INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE OCCIDENTE

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

Departamento de Electrónica, Sistemas e Informática

MAESTRÍA EN SISTEMAS COMPUTACIONALES

DESARROLLO DE UN MÓDULO DE VISIÓN COMPUTACIONAL PARA VEHÍCULOS AEREOS NO TRIPULADOS UTILIZANDO DEEP LEARNING

Trabajo recepcional que para obtener el grado de
Maestra en Sistemas Computacionales

Presenta: José Luis Magaña Vázquez

Asesor: Dr. José Francisco Cervantes Álvarez

Co-Asesor: Mtro. Luis Eduardo Pérez Bernal

Tlaquepaque, Jalisco a 2 de marzo de 2020

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

1.1. Antecedentes

El proyecto inicia en 2017 con el desarrollo de un sistema de información con el propósito de controlar las misiones de un vehículo autónomo no tripulado (UAV), así como la recepción y procesamiento de diferentes fuentes de información obtenidas por los UAV; fotografías, lecturas de sensores, etc. Para atender aplicaciones específicas tales como: ortofotos, NDVI, calidad del aire, salud de los árboles, prevención de desastres naturales y permitir la toma oportuna de decisiones.

1.2. Justificación

El Bosque La Primavera es uno de los ecosistemas más importantes de la ciudad de Guadalajara en México. A pesar de esto, el bosque ha sido afectado por factores como invasiones,construcciones ilegales, incendios, así como una alta visitación por parte de los usuarios que ha ido aumentando exponencialmente en los últimos años. Algunos de estos efectos se reflejan en el deterioro de los suelos, caminos y brechas, así como la pérdida de vegetación y fauna silvestre que junto con otros problemas relacionados a los proyectos logísticos de conectividad interestatal, ponen en riesgo la integridad del bosque [1]. Los incendios, con el porcentaje de hectáreas quemadas en la última década, se han convertido en una de las principales amenazas junto con el desarrollo territorial, debido a la relación que existe entre estos dos factores. A partir del trabajo de análisis en el bosque La Primavera [1] se desprenden ciertas propuestas iniciales; dentro de estas propuestas

Figura 1.1: Zona de invasiones en BLP. Fuente [1]

se encuentran el monitoreo de la vegetación y el crecimiento urbano como punto inicial para generar un diagnóstico del cuál se puedan focalizar acciones de acuerdo al problema que se pueda desprender del analisis (ejemplos: invasión del bosque, áreas deforestadas, zonas con altos niveles de combustibles naturales).

1.3. Problema

Por lo que el problema a abordar es parte del monitoreo del bosque La Primavera. Actualmente se cuenta con un área en el bosque propiedad del ITESO (revisar si este es estado jurídico de esta área) en donde se toman fotografías aéreas a través de vehículos aéreos no tripulados. Parte de este trabajo se centra en la localización de la vegetación, zanjas y construcciones que ayude a esta parte del monitoreo. Para ello se busca implementar uno o varios algoritmos de aprendizaje profundo para localizar objetos de interés durante el vuelo de un vehículo aéreo no tripulado en un área en el bosque de la primavera. Los objetos de interés en este trabajo son los siguientes:

- Árboles
- Zanjas de protección

- Construcciones

La tarea a llevar a cabo se pueden descomponer en varios subproblemas:

- Clasificación - Asignación de una clase a un objeto.
- Localización - Ubicar si hay o no un objeto de interés.
- Detección de Objeto - Clasificación y localización del objeto de interes.

En el capítulo 2 se presentan diferentes trabajos que presentan modelos de aprendizaje profundo que lidian con estos subproblemas. Dichos modelos que se presentaran, son modelos supervisados; los cuales necesitan una base de datos con etiquetas de los objectos de interés. Por lo que al usar este tipo de modelos se presentan otro tipo de problemas a abordar tales como:

- Generar una base de datos con etiquetas.
- Preprocesamiento de los datos de entrada al modelo.
- Uso de técnicas de aumento de datos.
- Entrenamiento de modelo.
- Métricas de evaluación del modelo.

1.4. Hipótesis

Se puede utilizar modelos de aprendizaje profundo para localizar los objetos de interes en imágenes obtenidas por un vehículo aéreo no tripulado que forme parte del monitoreo al bosque La Primavera.

1.5. Objetivos

1.5.1. Objetivo general

El propósito principal de este trabajo de obtención de grado es desarrollar un modulo de visión computacional basado en técnicas de Deep Learning que permita al vehículo aereo no

tripulado identificar objetos de interés durante su vuelo.

1.5.2. Objetivo específico

Obtener una base de datos con los objetos de interés para el entrenamiento del modelo. Entrenar un algoritmo basado en aprendizaje profundo para detectar objetos de interés. Implementar el algoritmo en un modulo que pueda realizar la detección durante el vuelo.

1.6. Novedad científica

Capítulo 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

A continuación se presenta una colección de trabajos que han aportado al estado del arte en la detección de objetos; en éste capítulo se han dividido los trabajos de acuerdo a los problemas planteados en el caítulo 1, que son la clasificación y detección de objectos(clasificación y localización). Una vez vistos los modelos de redes convolucionales para la detección de objetos se revisan algunos trabajos relacionados al problema planteado aquí, que incorporan a dichos modelos para la detección de vegetacióna través de información captada por vehículos aéreos no tripulados.

