Identificación de aves de la Sabana de Bogotá.

Autores:
Camila Alejandra Alvarez Rengifo
Francisco Andrés Diago Pérez

Director:
Francisco Calderón Bocanegra Ph.D

Pontificia Universidad Javeriana
Facultad de Ingeniería
Departamento de Ingeniería Electrónica
Bogotá D.C
Mayo 2019

TABLA DE CONTENDO

TABLA DE FIGURAS

1. INTRODUCCION

En la actualidad los trabajos de investigación y desarrollo en aprendizaje profundo han adoptado una postura innovadora, no solo para mejorar las tecnologías de inteligencia artificial y aprendizaje de máquina [1], Si no que también están surgiendo nuevas ideas en distintas disciplinas como la salud, la economía o el cambio climático para tener una entrada a los negocios de nuevas tecnologías. La Inteligencia Artificial (IA) y el aprendizaje profundo son tecnologías que están revolucionando y convirtiéndose en parte esencial del futuro, el dominar estos conceptos junto a otros como el aprendizaje de máquina, causan un amplio campo de acción para desarrollar e investigar en estas nuevas tecnologías.

En el informe Reshaping Business with Artificial Intelligence, elaborado por The Boston Consulting Group (BCG) y por la MIT Sloan Management Review [1], se destaca la trascendencia positiva que tiene la IA en la sociedad presente y futura, siendo un número muy corto de empresas que la incorporan en sus planes estratégicos. Es decir que existe una brecha excesiva entre la percepción de lo que vendrá y la puesta en ejecución de modelos de gestión que tengan en cuenta el posible impacto de la IA.

En todo este campo de IA se encuentran las redes neuronales artificiales (ANN) son modelos computacionales de las redes neuronales biológicas, las cuales tratan de simular su funcionamiento y su capacidad para procesar información [2]. Para este proyecto dichos datos representan características del patrón a clasificar o detectar, sin embargo, en deep learning, o aprendizaje profundo, el objetivo es que los mismos algoritmos extraigan características del objeto sin hacer un procesamiento previo a partir de redes neuronales artificiales (ANN), que constan de varias topologías que permiten diferentes características de aprendizaje.

En este caso, este trabajo de grado se desarrolló con el fin de identificar a partir de imágenes el orden y familia taxonómico de las aves de mínimo 13 especies de 12 familias diferentes que habitan en la Sabana de Bogotá usando ANN de topología convolucional, por lo que se ha diseñado algoritmos en software para la implementación de dicha topología de ANN proporcionando experiencia en la implementación de arquitecturas de redes neuronales, escogiendo la que mejor identifique la familia taxonómica del ave. Una de las arquitecturas es YOLO (you only look once) [3], y la otra arquitectura que se desarrollara es MobileNet [4]. Cabe resaltar que en toda la investigación del estado del arte no se evidencio documentos o trabajos que usaran estas mismas topologías de redes neuronales para la identificación de aves, existen documentos como “MobileNets for Flower Classification using TensorFlow” que identifican flores, pero dado a las extensas características de un ave no se es fácil realizar una aplicación de reconocimiento de aves, lo cual nos hace uno de los pocos documentos calificados en el campo de IA.

Conociendo lo anterior, en el presente documento se prepara al lector dentro de un marco teórico en donde se presentan preámbulos y se explican conceptos básicos y referencias de los temas pronunciados anteriormente como: las aves a detectar, conceptos de redes neuronales convolucionales y especificaciones de las redes a usar. Continuo a esto, el lector conocerá los objetivos del proyecto y las soluciones de estos, complementando así, esta amplia información en los capítulos de Desarrollo, explicando la tecnología utilizada y el proceso que demuestra el cumplimiento de los entregables definidos en la propuesta de trabajo de grado, Posteriormente el capítulo de protocolos de pruebas en donde se explican los procedimientos realizados de las pruebas para obtener los resultados evidencias del cumplimiento tanto del objetivo del proyecto como de los entregables, Seguido a esto el Análisis de resultados de las implementaciones en cada etapa y finalmente, se presentan conclusiones y mejoras a futuro para la continuación del proyecto.

2. MARCO TÓRICO

2.1 JUSTIFICACIÓN AL PROBLEMA DE CLASIFICACIÓN DE AVES

Colombia es el país con mayor diversidad de avifauna en el mundo, sin embargo, en la actualidad, la diversidad de aves se encuentra seriamente amenazada; la destrucción y fragmentación de hábitats, la contaminación y la cacería han llevado a un creciente número de especies a una situación precaria. Hasta el 2016 en total 140 aves están en alguna categoría de amenaza, de estas 17 se encuentran en peligro crítico, 56 en peligro, 67 vulnerables, 28 casi en peligro, de 9 hay datos insuficientes y una especie ya extinta; Los datos fueron tomados del libro rojo de aves de Colombia [5].

En el mundo, las aves y los humanos han tenido una relación estrecha, debido a que muchas de ellas son mascotas domésticas, fuentes de alimento, sus plumas son usadas para rellenar almohadas, colchas y muchas aves son símbolos culturales, pero lastimosamente muchas especies han desaparecido o están en situación crítica de extinción a consecuencia de actividades humanas. Por estos motivos es necesario fomentar el cuidado y conservación de la riqueza de fauna y flora del país, de forma tal que se pueda mantener y recuperar la población de avifauna en Colombia; con la finalidad de proteger la fauna y flora siendo la mayor riqueza que puede poseer un país. Con esta intención se pretende avivar el amor y dar a conocer las faunas de aves que habitan en la Sabana de Bogotá

2.1.1 FAMILIA TAXONOMICA DE UN AVE

La taxonomía es una ciencia que clasifica a los organismos y establece parámetros de diferencia entre uno y otro, los grupos taxonómicos se estructuran en una jerarquía de inclusión, en la que un grupo contiene a otros menores, de esta manera crea familias, ramas y conjuntos de razas. Los taxones principales, mostrados a continuación están ordenados de más a menos inclusivos, son: dominio, reino, filo o división, clase, orden, familia, género y especie [6].

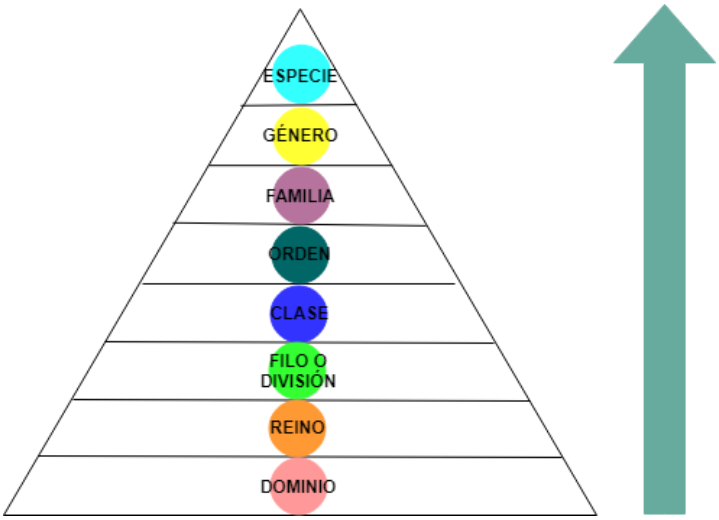


Ilustración 1. Categorías Taxonómicas.

Para entender con más facilidad el tipo de clasificación que este trabajo de grado que tiene como objetivo, es importante conocer el significado de cada grupo taxonómico mostrado en la figura anterior, sin embargo, haremos énfasis en el grupo orden y familia.

-Dominio: Esta categoría separa a los seres vivos por sus características celulares. Por esta razón, los sistemas de separación son: Eucariota y Procariota.

