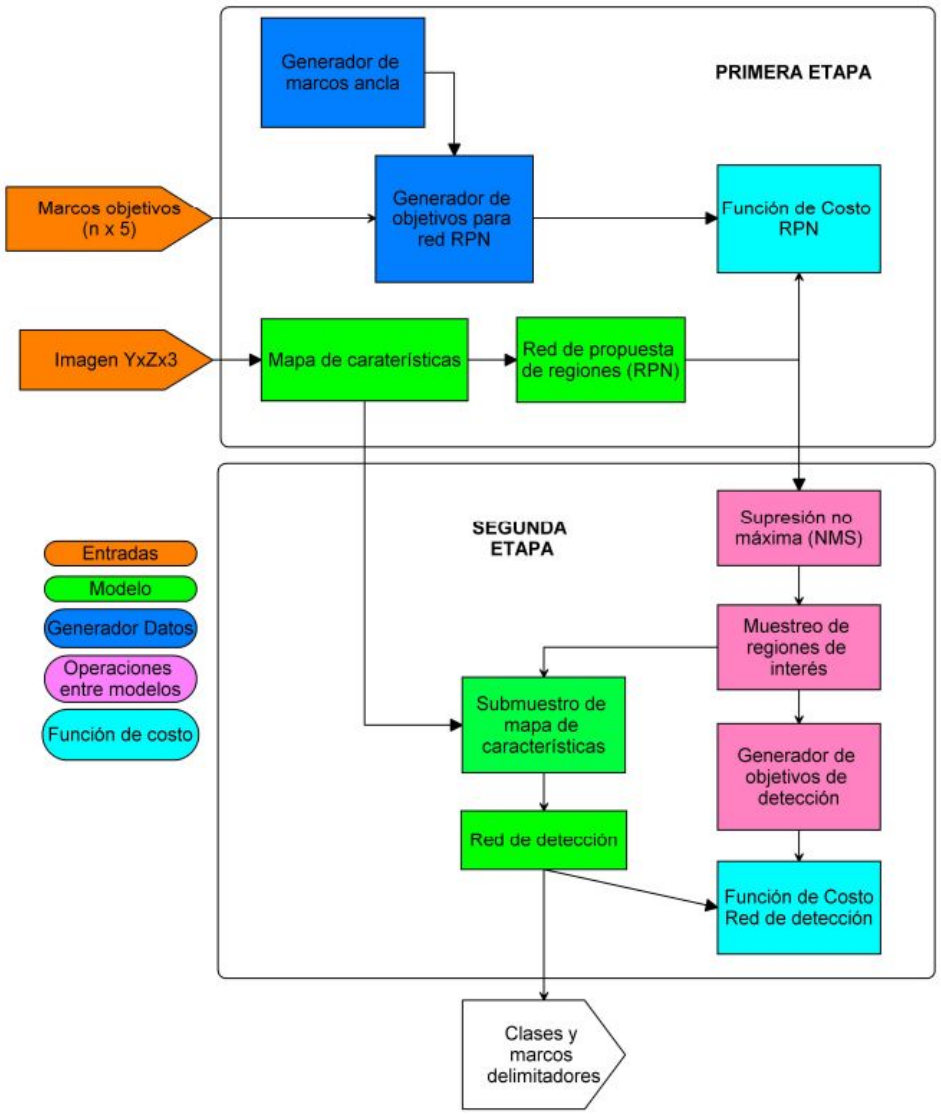


Figura 4.8: Diagrama de implementación del modelo Faster R-CNN

Debido a que no se cuenta con suficiente información para el entrenamiento de esta red, se usaron pesos pre-entrenados en el conjunto de datos imagenet[32]. Estos parámetros también serán entrenables para el ajuste fino con lo nuevos datos.

Una vez obtenido el mapa de características se manda a la red que se encarga de aprender a proponer regiones donde se pueda encontrar los árboles como se muestra en la figura 3.11. La cantidad de parámetros entrenables en esta red es depende del mapa de características.

4.3.1.3. Segunda etapa: Supresión no máxima y muestreo de regiones de interés

Una vez obtenidos la propuesta de regiones, estas se preprocesan antes de pasarlos a la red de detección. Se escogen 128 cuadros delimitadores que tengan mayor sobre posición con los cuadros delimitadores positivos y negativos esperados. El proceso se detalla en la sección 3.3.3.1.