2.2. Clasificación

Está sección contiene trabajos con redes neuronales clásicas y redes neuronales convolucionales cuya función es solamente la de clasificar. Si bien la meta es la implementación de un modelo de detección de objetos, estos modelos de clasificación sirven de base o como parte de modelos más

complejos en la detección de objetos.

2.2.1. Aprendizaje basado en gradiente descendiente aplicado al reconocimiento de documentos - LeCun

En éste artículo se nos muestra que dada una apropiada arquitectura de red neuronal, los algortimos de aprendizaje basados en el gradiente descendiente pueden ser usados para sintetizar un clasificador, tal como se muestra en su artículo, la red presentada apoya para la clasificación de caráteres escritos. La red utilizada, es una red neuronal convolucional las cuales tienen un gran desempeño en clasificación de imágenes. La arquitectura de la red neuronal convolucional presentada se le llama LeNet-5 [2] que consta de 3 capas convolucionales, 2 capas de muestreo y 2 capas completamente conectadas. El método usado para la actualización de los parámetros es el método de gradiente descendiente estocástico. Durante el desarrollo del proyecto se utilizara el tipo de arquitectura de red convolucional para la clasificación de los objetos de interés, así como se aplicarán ciertas modificaciones de artículos más recientes.

2.2.2. Clasificación ImageNet con redes neuronales convolucionales profundas - Krizhevsky

En este artículo Krizhevsky presenta con una red nueronal convolucional tal y como lo hace LeCun, la aportación de este trabajo es la introducción de un método de regularización para prevenir el sobre ajuste de la red, este método de regularización [3] es llamado "dropout". Este método consiste en forzar el valor a 0 a las neuronas en capas ocultas con una probabilidad de 0.5. De ese modo dichas neuronas no contribuyen a la clasificación haciendo que el entrenamiento no dependa de las mismas neuronas para realizar la clasificación y por lo tanto obliga a la red a aprender caractisticas de manera más robusta. De esta manera se evita el sobre ajuste, pero el costo es que se requiere mayor cantidad de entrenamiento para que la red neuronal converja. Durante el trabajo utilizaremos técnicas de normalización como el presentado por Krizhevsky, para evitar el sobre ajuste.

2.2.3. Red en Red - Lin

En el atículo presentado por Lin et al, se introduce una modificación a la red neuronal convolucional. En la capa de convolución normalmente se utilizan kernel locales para obtener los mapas de carecterísticas, en su lugar en éste trabajo se propone reemplazarlo por una red neuronal basados en el perceptrón [4], para ser el responsable de obtener los mapas de caracteríscas. Además agrega un método de regularización llamado "Global average pooling.º muestreo promedio global el cual toma el promedio de cada mapa de características el cuál se pasa a una capa "softmax", con cual los autores describen que refuerza las correspondencias entre los mapas de características y categorías, y es más robusto en translaciones espaciales del objeto de interés.

2.2.4. Redes convolucionales muy profundas para reconocimiento de imágenes a grande escala - Simonyan

Hasta ahora en los trabajos se han introducido cambios en las capas de convolución, pero qué tal aumentar o disminuir la profundidad de la red. En cuanto al manejo de la profundidad de la red neuronal, Simonyan y Andrew presentaron en su artículo el efecto de hacer redes más profundas y su precisión al modificarlas, con profundidad de hasta 19 capas, en su investigación [5] arrojaron que se puede aumentar la precisión haciendo más profunda la red logrando tener buenos resultados y disminuyendo la dimensión de filtros convolucionales o kernels locales, en este caso de dimensión 3x3.

2.2.5. Aprendizeje residual profundo para reconocimiento de imágenes - He

Siguiendo con los trabajos que experimentan con redes neuronales más profundas, He et al, que expone que diseñar redes neuronales muy profundas lleva a una degradación durante el entrenamiento donde la precisión se satura y comienza un degradamiento. Como parte de enfrentar este comportamiento, He et al, introducen un nuevo tipo de bloque a las redes neuronales llamado aprendizaje residual [6], el cual se tienen dos capas, la primera capa procesa normalmente la información de entrada, pero la segunda capa, toma como entrada, los datos de salida de la primera capa más los datos de entrada de la primera capa. Con esto se muestra que pueden diseñar redes con

mayor porfundidad y manejando el degradamiento que habían observado con las redes neuronals habituales. Con este tipo de bloque en la red neuronal, experimentaron con profundidades de hasta 152 capas a diferencia de las 19 capas usadas por Simonyan, este tipo de red se llamó ResNet.

2.2.6. Iendo profundo con convoluciones - Szegedy

En este trabajo Szegedy et al, introducen un nueva arquitectura llamada Ïnception" [7], esta arquitectura se basa en un modulo de ïnception", en este modulo todos los kernels son aprendidos y que es un conjunto de filtros de diferentes dimensiones que procesan la entrada y al final se combinan como preparación para el siguiente módulo de ïnception".

2.2.7. Agregar VGG

2.3. Localización y clasificación

Como parte de los problemas relacionados en la visión computacional, el problema que se aborda en este proyecto no solo es la clasificación sino también la localización del objetos dentro de la imagen. Existen varias implementaciones para abordar estas tareas en conjunto, al igual en la sección de **Clasificación**, estos trabajos son de interés como apoyo para la posterior descripción de trabajos específicos sobre el problema que se aborda en cuanto a la detección de objetos utilizando información de un vehículo aéreo no tripulado.