-Reino: Esta categoría divide a los seres vivos por su parentesco evolutivo.
-Filo o División: Es la categoría que agrupa a los animales por su mismo sistema de organización.
-Clase: La clase divide teniendo en cuenta las características semejantes entre los animales de un filo.
-Orden: Esta categoría divide la categoría anterior considerando más características comunes de algunos seres vivos dentro de una clase, en este caso las aves acaban en "-formes", como Perciformes o Cypriniformes, Columbiformes o Galliformes, en los bivalvos terminan en "-oida", como Veneroida u Ostreoida [6].
-Familia: Familia es una de las agrupaciones más importantes de los seres vivos dado que ofrece detalles exactos de la nomenclatura de diferentes familias ya que pueden agruparse en superfamilias, y los constituyentes de una familia pueden dividirse en subfamilias (y estos a su vez en infrafamilias), destacando determinados rasgos comunes y las relaciones de parentesco dentro de un orden [6].
-Genero: Esta categoría taxonómica agrupa las especies relacionadas entre sí que puede dividirse en varias especies por medio de la evolución.
-Especie: Es la categoría más conocida comúnmente. Esta categoría reúne individuos que cuentan con las mismas características genéticas permitiendo la descendencia fértil entre ellos.

2.1.2 AVES DE LA SABANA DE BOGOTA

Colombia es considerada uno de los países con mayor biodiversidad a nivel mundial. Esto se debe, en gran parte, a su privilegiada posición en la zona intertropical. En Colombia hay casi el 20% de avifauna del mundo, existen alrededor de 1800 especies de aves y de estas la Sabana de Bogotá alberga cerca de 200 [7]. Por ende, Colombia ha sido galardonada por tercer año consecutivo, el país registró el mayor número de especies de aves durante el concurso Global Big Day realizado el 4 de Mayo del 2019. Con 1590 registros superó a Perú que se quedó con el segundo lugar, los registros biológicos de Colombia señalan la existencia de más de 1.920 especies. De ellas se estima que 79 son endémicas, 197 migratorias y 193 casi endémicas [8].

A pesar de que Bogotá tiene una terrible historia de destrucción de ecosistemas, los humedales bogotanos se caracterizan por tener una alta diversidad de aves, aunque parezca increíble, a continuación, se indicaran y se describirán las 15 especies a detectar en este trabajo de grado, la información se obtuvo de la página de Wiki Aves de Colombia de la Universidad ICESI [33].

Tabla 1. Aves de la Sabana de Bogotá.

Nombre Común	**Nombre Taxonómico**	**Descripción**	**Imagen**
Copetón	Zonotrichia Capensis **Orden:** PASSERIFORMES **Familia:** EMBERIZIDAE	Mide entre 11.8 y 13.4 cm y pesa de 16.8 a 31 g. Es ligeramente crestada y con pico cónico de tamaño medio. Presenta cabeza gris, incluyendo una línea media del mismo color, dos listas negras en la coronilla, Las plumas de su cola son cafés, Su garganta es blanca y el pecho blanco grisáceo. Ambos sexos son similares.	
Azulillo norteño	Passerina Cyanea	Mide 14 cm, la macho pesa de 12.5 a 17.5 g y las hembras de 11.9	

	Orden: PASSERIFORMES **Familia:** CARDINALIDAE	a 18.5 g. Presenta mandíbula superior negruzca, mandíbula inferior blancuzca, patas negruzcas y el iris castaño. El macho en condición reproductiva presenta coronilla de color azul índigo, el cual se torna más azul en la nuca y las partes superiores. Su espalda también es azul, pero con la base de las plumas gris negruzco. La hembra completamente café por encima con café grisáceo en los hombros.	
Tórtola	Zenaida Macroura **Orden:** COLUMBIFORMES **Familia:** COLUMBIDAE	El macho mide de 22 a 28 cm y pesa de 102 a 125 g. La hembra mide de 22 a 26 cm y pesa cerca de 95 g. Tiene iris café a rojizo, piel orbital azul o gris, pico gris oscuro a negro y patas rojas. Su cola es cuneada. Sus partes superiores son café oliva con marcas negras en las alas. La hembra es similar al macho pero más opaca, con la cabeza, el cuello y partes inferiores menos rosáceas.	
Mirla Común	Turdus Fuscater **Orden:** PASSERIFORMES **Familia:** TURDIDAE	Mide alrededor de 18cm. Su pico y patas son color naranja y sus ojos blancos. Encima es gris pizarra oscuro, la cabeza negruzca con estrecho anillo ocular naranja. Por debajo es oliva grisácea pero su garganta es más pálida. El centro del pecho y abdomen son amarillo pálido (desvanecido o blanquecino). La coronilla, los lados de la cabeza y barbilla negruzcos. La garganta, el pecho y los lados gris oscuro.	

Colibri Rutilante o chillón	Colibri coruscans **Orden:** APODIFORMES **Familia:** TROCHILIDAE	Este es un colibrí de gran tamaño. El macho pesa unos 7,7 a 8,5 gramos y la hembra entre 6,7 a 7,5 gramos. Tiene una longitud de 13 cm. Su pico es robusto, mide unos 25 mm y es casi recto. Es de color principalmente verde brillante. Posee un estrecho parche violeta iridiscente desde debajo del pico hasta detrás de la región auricular. Ostenta un gran parche azul purpúreo iridiscente desde el centro del pecho hasta su abdomen. Su cola es de un matiz verde azuloso con una banda subterminal oscura conspicua.	
Azulejo Palmero	Thraupis palmarum **Orden:** PASSERIFORMES **Familia:** THRAUPIDAE	Mide entre 14 y 16 cm y pesa alrededor de 34 g. En su cabeza presenta tonos alimonados y grisáceos. Su cuerpo es principalmente oliva grisáceo, más oscuro y marcado en la espalda, con tonos azules que varían en intensidad según el ángulo de la luz. Ambos sexos son similares pero la hembra es un poco más pálida.	
Chamón	Molothrus bonariensis **Orden:** PASSERIFORMES **Familia:** ICTERIDAE	Mide alrededor de 22 cm. Tiene un pico corto, cónico y ojos oscuros en ambos sexos. Los machos son totalmente negros purpura lustroso. La hembra es café grisáceo opaco por encima, mucho más pálido debajo.	
Gavilán Maromero	Elanus leucurus **Orden:** ACCIPITRIFORMES **Familia**: ACCIPITRIDAE	Mide entre 38 y 43 cm. Tiene alas largas, agudas y cola larga de forma cuadrada. Encima presenta color gris perla, por debajo y en la cola es blanco; hombros prominentes negros; desde abajo se pueden ver pequeños parches negros en las muñecas. El inmaduro es similar, pero es estriado y teñido de café en partes superiores, pecho y cola. Es habitual que presente vuelos cernidos.	

Chulo	<u>Coragyps atratus</u> **Orden:** FALCONIFORMES **Familia:** CATHARTIDAE	Miden aproximadamente entre 56 y 66 cm, tiene un peso en la hembra de 1940 g y en el macho un peso de 1180 g. Es un ave compacta de cola corta cuadrada y de alas anchas. Cuello, patas y cabeza desnudo. presentan una coloración negro opaco, excepto por un parche blanco conformado en las bases de las plumas de vuelo. Iris y dedos marrón oscuro casi negros. Pico café negruzco con punta amarilla o blancuzca.	
Garza Ganadera	Bubulcus ibis **Orden:** PELECANIFORMES **Familia:** ARDEIDAE	Esta garza mide entre 46 y 56 cm y pesa de 340 a 390 g. Tiene un aspecto rechoncho y su cuerpo es completamente blanco con el pico amarillo y el patas verde opaco. Durante el periodo reproductivo también es blanca con la coronilla, la espalda y el pecho amarillentos y el pico y las patas rojizos. Los inmaduros presentan una coloración parecida a la del adulto, pero con las patas negras.	
Garza Real	Ardea cinerea **Orden:** Pelecaniformes **Familia:** Ardeidae	Mide de 91 a 102 cm., esbelta, de cuello y patas largos y delgados. Su plumaje es principalmente gris en las partes superiores y gris blanquecino en las inferiores, sus patas son largas, negras, iris y pico amarillos. En época de celo, los dos sexos ostentan un airón dorsal característico de la especie.	
Picaflor Enmascarado	Diglossa Cyanea **Orden:** Passeriformes **Familia:** Thraupidae	Picaflor bastante grande, casi todo azul con máscara negra y ojos rojos. Tiene el pico levantado y con un gancho en la punta, típico de un picaflor. Sexos iguales. Bastante activo como una reinita o una tangara pequeña, a menudo sigue una bandada de especies mixtas, pero a veces se alimenta por separado. Podría confundirse con Blue-and-black Tanager, pero nota la forma del pico y las alas azules (no negras). Se diferencia de otros picaflores por la	