4.3.1.4. Segunda etapa: Red de detección

En este punto se procede a extraer la información del mapa de características de acuerdo a las regiones propuestas, se redimensiona la información para tener el tamaño adecuado para el modelo de detección. En el trabajo original de Faster RCNN se utiliza una capa llamada MaxPool adaptativa, en esta implementación se uso un redimensionamiento por interpolación.

Enseguida se toman las regiones del mapa de características redimensionadas y se procede a realizar la detección. Como resultado se entregan la clasificación de las regiones de interés propuesta así como el ajuste de los cuadros delimitadores.

4.3.1.5. Entrenamiento del modelo

Como entrada recibe los siguientes parámetros:

-p : Ubicación del archivo con las etiquetas que se produjo en la sección 4.2.3.2.

-o : El formato del archivo de etiquetas, por defecto es el formato actual del archivo .txt

-n : Número de propuesta de regiones que se utilizarán para procesar a la vez.

--network : Red a usar para la extracción de características.

--hf : Vueltas horizontales a las imágenes para el aumento de datos.

--vf : Vueltas verticales a las imágenes para el aumento de datos.

--rot : Rotación de 90 grados.

--num_epochs : Número de epocas para el entrenamiento

--config_filename : Nombre del archivo de configuración.

--output_weight_path : Ubicación donde se guardaran los pesos del modelo entrenado.

--input_weight_path : Ubicación de pesos si ya se tienen de entrenamientos previos.

Durante el desarrollo del proyecto se intercambió la red para la extracción de características que se muestran en la tabla 4.1 en el parámetro --network. Los demás parámetros se fijaron como sigue:

-o : simple

-n : 80

--rot : False

--num_epochs : 100

Y para el entrenamiento con aumento de datos se modifican los parámetros:

--hf : True

--vf : True

A continuación se muestran gráficas del comportamiento de las métricas durante el entrenamiento de 100 épocas con diferentes modelos para la extracción del mapa de características. Para la presentación de las gráficas se uso la biblioteca de tensorboard.

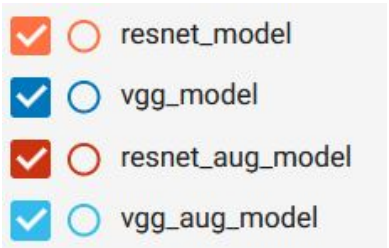


Figura 4.9: Leyendas para las gráficas

En las figuras 4.10 y 4.11 se muestra la disminución de la pérdida o costo de las redes de clasificación en la sección de propuesta de regiones y en la detección.

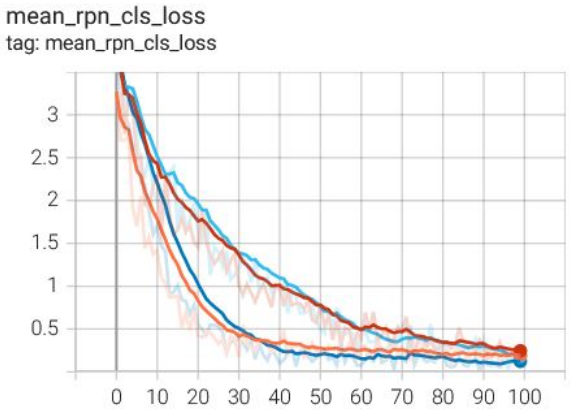


Figura 4.10: Pérdida de clasificación en RPN durante el entrenamiento



Figura 4.11: Pérdida de clasificación en la detección durante el entrenamiento

En la siguiente figura 4.12 se muestra la pérdida o costo total del modelo que se calcula como la suma de los errores en la predicción de clase y determinación de las cajas delimitadores de ambas etapas de entrenamiento (propuesta de regiones y detección).

Por último en la figura 4.13 se muestra la precisión del modelo en la clasificación ya en la etapa de detección.

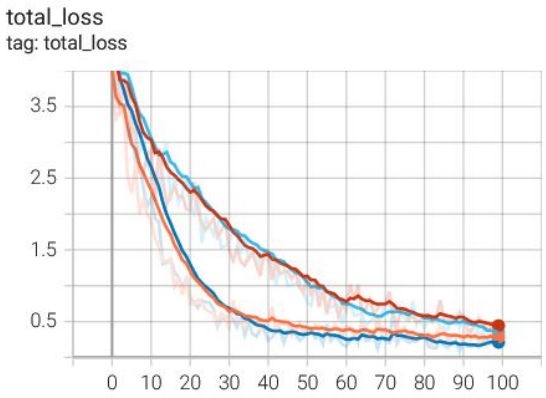


Figura 4.12: Pérdida del modelo en la detección durante el entrenamiento



Figura 4.13: Precisión en la detección durante el entrenamiento

4.3.2. Detección de árboles

Al realizar algunas detecciones de árboles usando el modelo resnet se puede observar los siguientes resultados.