2.3.1. Solo observas una vez: Unificado, detección de objetos en tiempo real - Redmon

Parte del manejo de imágenes no solamente es la clasificación sino también la localización en un imagen de varios elementos de interes. En este trabajo Redmon et al, presenta una arquitectura que diferente a lo que se había hecho en trabajos previos donde la detección de objetos se han dividido en dos etapas, la localización y la clasificación. Redmon et al, propone una sola etapa en su arquitectura YOLO [8]. Arquitecturas previas parten la imagen en varias regiones, donde genera cuadros delimitadores potenciales para ejecutar la clasificación sobre estas cajas protenciales. Al

final un post procesamiento es realizado para refinar los cuadros delimitadores, eliminar duplicados de imagen y generar las clasificaciones finales. YOLO predice simultaneamente multiples cajas delimitadoras y clases de probabilidad de cada una. Esta arquitectura es entrenada para en imagenes completas para la optimización. Por lo que aprende del objeto de interés y su contexto, lo que lo lleva a generar menos falsos en imagenes que no son de interés. En contra tiene que al generar menos cajas delimitadoras y tiene un límite de predicciónes por cada una, va por detrás de otros modelos en la detección de objetos pequeños.

2.3.2. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks - Girshick

En el trabajo se presenta un modelo mejorado de Fast-RCNN, el cual a partir propuestas de regiones de interés dadas por un algoritmo llamado "selective search.º busqueda selectiva se ingresan a una red neuronal llamada Fast-RCNN que da como resultado la ubicación y clasificación de los objectos de interés. Como mejora en éste modelo, se reemplaza el algoritmo "selective searchçon otra red neuronal convolucional llamada RPN o Region Proposal Networks", la cual mejora las regiones de interés que luego seran consumidas en la red Fast-RCNN. Este tipo de detectores son llamados de dos etapas, ya en una primera etapa se generan las regiones de interes, luego se clasifica y localiza en una etapa posterior. La introducción de la nueva red RPN que genera las regiones de interés introduciendo un concepto de anchor box, que es una caja delimitadora de cierto tamaño y aspecto fijo que se usan para poder entrenar al la red RPN.

2.3.3. SSD: Single Shot Multibox Detector - Liu

2.4. Trabajos relacionados

2.4.1. Automated Detection of Conifer Seedlings in Drone Imagery Using Convolutional Neural Networks - Fromm

2.4.2. Very High Density Point Clouds from UAV Laser Scanning for Automatic Tree Stem Detection and Direct Diameter Measurement - Karel

2.4.3. A New Individual Tree Crown Delineation Method for High Resolution Multispectral Imagery - Lin

2.4.4. Individual Tree Detection in a Eucalyptus Plantation Using Unmanned Aerial Vehicle (UAV)-LiDAR - Picos

2.4.5. A Self-Adaptive Mean Shift Tree-Segmentation Method Using UAV LiDAR Data - Wanquian

Capítulo 3

Marco Teórico Conceptual

Resumen: *En este capítulo se resumen algunos conceptos utilizados en los modelos de aprendizaje profundo que seran aplicados en los modelos desarrollados en el capítulo 4*

3.1. Aprendiendo a partir de datos

Existen varios acercamientos de aprendizaje máquina automático, pero uno de los acercamientos más exitosos ha sido el llamado aprendizaje basado en el gradiente. El modelo de aprendizaje calcula una función $\mathbf{y}^p = f(\mathbf{z}^p, \mathbf{W})$[3] donde $\mathbf{z}^p$ es la p-th patrón de entrada, $\mathbf{W}$ representa la colección de parámetros ajustables en el sistema y $\mathbf{y}^p$ puede ser interpretada como la etiqueta o clase de la entrada $\mathbf{z}^p$, o la probabilidad asociada a cada clase a predecir. Es decir que a partir de una entrada se predice la clasificación deseada por medio de una función f.

$E^p = D(\mathbf{d}^p, f(\mathbf{z}^p, \mathbf{W}))$ [3], es una función que mide la discrepancia o error entre $\mathbf{d}^p$ el cuál es el valor correcto que deseamos predecir a partir del patrón de entrada $\mathbf{z}^p$ y la predicción $\mathbf{y}^p$. Esto implica tener una base de datos con los patrones de entrada con su respectiva salida correcta para poder realizar esta operación con la cuál se entrena el modelo. Con el cálculo del error de cada predicción se puede calcular el promedio y obtener un error de entrenamiento $E_{train}(\mathbf{W})$ que se llamará función de costo o pérdida. Así muchos algoritmos de aprendizaje tratan de minimizar $E(W)$.

3.2. Gradiente descenciente

El algoritmo de gradiente descendiente intenta minimizar la función de costo $E(W)$ a través de variaciones en los pesos W. Estas variaciones son introducidas a cada peso partir del gradiente de la función de perdida con respecto a los pesos. Pero para ello se tiene que cumplir que los parámetros de W son valores reales para los cual la función de costo es una función continua y diferenciable, de esta manera se introducen variaciones en cada parámetro de W iterativamente de la siguiente forma: $W_{(i,j)} = W_{(i,j)-1} - \varepsilon \frac{\partial E(\mathbf{W})}{\partial \mathbf{W}}$[3] Existen variaciones del gradiente descendiente siendo el presentado uno de los más simples.