		combinación del color azul brillante con la máscara negra.	
Picaflor Canela	Diglossa Sittoides **Orden:** Passeriformes **Familia:** Thraupidae	Picaflor pequeño. El macho es distintivo: gris oscuro arriba y beige-rojizo abajo sin marcas contrastantes en las alas. Tiene el pico levantado y con un gancho en la punta, típico de un picaflor. La hembra puede ser más confusa, es mayormente marrón. Las rayas indistintas en el pecho la separan de todos los otros picaflores. Se encuentra en áreas arbustivas bastante abiertas, arboledas y jardines; no en el bosque.	
Sirirí Común	Tyrannus melancholicus **Orden:** PASSERIFORMES **Familia:** TYRANNIDAE	Mide cerca de 22cm y pesa 40g. Tiene la cabeza gris con una máscara negruza, posee un parche naranja oculto en la coronilla el cual es más pequeño en las hembras. Su espalda es oliva grisácea, sus alas y cola son ligeramente ahorquilladas de un tono café negruzco, su garganta es gris pálido, las partes bajas inferiores son amarillas con un fuerte lavado oliva en el pecho. El pico y las patas son negros.	
Zambullidor Común	Podilymbus Podiceps **Orden:** PODICIPEDIFORMES **Familia:** PODICIPEDIDAE	Mide de 30 a 38 cm y pesa de 339 a 458 g. Presenta pico blanquecino con banda negra y ojos oscuros. En plumaje nupcial presenta garganta negra y resto del cuerpo café grisáceo, y centro del abdomen blancos.	

2.2 REDES NEURONALES ARTIFICIALES

Una red neuronal artificial (ANN) es un sistema capaz de extraer características y resolver problemas intentando imitar las funciones cerebrales por medio de un modelo de computadora.

Estas redes neuronales se componen de un gran número de elementos de procesamiento o también llamadas neuronas que reciben y combinan señales provenientes de otras neuronas a través de rutas que funcionan como entradas; en donde cada una de estas conexiones posee un valor llamado peso, Los pesos (w) representan la magnitud efectiva de la transmisión de información entre las neuronas [9].

Por consiguiente, cada neurona posee una función de activación (1) que conecta cada peso a la entrada de la neurona para luego ser sumado con cada uno de los resultados de los productos de las otras neuronas, Esta suma se somete luego a un filtro no lineal que generalmente se denomina función de

transferencia, que generalmente es una función de umbral o un sesgo en el que las señales de salida se generan solo si la salida excede el valor de umbral [10].

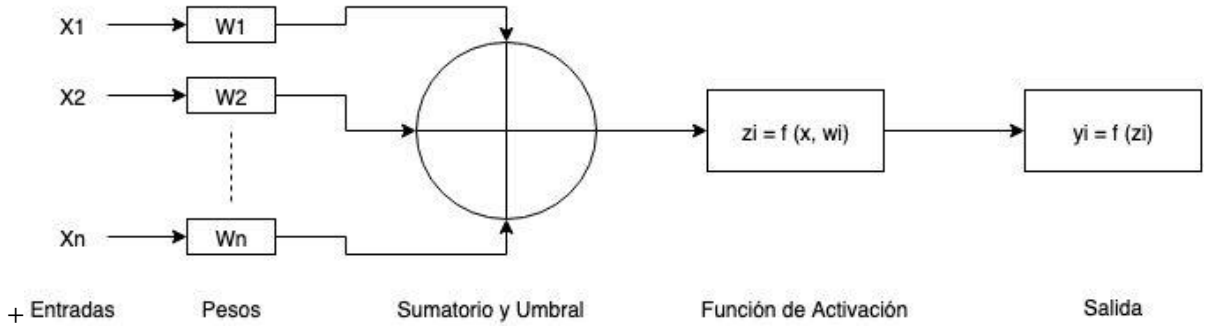


Ilustración 2. Modelo Red Neuronal de una Capa.

-Conjunto de entradas: Estas pueden ser provenientes del exterior o de otras neuronas artificiales [10].

-Enlaces de conexión: Parametrizados por los pesos sinápticos w. Es importante resaltar que el primer subíndice corresponde a la neurona receptora, mientras que el segundo subíndice corresponde a la neurona emisora. Existen dos tipos de enlaces:

Sinápticos: Relación lineal.

De activación: Relación no lineal.

-Función de activación: Es una transformación no lineal que determina la activación o el estado de la neurona, esta función se puede dividir en dos grupos, el primero basado en productos de puntos y el segundo basado en medidas de distancia [10].

$$zi = f\,(x, wi) \tag{1}$$

Las funciones de activación más comunes son:

Función Sigmoidal: $$f(u_i) = \frac{1}{1+\exp(-\frac{u_i}{\sigma^2})} \tag{2}$$

Función Gausiana: $$f(u_i) = c\,\exp(-\frac{u_i^2}{\sigma^2}) \tag{3}$$

-Función de salida: corresponde a la actividad general transmitida a la siguiente neurona en el flujo de procesamiento [10].

$$yi = f\,(zi) \tag{4}$$

En general, es posible identificar tres clases diferentes de arquitecturas neuronales:

-Redes neuronales Unicapa: Es el caso más simple de ANN, la capa de entrada se conecta directamente a la capa de neuronas de salida por medio de las sinapsis. Se le da designación de unicapa, porque solo tiene una capa con nodos ya que no se toma en cuenta la capa de nodos de entrada.

-Redes neuronales Multicapa: Se distinguen por tener una o más capas de neuronas ocultas, cuyos nodos se denomina neuronas ocultas, todos los nodos están totalmente conectados a todos los nodos de la siguiente capa.

-Redes neuronales Recurrentes: Estas redes se diferencian de las anteriores en que por lo menos tienen un lazo de retroalimentación.

Además, las redes neuronales ofrecen ventajas como:

Tabla 2. Ventajas de Redes neuronales.

No linealidad:	El procesador neuronal es básicamente no lineal y, por consecuencia, la red neuronal también.
Transformación entrada-salida:	El proceso de aprendizaje consiste básicamente en presentar a la red un ejemplo y modificar sus pesos sinápticos de acuerdo con su respuesta. Aprende, por lo tanto, una transformación entrada/salida.
Adaptabilidad:	La red tiene la capacidad de adaptar sus parámetros, aun en tiempo real.
Tolerancia a fallas:	Debido a la interconexión masiva, la falla de un procesador no alera seriamente toda la operación.
Analogía con las redes biológicas:	Esto permite la utilización mutua del conocimiento de ambas áreas.

Desde el punto de vista de aplicaciones de ingeniería, las características más importantes de las redes neuronales son: aprendizaje por interacción con el medio y mejoramiento de su desempeño por medo de este aprendizaje. La red neuronal aprende variando sus parámetros, en particular los pesos y el umbral, el aprendizaje es un proceso por el cual los parámetros se adaptan, por la interacción continua con el medio [11].

De acuerdo con lo anterior decimos que las redes neuronales a menudo son capaces de realizar acciones que los humanos o los animales hacen bien, pero que las computadoras convencionales a menudo hacen mal.