Figura 4.14: Detección de árboles separados

En la figura 4.14 se observa que se detectaron los arboles más definidos pero no así el árbol que borde superior incompleto.



Figura 4.15: Detección en una zona densa de vegetación

En la figura 4.15 se observa como en una zona densamente poblada de árboles, el modelo es capaz de detectar algunos árboles.

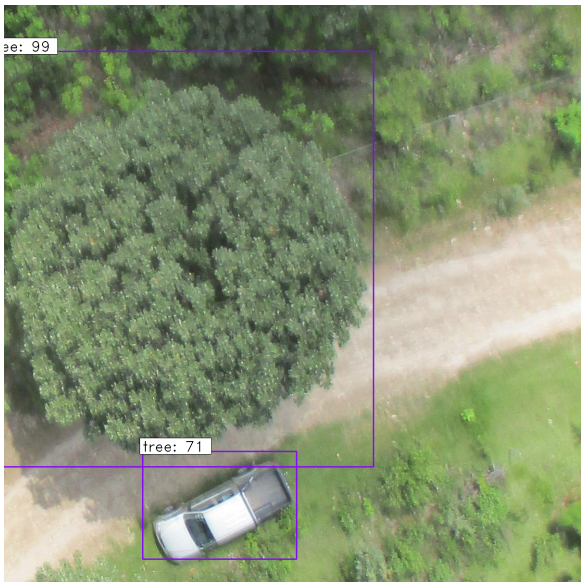


Figura 4.16: Error en la detección

En la figura 4.16 se observa un falso positivo de un coche.



Figura 4.17: Detección en una imagen extraida de Google Maps

En la figura 4.17 se realizó la detección de árboles en una imagen diferente del conjunto de datos del bosque de la primavera.

5. RESULTADOS Y DISCUSIÓN

Resumen: *Se presentan los resultados de los modelos entrenados en el capítulo 4*

5.1. Resultados del modelo

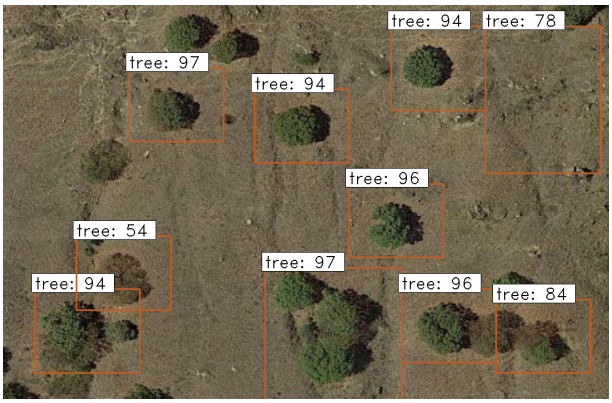
En las tabla 5.1, se muestran unas medidas de evaluación del modelo $map@X_{tree}$, donde X se refiere al límite en porcentaje IOU para considerar que el marco delimitador predicho por el modelo sea considerado correcto. También se incorpora el tiempo de inferencia del modelo, el error final y el tiempo medio transcurrido en cada época del entrenamiento.

Resultados	Resnet50	VGG16	Resnet50 Aug	VGG Aug
$mAP@50_{tree}$	0.94	0.93	0.81	0.84
$mAP@75_{tree}$	0.92	0.93	0.57	0.67
$mAP@90_{tree}$	0.49	0.37	0.33	0.30
Error final	0.28	0.21	0.44	0.34
Tiempo de inferencia[s]	1.11	0.74	1.11	0.74
Tiempo de entrenamientos por época[s]	90	78	90	113

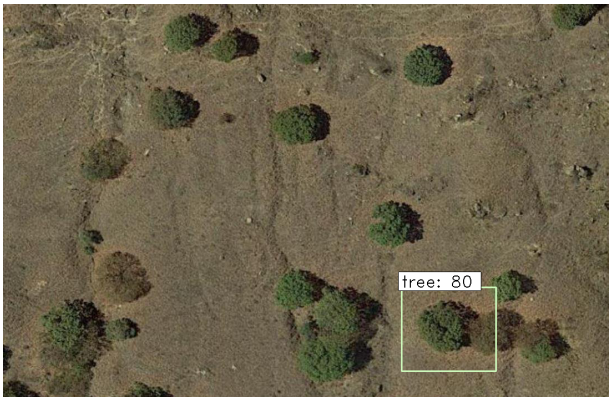
Tabla 5.1: Resultados del entrenamiento con diferentes redes de extracción de características