3.3. Perceptrón

El perceptrón es un modelo computacional de una neurona[4], donde una sola neurona tiene varias entradas (dentritas), un cuerpo celular y una salida (axón). Replicando esto el perceptrón tiene varias entradas y una sola salida, el cuál consiste en un vector de "pesos" w = [w1 ... wm], un peso para cada entrada más un "bias" o tendencia cuyo parámetro es descrito como b. Con los parámetros dados w y b el perceptrón realiza el siguiente cálculo:

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

$$f(x) = 0 \qquad de\ otra\ manera$$

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 a^k, donde a^k será 1 o 0 en este caso. A continuación, se presenta el algoritmo del perceptrón[4]:

1. Inicializar los parámetros w y b

2. Durante N iteraciones (definidas por el desarrollador) o hasta que los pesos no cambien realizar lo siguiente:

   a) Para cada ejemplo de entrenamiento x^k con respuesta a^k

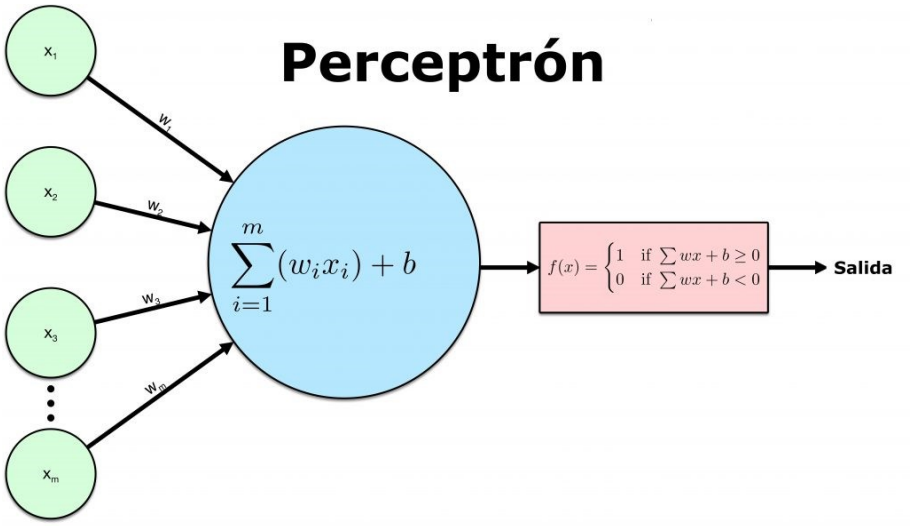

Figura 3.1: Modelo del perceptrón

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

$$w_i = w_i + \left(a^k - f\left(x^k\right)\right) x_i$$

Donde si existe una configuración de pesos w con los cuales se puedan clasificar correctamente cada elemento de entrenamiento, el algoritmo lo encontrará, aunque en ejemplos reales no siempre es posible, el algoritmo puede encontrar una configuración de pesos w para clasificar correctamente un porcentaje de los elementos de entrenamiento. En casos donde se necesite más clases se puede realizar incrementando la cantidad de perceptrones por cada clase, siendo el valor 0 el indicador de no pertenencia a la clase y el número 1 la pertenencia a la clase, teniendo así una llamada red neuronal.

3.4. Red Neuronal Clásica

Cómo se ha visto con el perceptrón, una red neuronal clásica consiste en conjunto de modelos computacionales llamadas neuronas, las cuales tienen parámetros ajustables w y "bias" que, por medio de un algoritmo, la red puede aprender a clasificar correctamente las clases a partir

de los datos de entrada x^k. En la Figura 3.2 se puede observar un modelo de red neuronal, cuyas

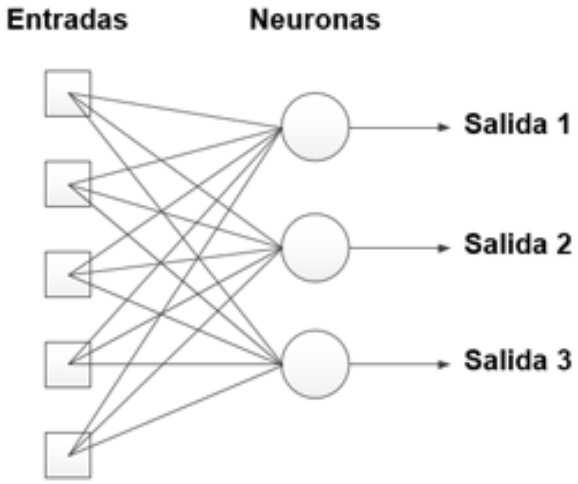

Figura 3.2: Modelo de una red neuronal de una sola capa

neuronas tienen una salida. Ahora, las salidas de las neuronas en la imagen podríamos tomarlas como entrada para un siguiente conjunto de neuronas, a cada conjunto de neuronas se les llama capas, siendo una capa las neuronas que reciben la entrada original, una segunda capa de neuronas como entrada la salida de la primera capa, teniendo una llamada red neuronal multicapa, donde a las capas entre la entrada y la salida se les llama capas ocultas como se presenta en la Figura 3.3. De esta manera se pueden obtener redes neuronales de n capas, por lo que cada capa aumenta el tamaño o la "profundidad" de la red neuronal, y es aquí de donde procede el termino aprendizaje profundo o también conocido en inglés como "Deep learning".