2.2.1 REDES NEURONALES CONVOLUTIVAS (CNN)

Particularmente en los últimos años, los métodos de detección que utilizan la red neuronal convolucional (CNN) han demostrado una alta precisión para la detección de objetos. Las redes neuronales convolucionales o de convolución, son un tipo de red inspiradas en la forma que simula la visión del ser humano.

Las redes neuronales convolucionales se construyen sobre múltiples capas de filtros convolucionales de una o más dimensiones, pero también tienen otros tipos de capas; La más sencilla es la capa completamente conectada a una capa de red neural normal en la que todos los resultados de la capa anterior están conectados a todos los nodos de la capa siguiente. El otro tipo de capa que se ve en las redes convolucionales es la capa de agrupación. Viene en diferentes formas, pero la que más habitualmente se utiliza es la agrupación máxima, en la que en la matriz de entrada se divide en segmentos del mismo tamaño, y se toma el valor máximo de cada segmento para rellenar el elemento correspondiente de la matriz de salida, normalmente, estas capas van hacia el final de la red [10].

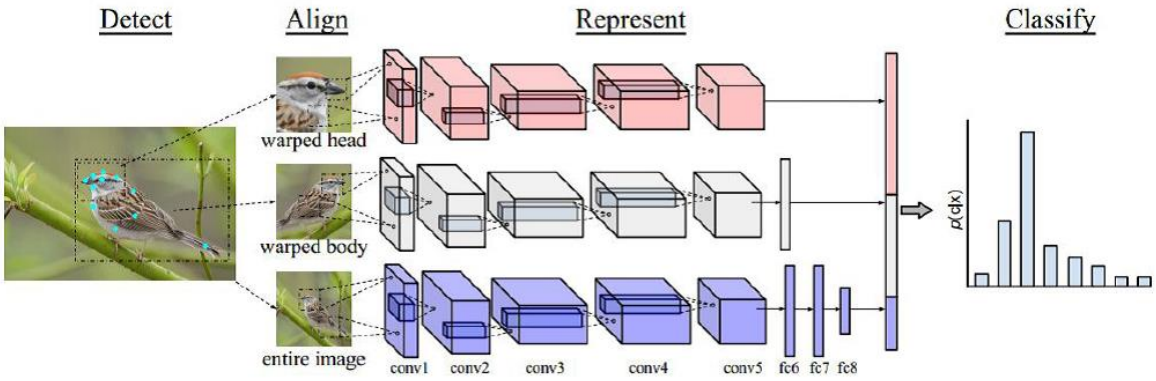


Ilustración 3. Categorización de Especies de aves usando redes neuronales convolucionales [12].

Como su propio nombre indica «red neuronal de convolución», la red emplea una operación matemática llamada convolución. La convolución es una operación matemática lineal, en la que una función se "aplica" de alguna manera a otra función. En síntesis, las redes convolucionales son simplemente las redes neuronales que utilizan convolución en lugar de la multiplicación general de matrices en una de sus capas. Las CNN explotan la correlación espacial local mediante la aplicación de un patrón de enlace local entre las neuronas de las capas adyacentes [13]. Así pues, la arquitectura garantiza que los filtros aprendidos producen la respuesta de un patrón de entrada en el espacio local. La Ilustración 4. Capa Convolucional [13]. se ve la forma en que la capa de convolución recibe los datos de entradas.

Cabe destacar que, en este tipo de red, la operación de convolución y reducción de muestreo se realizan en la primera capa del nivel, con la finalidad de detectar características locales y la capa posterior se encarga de sumar las características similares entre sí.

-Capa Convolucional: Resulta que las convoluciones son realmente buenas para detectar estructuras sencillas de una imagen, para después construir funciones aún más complejas. La convolución es la operación del cálculo del área bajo la curva del producto entre dos funciones, una de estas rotada en torno al origen como se ve en la ecuación (3), y en función de un tiempo continuo τ.

$$y(t) = x(t) * h(t) = \int_{-\infty}^{+\infty} x(\tau)h(t - \tau)d\tau \quad (5)$$

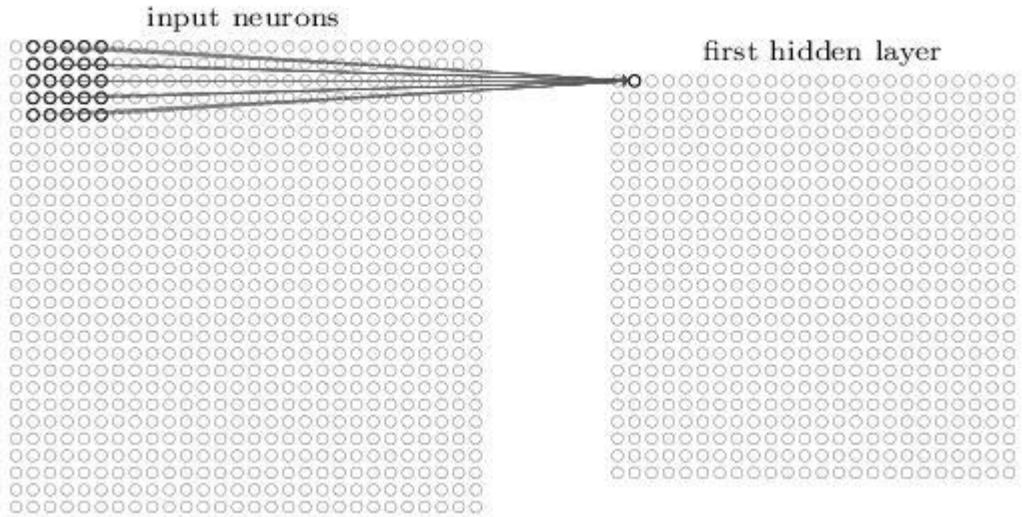


Ilustración 4. Capa Convolucional [13].

La convolución se realiza en tiempo discreto y sobre arreglos multidimensionales; para el caso de las CNN los arreglos son: segmentos de píxeles de la imagen de entrada (I (m, n)) y el kernel (K (i −m, j − n), que es el parámetro adaptado por el algoritmo de aprendizaje [13]. Lo anterior, significa que teniendo en cuenta la ecuación (3) el cálculo de la convolución entre dichos arreglos se puede realizar como una sumatoria del corrimiento o desplazamiento del uno sobre el otro. De esta manera, en (4) muestra la convolución en dos dimensiones para el procesamiento de datos en las CNN.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i - m, j - n) \quad (6)$$

m y n son las dimensiones de la imagen filas y columnas
i y j son variables que indican el corrimiento del kernel en filas y columnas sobre la misma.

Pero al final la operación de convolución en dos dimensiones termina viéndose como una correlación entre matrices debido a que no se realiza la rotación de las funciones de imagen o kernel en torno al origen; lo que permite, que en las CNN no todos los datos de la entrada se conecten a cada una de las neuronas (kernel) si no que más bien genera un proceso eficiente de generación de imágenes nuevas por conjuntos de datos, aumentando la eficiencia en el algoritmo, permitiendo la disminución de los tiempos de cómputo y logrando establecer características como detección de bordes o formas en la imagen o patrón de entrada a clasificar.

-Capa de Maxpooling: Se encarga de la reducción de muestreo que se realizan en la primera capa, con la finalidad de detectar características locales, la capa posterior se encarga de sumar las características similares entre sí. Esta capa, además, ayuda al sistema a tolerar modificaciones en la entrada generando que la salida sea aproximadamente invariante.