Resultados	Resnet50	VGG16	Resnet50 Aug	VGG Aug
$mAP@50_{tree}$	0.58	0.48	0.48	0.48
$mAP@75_{tree}$	0.39	0.32	0.29	0.28
$mAP@90_{tree}$	0.32	0.32	0.29	0.25

Tabla 5.2: Resultados del en imágenes de validación con diferentes redes de extracción de características



(a) Resnet50 Aug



(b) VGG Aug

Figura 5.1: Predicción de modelos.

En la tabla 5.3 se muestran la cantidad de parámetros total del modelo con diferentes extractores del mapa de características y el tamaño del modelo en mega bytes.

Modelo Faster RCNN	Número de parámetros	Tamaño [MB]
Resnet50	28342195	111
VGG16	135497080	529

Tabla 5.3: Parámetros y tamaño de los modelos

En la figura 5.1 se puede observar la predicción del modelo Faster R-CNN con diferentes capas de extracción de características Resnet50 y VGG sobre una imagen fue obtenida de Google Maps que no fue parte del entrenamiento.

5.2. Discusión

En los resultados de validación 5.2, el modelo Faster R-CNN usando Resnet como extractor de características, tuvo mejores resultados, aunque sobre los datos de entrenamiento

5.1 usando VGG, se tuvieron ligeramente mejores resultados que Resnet50. Como ejemplo en la figura 5.1 donde se utiliza una imagen que no se usó durante el entrenamiento, se obtienen más predicciones correcta en la detección de árboles. También cabe observar que el tamaño del modelo VGG es casi 5 veces más grande por lo que una mayor cantidad de datos puede ser necesaria para obtener mejores resultados usando ese extractor de características.

En la sección 4.3.2 se puede observar como el modelo puede detectar árboles que se encuentran separados entre sí, pero el resultado en zonas densas no es muy efectivo. Además de la detección de ciertos objetos como árboles como en la figura 4.16, esto se puede deber a la falta de más ejemplos negativos y a los pesos pre entrenados en imagenet en el mapa de características.

Al realizar la detección sobre una imagen diferente al conjunto de datos como en la figura 4.17, se puede ver como el modelo ha podido generalizar en cierto grado un árbol desde la perspectiva aérea, aunque se nota que con objetos desconocidos por el modelo que se sobrepongan sobre los árboles basta para que el modelo no los detecte.

6. CONCLUSIONES

6.1. Conclusiones

En la sección la sección 1.3 se planteó como problema el monitoreo automático de árboles a través de imágenes obtenidas por un UAV, por el cual se desprendieron los objetivos de la sección 1.4. Con los resultados de este trabajo se puede concluir que se alcanzaron los objetivos en mayor grado.

En cuánto al modelo, se identificó el detector Faster R-CNN como adecuado para localizar y clasificar los árboles, se eligió por ser un detector con buenos resultados en el conjunto de datos ILSVRC y por la mayor disponibilidad de información acerca del modelo.

Para el entrenamiento del detector se logró generar un conjunto de datos con anotaciones en las imágenes obtenidas por un UAV. En esta parte del proceso llevó más tiempo del esperado dado que las anotaciones se hicieron de forma manual con ayuda de la herramienta supervisely y la cantidad de datos generados es una muestra pequeña. Dado la poca o nula disponibilidad pública en cuanto a un conjunto de datos como los usados en este trabajo el modelo puede ser mejorado incrementando en cantidad y calidad de las anotaciones y ejemplos negativos.

Se logró entrenar el detector Faster R-CNN con resultados aceptables mostrados en la tabla 5.2 en la detección de árboles donde se pueden distinguir los bordes de los mismos y una detección menor donde el bosque es más denso como se muestra en la figura 4.15.

Se concluye de este trabajo que un modelo de aprendizaje profundo Faster R-CNN, puede ser usado para satisfacer la tarea de monitoreo de árboles en el bosque la primavera, dado que en los resultados se muestra que el modelo pudo generalizar la identificación de árboles en las imágenes aéreas. Una mejora en la calidad de las anotaciones debería mejorar las inferencias del modelo.