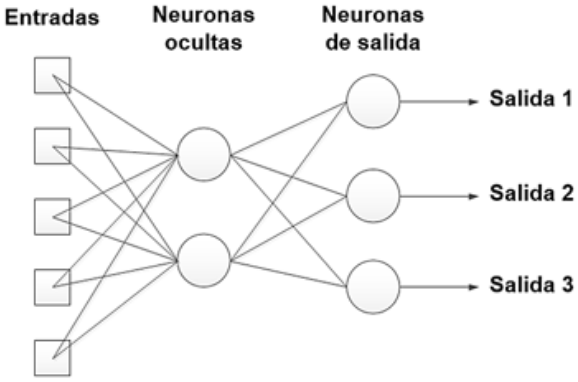

Figura 3.3: Modelo de una red neuronal con una capa oculta

3.5. Red Neuronal Feed Fordward y Back Propagation

Son redes neuronales cuyo grafo con dirección no contienen ciclos, normalmente las redes neuronales multicapa utilizan un algoritmo de gradiente descendiente para calcular el valor de modificación de los pesos w, donde la diferencia de las neuronas del tipo perceptrón es el cambio de la función de activación, donde la función de activación del perceptrón está dada por:

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

$$f(x) = 0 \qquad de\ otra\ manera$$

Esto debido que, para hacer uso del gradiente descendiente necesitamos que la función de costo o pérdida sea diferenciable y continua para el conjunto de valores reales de los pesos w. Dado que la función de pérdida $E^p = D(D^p, F(W, Z^p))$[3] depende de la función de activación, la función del perceptrón no cumple los requisitos para implementar el gradiente descendiente. Existen varias funciones de activación, una de las funciones usadas para las redes neuronales "feed forward" es la función sigmoide:

$$f(z) = \frac{1}{1 + e^{-z}}$$

La cual es derivable:

$$f'(z) = f(z)(1 - f(z))$$

Donde

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

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 "back propagation".

3.6. Red Neuronal Convolucional

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.[4] 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 reduciendo la sensibilidad de la red al desplazamiento y las distorsiones de la imagen entrada. En la Figura 3.4 se presenta un ejemplo de una red convolucional llamada LeNet-5 [3] en

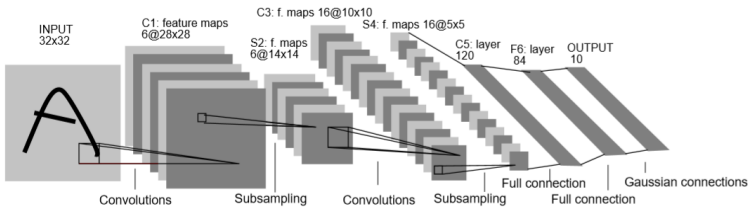

Figura 3.4: Modelo red neuronal LeNet-5

donde se puede observar 3 capas convolucionales C1, C3 y C5, dos capas de agrupación S2 y S4 y una capa completamente conectada.

3.7. RetinaNet

En este trabajo, Lin et al, proponen una nueva función de costo llamado "Focal Loss" [9] que agrega un factor al criterio de entropía cruzada estándard. Dicho factor, cuando es mayor a 0 reduce la función de costo en los ejemplos bien clasificados, dandole mayor enfoque en los ejemplos más díficiles de clasificar. En el desarrollo de este trabajo muestra una mejora en la precisión en la detección de objetos densos. La función de costo es ejercitada en un detector de objetos llamado R-CNN. Para evaluar su función de costo, entrenan a un detector de densidad llamado RetinaNet".

Capítulo 4

Desarrollo Metodológico

Resumen: *A continuación se presentan las actividades desarrolladas en el trabajo de grado para lograr implemetar modelos de aprendizaje profundo supervisado, el código de la implementación puede ser encontrado en* https://github.com/mvjluis/TOG

4.1. Datos

4.1.1. Proceso de obtención 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 4.1) se generó una ruta de vuelo en el cual se van tomando imágenes cada segundo, se tienen imágenes de 20MP en formato de colores sRGB 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 4.2. El recorrido del vehículo aéreo no tripulado es a una velocidad de 5m/s y una altura de 80m.

4.1.2. 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

Figura 4.1: Polígono del bosque La Primavera. Fuente Google Earth

Figura 4.2: Foto aérea tomada por vehículo aéreo no tripulado en bosque de la primavera.

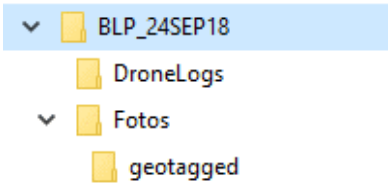

Figura 4.3: Estructura de carpetas de los datos obtenidos

“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).

4.1.3. Proceso de preparación de los 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.

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 los necesarios para poder entrenar un modelo. Existen varias herramientas para el etiquetado e incluso hay empresas que prestan servicios completos desde la colección de datos, 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 4.4: Visualización de la herramienta Supervisely

Hay diferentes formas de guardar las anotaciones, entre las bases de datos más usadas como son: COCO Common Objects in COntext, 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” Pascal Visual Object Classes (VOC): 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. Modelos

Para el trabajo de grado se planteó utilizar los modelos Faster R-CNN, SSD(Single Shot Detector), a continuación se describe el desarrollo de cada uno.