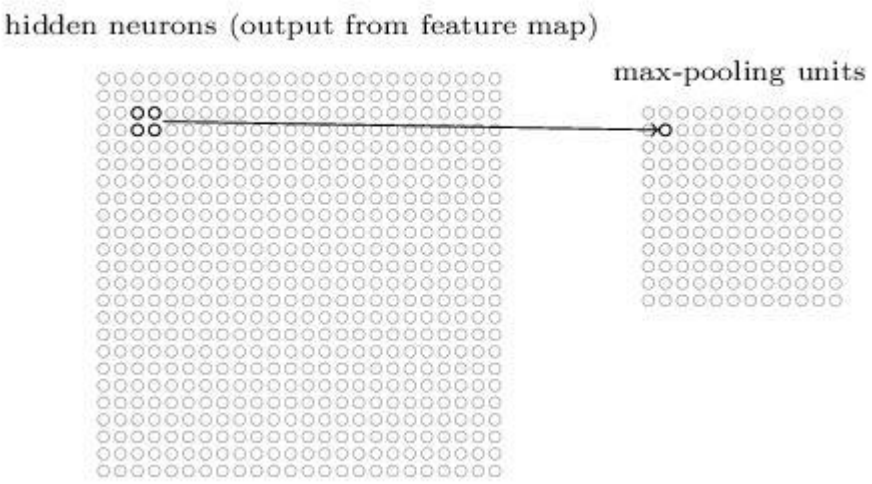


Ilustración 5. Capa de agrupación [13].

En pocas palabras, son redes neurales capaces de construir funciones completas a partir de otras menos complejas. Un ejemplo clásico es la detección facial, en la que primeras capas eligen líneas horizontales y verticales mientras que las capas posteriores encuentran narices y bocas.

2.2.2 RED NEURONAL CONVOLUTIVA ARTIFICIAL PROFUNDA

Redes neuronales convolucionales profundas llegan como una alternativa para la identificación y análisis de objetos, principalmente por el aumento del interés por el aprendizaje profundo debido a su alto rendimiento en clasificación de imágenes y detección de objetos [14]. Por lo tanto, en comparación con las

redes neuronales estándar tienen muchas menos conexiones y parámetros, por lo que son más fáciles de entrenar, mientras que su rendimiento probablemente sea solo bajo.

Las redes neuronales profundas tienen una arquitectura está compuesta por distintas capas. Las capas inferiores detectan características simples y se introducen en capas superiores, que a su vez detectan características más complejas, normalmente, cuando los datos no son linealmente separables hay tres o más capas intermedias o "ocultas" donde la salida de cada neurona es la entrada de otra [9], es decir las D-CNN cuentan con una capa de entrada de datos, donde xb recibe las características del patrón a clasificar; seguido a esto se encuentran las capas ocultas, el número de estas capas y neuronas necesarias depende del entrenamiento y de la aplicación para la cual es usada la red, En la parte final se encuentra la capa de clasificación, de la cual se obtendrán como resultado final la etiqueta de cada dato clasificado. Lo anterior se muestra en la Ilustración 6.

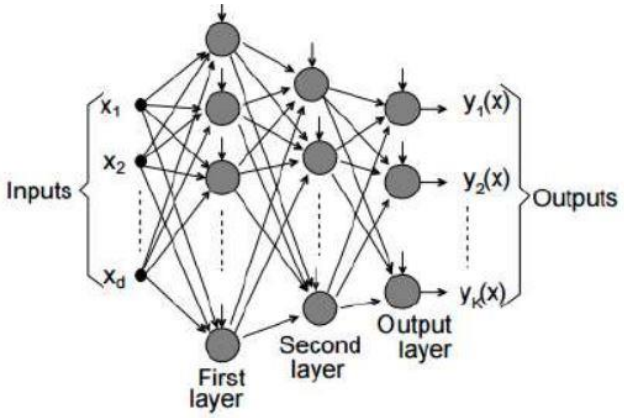


Ilustración 6. Modelo Red Neuronal Multicapa [12].

El funcionamiento de una red neuronal artificial convolutiva profunda implica de dos procesos: aprendizaje y recuperación. En la etapa de aprendizaje se adaptan los pesos de conexión dependiendo de la respuesta de los estímulos presentados la entrada. La recuperación es el proceso de aceptar una entrada y producir una respuesta determinada por el aprendizaje de la red [9].

La etapa de aprendizaje se subdivide en dos tipos; el aprendizaje supervisado, el cual se utiliza en este trabajo, es el que predice una categoría en un grupo de datos a partir de un previo entrenamiento. Los datos de entrenamiento consisten en las muestras y etiquetas que son los resultados deseados para cada salida. El objetivo de este tipo de aprendizaje es crear un algoritmo capaz de predecir el valor de una muestra de entrada después de que se le haya enseñado utilizando un grupo de muestras de entrenamiento [9], en caso de que no coincida la salida con lo esperado, se procede a modificar los pesos de las conexiones, con el fin de que la salida se aproxime a lo deseado y por lo general los algoritmos de aprendizaje supervisado se llaman clasificadores.

El aprendizaje no supervisado por otro lado cuenta con un conjunto de muestras siendo estos la entrada al sistema. El sistema no tiene información sobre las categorías de esos ejemplos. Por lo tanto, el sistema tiene que reconocer los patrones presentados en las muestras y clasificar las nuevas entradas [9].

2.3 YOLOv3

YOLO (you only look once) o en español Solo Mira Una Vez, es un sistema de detección de objetos destinado para el procesamiento en tiempo real [15].

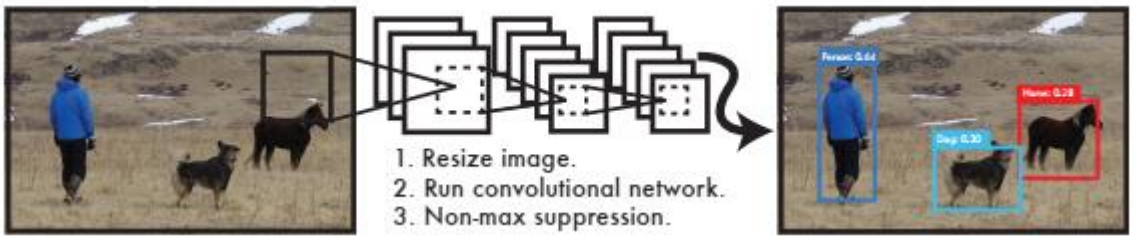
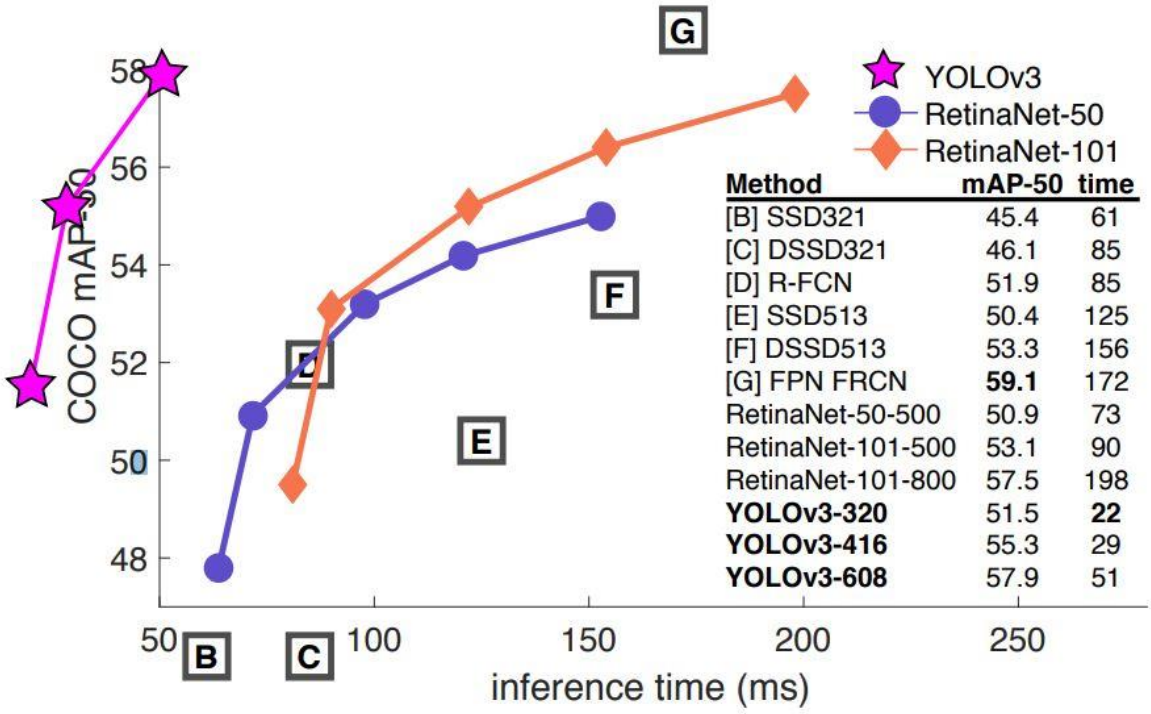


Ilustración 7. Proceso de imágenes con YOLO [15].