Dado el reto de encontrar alguna base de datos que pudiera ser de usa para este trabajo de grado y ante el acelerado proceso de adopción de modelos de aprendizaje profundo en el mundo, creo beneficioso que instituciones en el ámbito público y privado tuvieran como objetivo la generación de bases de datos públicas para la investigación y el desarrollo en universidades y centros de investigación.

6.2. Trabajo Futuro

Ante los resultados del presente trabajo, se pueden seguir una línea donde se integre el modelo en un sistema embebido en los UAV o en una aplicación web.

Realizar experimentos en otro tipo de modelos de aprendizaje profundo como los "transformersü otro tipo de detectores.

Aumentar la cantidad y calidad de los datos públicos disponibles para continuar con el desarrollo e investigación en la detección en áreas forestales.

Realizar comparativa del modelo entrenado en el presente trabajo con el estado del arte presentado en la sección 2.4 de trabajos relacionados.

BIBLIOGRAFÍA

[1] (). Ficha S I M E C | Comisión Nacional de Áreas Naturales Protegidas | Gobierno | gob.mx, dirección: https://simec.conanp.gob.mx/ficha.php?anp=35&=11 (visitado 03-05-2022).

[2] (). Conociendo Jalisco, dirección: https://iieg.gob.mx/contenido/Economia/Conociendo_Jalisco_2014.pdf (visitado 03-05-2022).

[3] S. Lambros, "ANÁLISIS, ESTRATEGIAS Y PROYECCIÓN DEL CENTRO DE RECREACIÓN AMBIENTAL SAN MIGUEL, EN EL BOSQUE LA PRIMAVERA, JALISCO, MÉXICO.", 1 de nov. de 2016. dirección: https://opac.biblio.iteso.mx/vufind/Record/000392797 (visitado 22-11-2020).

[4] L. Liu, W. Ouyang, X. Wang, P. Fieguth, J. Chen, X. Liu y M. Pietikäinen, "Deep learning for generic object detection: A survey", *Int J Comput Vis*, vol. 128, n.º 2, págs. 261-318, 1 de feb. de 2020, ISSN: 1573-1405. DOI: 10.1007/s11263-019-01247-4. dirección: https://doi.org/10.1007/s11263-019-01247-4 (visitado 28-03-2021).

[5] P. Viola y M. Jones, "Rapid object detection using a boosted cascade of simple features", 2001. DOI: 10.1109/CVPR.2001.990517. dirección: http://ieeexplore.ieee.org/document/990517/ (visitado 24-04-2021).

[6] N. Dalal y B. Triggs, "Histograms of oriented gradients for human detection", 2005. DOI: 10.1109/CVPR.2005.177. dirección: http://ieeexplore.ieee.org/document/1467360/ (visitado 24-04-2021).

[7] P. Felzenszwalb, D. McAllester y D. Ramanan, "A discriminatively trained, multiscale, deformable part model", jun. de 2008. DOI: 10.1109/CVPR.2008.4587597. dirección: http://ieeexplore.ieee.org/document/4587597/ (visitado 24-04-2021).

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg y L. Fei-Fei, "ImageNet large scale visual recognition challenge", 1 de sep. de 2014. dirección: https://arxiv.org/abs/1409.0575v3 (visitado 29-03-2021).

[9] Alex Krizhevsy, I. Sutskever y G. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", 2012.

[10] R. Girshick, "Fast R-CNN", 27 de sep. de 2015. arXiv: 1504.08083. dirección: http://arxiv.org/abs/1504.08083 (visitado 24-04-2021).

[11] S. Ren, K. He, R. Girshick y J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *arXiv:1506.01497 [cs]*, 6 de ene. de 2016. arXiv: 1506.01497. dirección: http://arxiv.org/abs/1506.01497 (visitado 24-04-2021).

[12] J. Dai, Y. Li, K. He y J. Sun, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", *arXiv:1605.06409 [cs]*, 21 de jun. de 2016. arXiv: 1605.06409. dirección: http://arxiv.org/abs/1605.06409 (visitado 24-04-2021).

[13] K. He, G. Gkioxari, P. Dollár y R. Girshick, "Mask R-CNN", *arXiv:1703.06870 [cs]*, 24 de ene. de 2018, version: 3. arXiv: 1703.06870. dirección: http://arxiv.org/abs/1703.06870 (visitado 24-04-2021).