4.2.1. Faster R-CNN

En el árticulo Faster R-CNN se describe una arquitectura de modelo de un detector de la siguiente forma: En una primera etapa, se introdujo como mejora 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, esta segunda red es llamada Fast R-CNN. Así el modelo de la red RPN y Fast R-CNN fue llamado Faster R-CNN. Debido a este tipo de implementación, se dice que es un detector de dos etapas. Ahora se ahondará

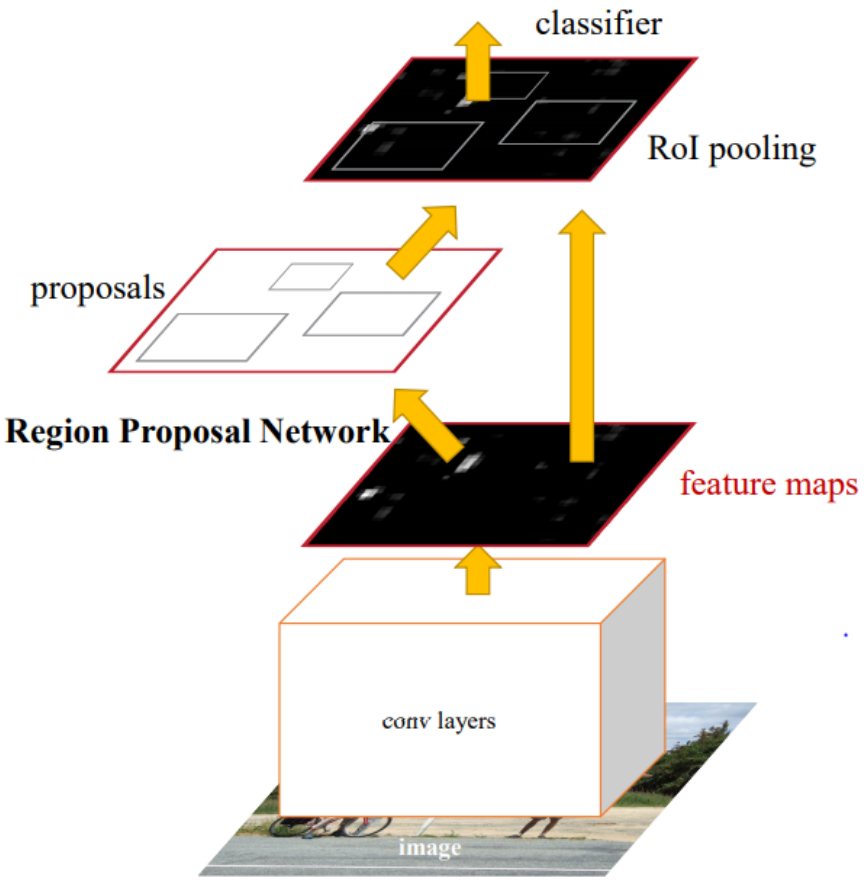

Figura 4.5: Modelo Faster R-CNN

en los detalles de las ideas e implementación del modelo. Las imágenes de entrada que se usaran son del tamaño 800x800 pixeles y 3 canales(RGB).

Mapa de características

Primero comenzamos con la red RPN (Region Proposal Network). Como se observa en la figura 4.5 se tiene primero una generación de mapas de características o "feature maps.ª partir de de la imagen de entrada. En el artículo, este mapa de carácterísticas se obtiene usando una red VGG16 sin las últimas capas densas como se ve a continuación en la figura 4.6. La salida del mapa

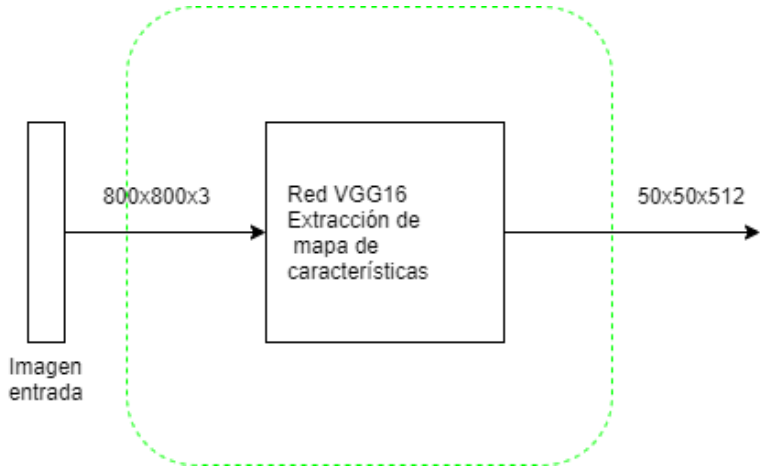

Figura 4.6: Extracción de mapa de características

de carácterísticas es de 50x50x512, lo que significa que se tienen 512 mapas, si tomamos un solo mapa de 50x50 lo podemos ver como un submuestro de la imagen original de tamaño 800x800 a 50x50, y cada pixel(es la forma correcta de decirlo?) del mapa de características representaría a 16x16 pixeles de la imagen original.