YOLOv3 tiene un diseño especial que permite entrenar de extremo a extremo en tiempo real a 45 cuadros por segundo sin procesamiento por lotes en una GPU Titan X. Esta red neuronal predice cuadros de límites y probabilidades de clase directamente de imágenes que solo mira una vez para predecir qué objetos están presentes y dónde están.

Esta red en comparación con otros métodos se ejecuta significativamente más rápido con un rendimiento comparable, como se ve en la Ilustración 8.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Ilustración 8. Comparación de tiempos de ejecución en términos de velocidad/precisión en el mAP en .5 métrica de IOU. Se puede decir que YOLOv3 es bueno porque es muy alto y está muy lejos a la izquierda. [16].

Para entender un poco mejor **mAP** significa media de precisión media para la detección de objetos. AP (precisión media) es una métrica popular exclusivamente para medir la precisión de detectores de objetos. La precisión promedio calcula el valor de precisión promedio para el valor de recuperación de 0 a 1 [23].

IoU (Intersección sobre unión) mide la superposición entre 2 límites. Usamos esta métrica para medir cuánto se superpone nuestro límite predicho con la verdad básica (el límite del objeto real). En algunos conjuntos de datos, predefinimos un umbral de IoU, por lo general 0.5, este umbral nos ayuda a clasificar si la predicción es un verdadero positivo o un falso positivo [23].

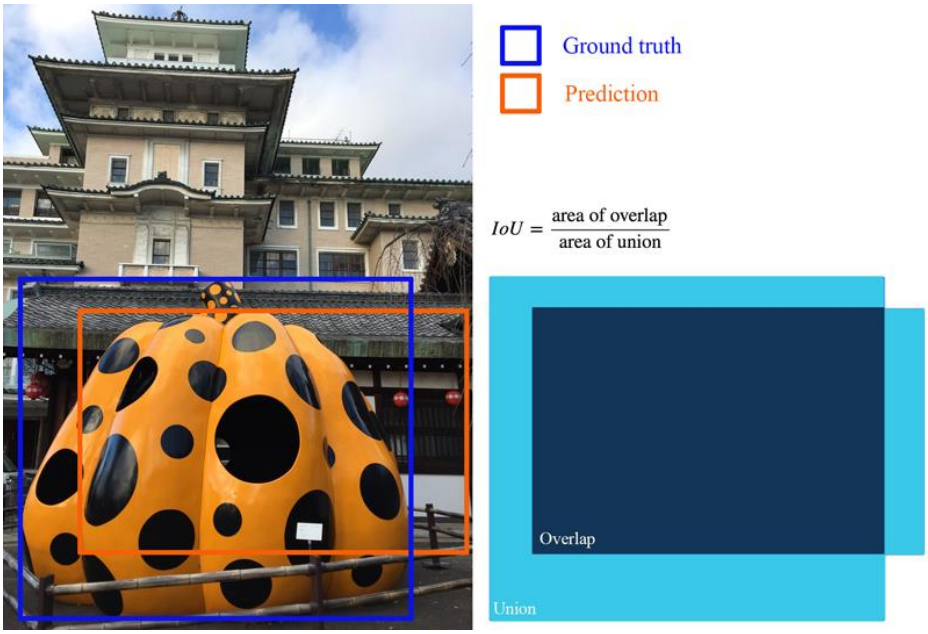


Ilustración 9. IOU [23].

El sistema de YOLOv3 predice un cuadro delimitador utilizando regresión lineal y dos capas totalmente conectadas para hacer predicciones del cuadro de límite grupos de dimensiones como cuadros de anclaje [15]. La red también predice 4 coordenadas para cada cuadro delimitador, tx, ty, tw, th, Si la celda está desplazada desde la esquina superior izquierda de la imagen por (cx, cy) y el cuadro delimitador anterior tiene ancho y alto pw, ph, entonces las predicciones corresponden Ilustración 10. Cajas de contorno con previos de dimensión y predicción de ubicación [16], [17].

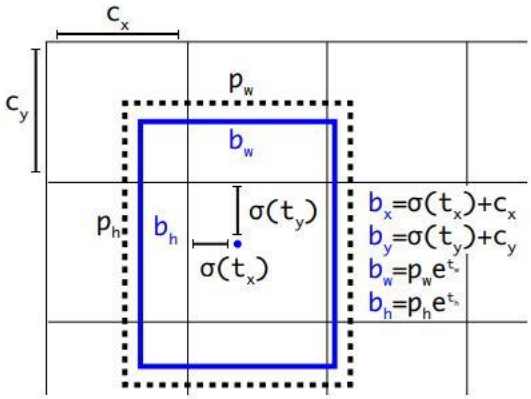


Ilustración 10. Cajas de contorno con previos de dimensión y predicción de ubicación [16], [17].

YOLOv3 tiene 24 capas convolucionales que se encuentran en la parte inicial de la red, estas capas se encargan de extraer características de la imagen mientras 2 capas que están totalmente conectadas predicen las probabilidades y coordenadas de salida. Cabe resaltar que la arquitectura de esta red está inspirada en el modelo de GoogLeNet para la clasificación de imágenes [18] solo que en lugar de utilizar los módulos de inicio de GoogLeNet, YOLOv3 simplemente utiliza capas de reducción 1 × 1 seguidas de capas convolucionales 3 × 3 [16].

El extractor de características que YOLOv3 usa se llama **Darknet-53**, Darknet es un marco de red neuronal de código abierto escrito en C y CUDA. Es rápido, fácil de instalar y admite el cálculo de CPU y GPU. Darknet se compone principalmente de 53 capas y de filtros 3 × 3 y 1 × 1 con conexiones de omisión.

Darknet-53 tiene menos BFLOP (miles de millones de operaciones de punto flotante) que ResNet-152, pero alcanza la misma precisión de clasificación 2 veces más rápido [16], es decir que Darknet-53 utiliza mejor la GPU lo que la hace es mucho más poderosa y eficiente en comparación con ResNets para evaluar ya que tienen demasiadas capas y no son muy eficientes.

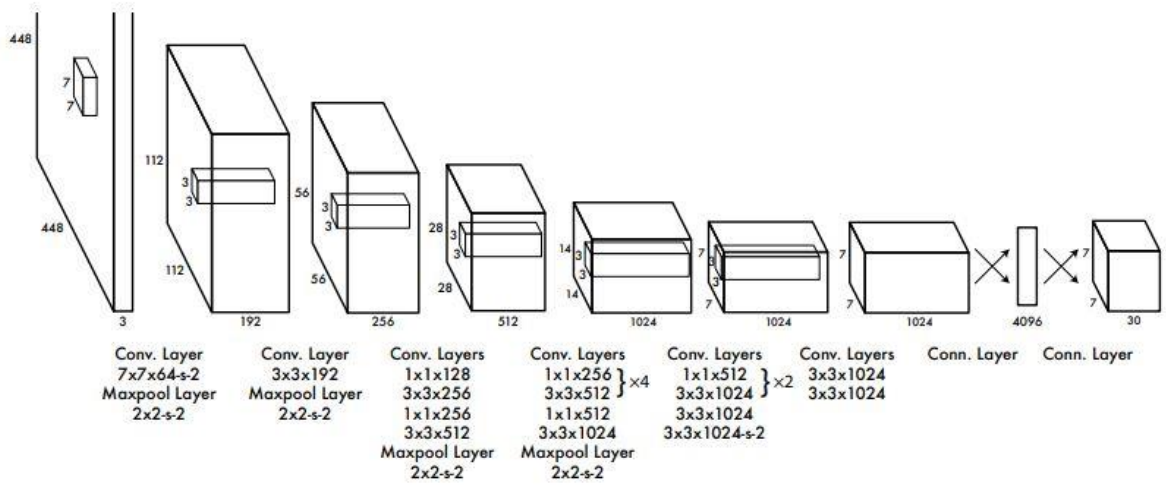


Ilustración 11. Arquitectura [15].