[14] Z. Cai y N. Vasconcelos, "Cascade R-CNN: Delving into High Quality Object Detection", 3 de dic. de 2017. arXiv: 1712.00726. dirección: http://arxiv.org/abs/1712.00726 (visitado 24-04-2021).

[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus e Y. LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks", *arXiv:1312.6229 [cs]*, 23 de feb. de 2014. arXiv: 1312.6229. dirección: http://arxiv.org/abs/1312.6229 (visitado 24-04-2021).

[16] J. Redmon, S. Divvala, R. Girshick y A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection", *arXiv:1506.02640 [cs]*, 9 de mayo de 2016. arXiv: 1506.02640. dirección: http://arxiv.org/abs/1506.02640 (visitado 11-05-2020).

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu y A. C. Berg, "SSD: Single Shot MultiBox Detector", vol. 9905, 2016. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325. dirección: http://arxiv.org/abs/1512.02325 (visitado 24-04-2021).

[18] T.-Y. Lin, P. Goyal, R. Girshick, K. He y P. Dollár, "Focal Loss for Dense Object Detection", *arXiv:1708.02002 [cs]*, 7 de feb. de 2018. arXiv: 1708.02002. dirección: http://arxiv.org/abs/1708.02002 (visitado 05-06-2020).

[19] H. Law y J. Deng, "CornerNet: Detecting objects as paired keypoints", pág. 17,

[20] B. G. Weinstein, S. Marconi, S. Bohlman, A. Zare y E. White, "Individual tree-crown detection in RGB imagery using semi-supervised deep learning neural networks", *Remote Sensing*, vol. 11, n.º 11, pág. 1309, ene. de 2019, Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, ISSN: 2072-4292. DOI: 10.3390/rs11111309. dirección: https://www.mdpi.com/2072-4292/11/11/1309 (visitado 18-02-2022).

[21] M. Onishi y T. Ise, "Automatic classification of trees using a UAV onboard camera and deep learning", pág. 9,

[22] D. Choi, C. J. Shallue, Z. Nado, J. Lee, C. J. Maddison y G. E. Dahl, "On Empirical Comparisons of Optimizers for Deep Learning", *arXiv:1910.05446 [cs, stat]*, 15 de jun. de 2020. arXiv: 1910.05446. dirección: http://arxiv.org/abs/1910.05446 (visitado 04-04-2022).

[23] E. Charniak, *Introduction to Deep Learning*, 1st. The MIT Press, 2018, ISBN: 978-0-262-03951-2.

[24] A. Gerón, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd. OReilly, 2019, pág. 287, ISBN: 978-1-492-03264-9.

[25] D. E. Rumelhart, G. E. Hinton y R. J. Williams, "Learning Internal Representations by Error Propagation", 1 de sep. de 1985. dirección: https://apps.dtic.mil/sti/citations/ADA164453 (visitado 14-04-2022).

[26] A. Gerón, *Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow*, 2nd. OReilly, 2019, pág. 491, ISBN: 978-1-492-03264-9.

[27] J. Schott, *Remote Sensing: The Image Chain Approach*, 2nd. New York: Oxford University Press, 2007, ISBN: 9780195178173.

[28] Y. LeCun, L. Bottou, Y. Bengio y P. Haffner, "Gradient-Based Learning Applied to Doument Recognition", *Proc. of the IEEE*, 1998.

[29] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick y P. Dollár, "Microsoft COCO: Common Objects in Context", arXiv, arXiv:1405.0312, 20 de feb. de 2015, type: article. DOI: 10.48550/arXiv.1405.0312. arXiv: 1405.0312[cs]. dirección: http://arxiv.org/abs/1405.0312 (visitado 14-05-2022).

[30] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn y A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective", *International Journal of Computer Vision*, vol. 111, n.º 1, págs. 98-136, ene. de 2015.

[31] you359, *TensorFlow Model Faster R-CNN*, https://github.com/you359/Keras-FasterRCNN, 2020.

[32] K. Simonyan y A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", *arXiv:1409.1556 [cs]*, 10 de abr. de 2015. arXiv: 1409.1556. dirección: http://arxiv.org/abs/1409.1556 (visitado 11-05-2020).

Apéndice A

$\mathbf{A}$	matriz
A_i^j	elemento *(i,j)* de $\mathbf{A}$
$\mathbf{x}, \mathbf{v}, \mathbf{w}$	vectores
$x_i, v_i.w_i$	elemnto ith de un vector
$\mathbf{w}^t$	vector t de un conjunto de vectores