Red de propuesta de regiones o RPN

Una vez obtenidos se pasan a una red neuronal convolucional que se divide en dos ramas, en una rama predecirá si en esa región hay o no un objeto de interés y en otra la predicción de la ubicación de dicha región. Tomando en cuenta esto, si se imagina que por cada pixel del mapa de 50x50 en una rama se generan predicciones de si es una region de interés o no representado como una tupla donde por ejemplificar [1, 0] representa que no es una región de interés y [0, 1] sí es una

región de interés. Suponiendo esto, se tendrían 50x50x2 = 5000 valores predecidos en un arreglo de la forma (2500,2). De igual forma en la otra rama se tendrían que predecir 4 coordenadas de la ubicación del región, por lo que se tendría 50x50x4 = 10000 valores predecidos en un arreglo de la forma (2500,4). Allí se terminaría la red llamada Red de propuesta de regiones o RPN (por sus siglas en inglés Region Proposal Networks"), pero con un detalle más, los autores del modelo no solo predicen 1 región por cada pixel del mapa de características, se predicen 9 regiones por cada pixel. A continuación se muestra la razón de este número. Un problema al que se enfrenta un detector de objetos es la escala, ejemplificando, si se tuviera una imagén donde el objeto de interés fuese un gato, se esta de acuerdo que lo que se esperaría de un detector es que independientemente del tamaño (un gato en la imagen ya sea que apareciera en todo el espacio de la imagen o si el gato apareciera en una pequeña región), el gato fuera detectado como se muestra en la figuras 4.7 y 4.8. 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 predicciónes 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 4.9 se puede observar un ejemplo de los aspectos de los marcos que se esperan predecir y en la figura 4.10 se muestra el modelo de red convolucional que realiza las predicciones a partir del mapa de características. (Agregar imagen del modelo RPN).

Figura 4.7: Objeto de interés abarcando la mayor cantidad de la imagen

Figura 4.8: Objeto de interés abarcando una pequeña cantidad de la imagen

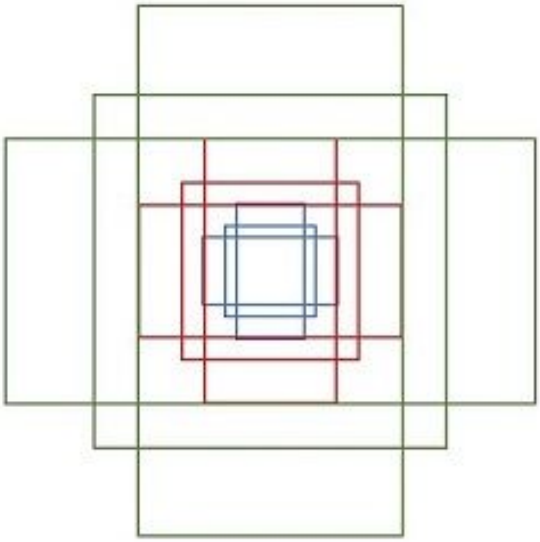

Figura 4.9: Marcos ancla

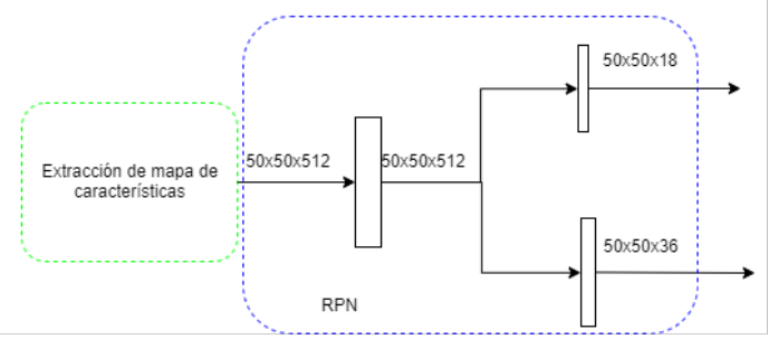

Figura 4.10: Red de propuesta de regiones

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 predecidas. Hay que recordar que el modelo es supervisado y se tienen que generar 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 predecidos por la primera etapa del RPN. Por lo tanto necesitamos generar estos marcos que serviran 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 como se muestra en la figura(referencia a la imagen con los centro ancla). 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 númerica 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 platicado del 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 por sus siglas en inglés Ïntersection Over Union") con cada marco delimitador objetivo. Es decir si la imagen contiene 4 marcos delimitadores de los objetos de interés que se quieren

predecir, se tendran 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 interes. 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, clasificandolos 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, clasificandolos 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), dado el desbalance entre los marcos ancla con etiqueta 0 y los marcos ancla con la etiqueta 1, los autores siguen 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 los marcos ancla con etiqueta 1 y se toman los marcos ancla con etiqueta 0 que sean necesarios para completar los 256. Por útlimo 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 hemos visto la interpretación de la salida de la rama de regresión(rama que predice la ubicación del marco delimitador objetivo). Al inicio suponiamos que la salida de esta rama nos 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 parámetrización 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 predecido 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.

Función de costo de la red RPN

La función de costo se define como:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*)$$

$$+\lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

Donde i es el indíce del marco ancla dentro de los 256 marcos tomados para el entrenamiento y p_i es la probabilidad del marco i predecido por la red RPN de ser un objeto. La etiqueta p_i^* del marco ancla como vimos 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 coordenas parametrizadas del marco predecido, 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 soslo 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. En el caso de λ es un parámetro para balancear los dos terminos y donde el valor predeterminado es 10.