Nuestra capa final predice tanto las probabilidades de clase como las coordenadas del cuadro delimitador. Normalizamos el ancho y la altura del cuadro delimitador según el ancho y el alto de la imagen para que se encuentren entre 0 y 1. Parametrizamos las coordenadas x e y del cuadro delimitador para que sean desplazadas de una ubicación de celda de cuadrícula particular, de modo que también estén delimitadas entre 0 y 1. Usamos una función de activación lineal para la capa final y todas las demás capas usan la siguiente activación lineal rectificada con fugas [15].

$$\emptyset(x) = \begin{cases} x, & si\ x > 0 \\ 0.1x, & de\ otra\ forma \end{cases} \tag{7}$$

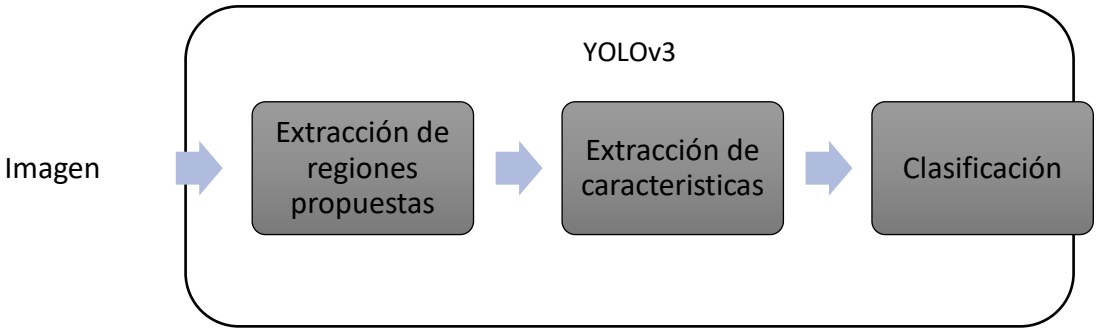


Ilustración 12. Diagrama de bloques de Yolov3.

YOLO aprende utilizando aprendizaje reforzado, ya que la forma en cómo se entrenó funciona depende ello, El aprendizaje reforzado es un tipo importante de aprendizaje automático ya que se va conociendo los resultados a medida que se van ejecutando las acciones.

2.4 MobileNet

Las redes neuronales han revolucionado muchas áreas de la artificial, permitiendo llegar con alta precisión a tareas de reconocimiento de imagen difíciles. Sin embargo, el impulso para mejorar la precisión a menudo tiene un costo, pues las redes modernas requieren altos recursos computacionales, Dado esto, MobileNet llega como una solución, puesto que es una rede neuronal de visión general diseñadas para dispositivos móviles con el fin de clasificar y detectar, para maximizar la precisión de manera efectiva y tener en cuenta los recursos restringidos para una aplicación integrada o en el dispositivo.

La capacidad de ejecutar redes profundas en dispositivos móviles personales mejora la experiencia del usuario, ofreciendo acceso en cualquier momento y desde cualquier lugar, con beneficios adicionales para la seguridad, privacidad y consumo de energía [19].

MobileNet son modelos pequeños, de baja latencia y de baja potencia, conformado por 28 capas, de las cuales se explicarán las capas más esenciales sobre las que se basa MobileNet, estas capas están constituidas por filtros separables en profundidad Ilustración 13. Esta red se basa en convoluciones separables en profundidad, las cuales son un bloque de construcción clave para muchas arquitecturas de redes neuronales eficientes, La idea es reemplazar un operador convolucional completo con una versión factorizada que divide la convolución en dos capas separadas, es decir es una forma de convolución factorizada que factoriza una convolución estándar en una convolución profunda y una convolución de 1 × 1 llamada convolución puntual [20]. Para MobileNet, la primera capa se llama convolución en profundidad, realiza un filtrado ligero mediante la aplicación de un único filtro convolucional por canal de entrada y se puede escribir como:

$$\widehat{G_{k,l,m}} = \sum_{i,j} \widehat{K_{i.j.m}} * F_{k+i-1,l+j-1,m} \quad (8)$$

Donde $\widehat{K}$ es el núcleo convolucional en profundidad de tamaño DK × DK × M

La convolución en profundidad tiene un costo computacional de:

$$DK \cdot DK \cdot M \cdot DF \cdot DF \quad (9)$$

donde DF es el ancho espacial y la altura de un mapa de entidad de entrada cuadrada. M es el número de canales de entrada (profundidad de entrada) y DK es la dimensión espacial del núcleo que se supone que es cuadrada.

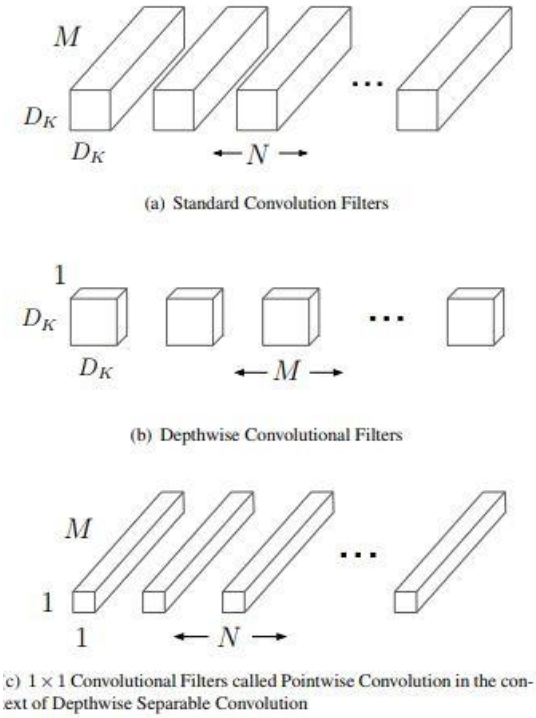


Ilustración 13. Los filtros convolucionales estándar en (a) se reemplazan por dos capas: convolución profunda en (b) y convolución puntual en (c) para construir un filtro separable en profundidad [21].

La segunda capa es una convolución de 1 × 1, llamada convolución puntual, que es responsable de crear nuevas características a través de la computación de combinaciones lineales de los canales de entrada [20]. La Ilustración 13 muestra cómo una convolución estándar 2 (a) se factoriza en una convolución profunda 2 (b) y una convolución puntual 1 × 1 2 (c).