Preparación de las propuestas para la segunta etapa

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 por lo que se realiza un procesamiento de estos datos para pasar a la siguiente red llamada Fast R-CNN solo una muestra de 128 marcos predecidos, estos marcos son llamados regiones de interés o ROI por sus siglas en inglés (Region of Interest). En la siguientes secciones se muestra el procedimiento para obtener esta muestra.

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 por sus siglas en inglés (No Maximum Suppression) figura 4.11.

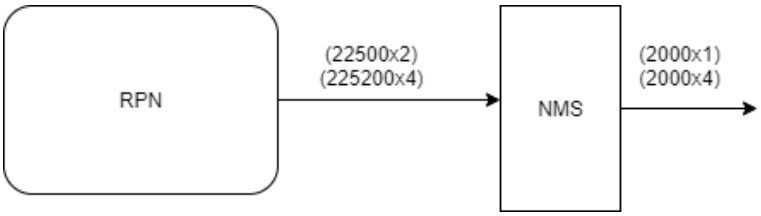

Figura 4.11: Supresión no máxima

1) Se toman las coordenadas parametrizadas de los marcos predecidos 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 tenemos las predicciones en el rango de las dimensiones de la imagen real.

3) Se descartan aquellos marcos predecidos 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 predecidos 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 predecidos. Una vez realizado este procedimiento, muchos marco habran sido descartados y solo quedarán aquellos con mayor representación de ciertas áreas.

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

Muestreo de regiones de interés

Una vez obtenidos los 2000 marcos sin traslape después de pasar por la capa de NMS, se toman aleatoriamente 128, de las que el 25 % serán muestras positivas y el 75 % restante serán muestras negativas, figura 4.12.

Figura 4.12: Muestreo

Fast R-CNN

Con las 128 muestras se pasan al través de la red Fast R-CNN. Esta parte del modelo refina las coordenadas del marco predecido 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 4.13.

Función de costo de la red Fast R-CNN

La función de costo de la red Fast R-CNN es igual a la de la red RPN, solo que esta toma con datos para calcular la salida de la red Fast R-CNN.

4.2.2. SSD

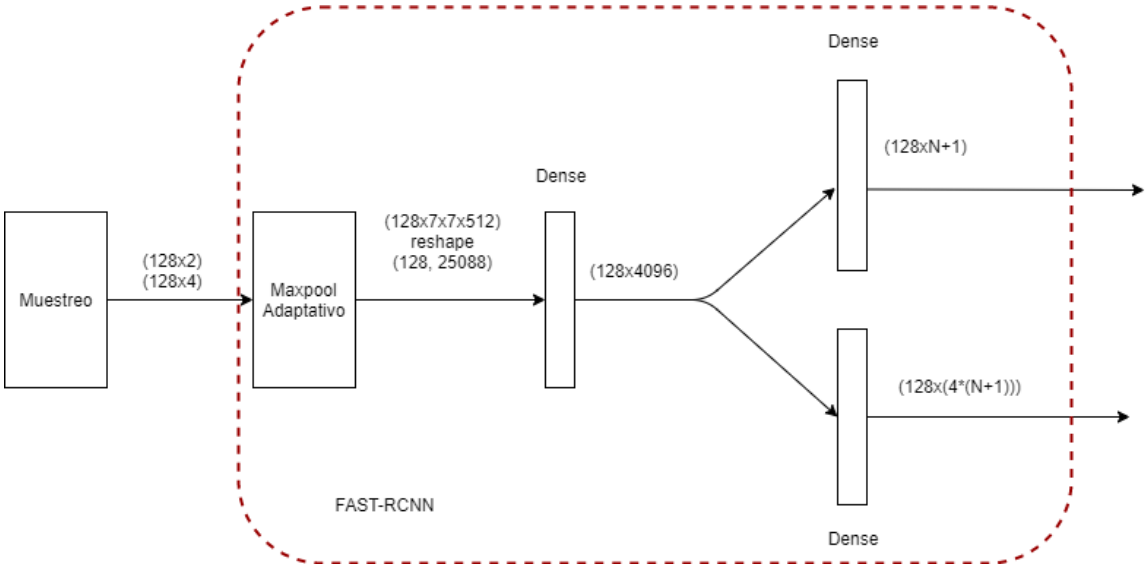

Figura 4.13: Fast RCNN

Capítulo 5

Resultados y Discusión

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

Apéndice A

Notación

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

Bibliografía

[1] S. Lambros, "AnÁlisis, estrategias y proyecciÓn del centro de recreaciÓn ambiental san miguel, en el bosque la primavera, jalisco, mÉxico." [Online]. Available: https://opac.biblio.iteso.mx/vufind/Record/000392797

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to doument recognition."

[3] Alex Krizhevsy, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks."

[4] M. Lin, Q. Chen, and S. Yan, "Network in network." [Online]. Available: http://arxiv.org/abs/1312.4400

[5] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition." [Online]. Available: http://arxiv.org/abs/1409.1556

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition."

[7] C. Szegedy, W. Liu, and Y. Jia, "Going deeper with convolutions."

[8] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection." [Online]. Available: http://arxiv.org/abs/1506.02640

[9] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection." [Online]. Available: http://arxiv.org/abs/1708.02002