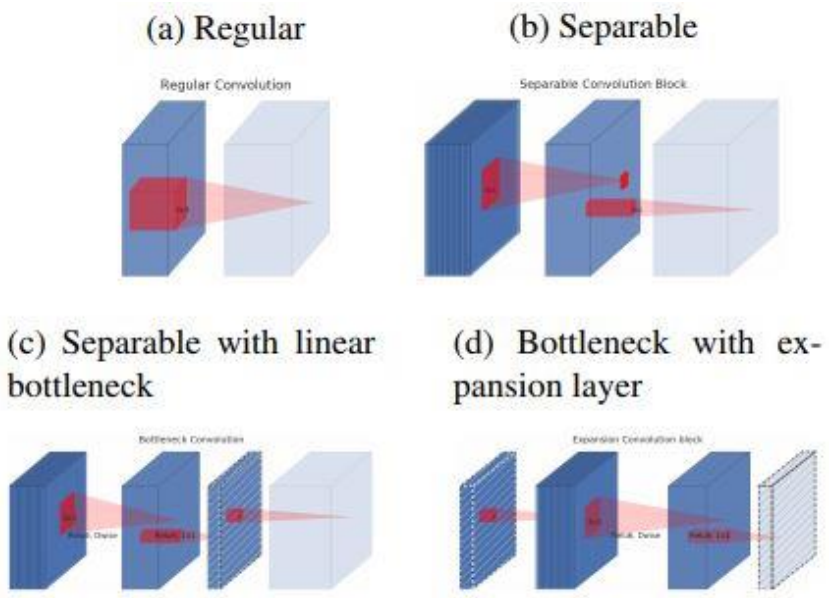


Ilustración 14. Evolución de los bloques de convolución separables [20].

En la Ilustración 14Ilustración 14. Evolución de los bloques de convolución separables *[20]*., la textura sombreada diagonalmente indica capas que no contienen no linealidades. La última capa (de color claro) indica el comienzo del siguiente bloque.

La estructura de MobileNet se basa en convoluciones separables en profundidad como se mencionó anteriormente, también introduce dos nuevas características a la arquitectura: 1) cuellos de botella lineales entre las capas, y 2) conexiones de acceso directo entre los cuellos de botella, La estructura básica se muestra a continuación:

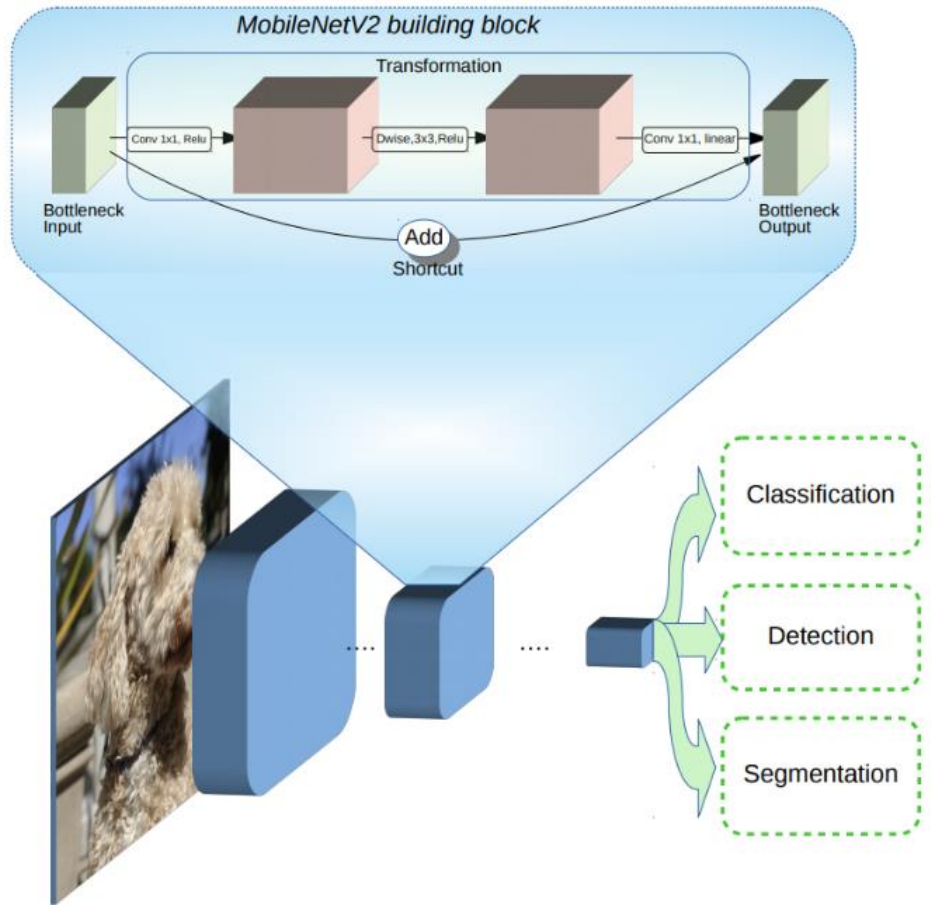


Ilustración 15 . Descripción general de la arquitectura MobileNet. Los bloques azules representan bloques de construcción convolucionales compuestos [22].

Como lo hemos dicho, la creación de modelos precisos de aprendizaje automático capaces de localizar e identificar múltiples objetos en una sola imagen sigue siendo un desafío fundamental en la visión por computadora. Los modelos de esta red de detección están entrenados en **Keras** en un marco de código abierto construido sobre **TensorFlow** que facilita la construcción, entrenamiento y despliegue de modelos de detección de objetos como lo es MobileNet.

TensorFlow es una interfaz que se utiliza para expresar algoritmos de aprendizaje automático. Una aplicación que usa TensorFlow puede ejecutarse desde dispositivos móviles como teléfonos, tabletas, dispositivos computacionales como tarjetas GPU [23]. El sistema de TensorFlow es flexible y se puede usar para expresar una amplia variedad de algoritmos de entrenamiento para modelos de redes neuronales profundas, también se ha utilizado para para implementar sistemas de aprendizaje automático incluyendo el reconocimiento de voz, la visión por computador, robótica, recuperación de información y el descubrimiento computacional de medicamentos.

Keras es una API de redes neuronales de alto nivel, escrita en Python capaz de ejecutarse sobre TensorFlow con perfecto funcionamiento en CPU o GPU. Su enfoque de desarrollo se basa en permitir una experiencia rápida para pasar de la idea al resultado con el menor retraso. Keras es recomendado para el uso de aprendizaje profundo, puesto que permite una creación de prototipos fáciles y rápidos, además admite dos tipos de redes; convolucionales y redes recurrentes [24].

En este documento, hemos mostrado el rendimiento experimental del modelo MobileNet en la plataforma TensorFlow para entrenar los conjuntos de datos, lo que puede minimizar en gran medida el tiempo y el espacio para la clasificación de las aves.

3. OBJETIVO DEL PROYECTO

3.1 OBJETIVO GENERAL

Identificar a partir de imágenes el orden y familia taxonómico de las aves de mínimo 13 especies de 12 familias diferentes que habitan en la Sabana de Bogotá.

3.2 OBJETIVOS ESPECÍFICOS

1. Recopilar un mínimo de 2000 muestras con etiquetas automáticamente para alimentar la base de datos que se usara para entrenar y evaluar el identificador de aves a partir de publicaciones en Flickr, Facebook y similares.
2. Adecuar las topologías: YOLO y MobileNet para la detección de una o más aves en una imagen.
3. Adecuar las topologías: YOLO y MobileNet para la detección de la familia taxonómica de un ave en una imagen.
4. Evaluar a partir de métricas sobre el cuadro contenedor el porcentaje de acierto en la identificación de la familia del ave.

4. DESARROLLO

En este capítulo se pretende mostrar el desarrollo de la metodología que se usó para llevar a cabo el cumplimiento de cada uno de los objetivos mencionados en el capítulo anterior. En primer lugar, se describirá el proceso de creación de la base de datos utilizada para la clasificación de las aves en la Sabana de Bogotá. Posteriormente, se describirá el desarrollo de las fases de Entrenamiento y Pruebas que se llevaron a cabo con cada red neuronal.

4.1 CREACION DE LA BASE DE DATOS

El presente trabajo de grado consiste en el entrenamiento de redes neuronales convolutivas para estimar hasta la familia taxonómica de un ave a partir de una imagen, Por lo que es necesario crear una base de datos con más de 2000 fotos de aves etiquetadas (nombre científico del ave) que se encuentran en la Sabana de Bogotá.

A partir de dos programas encontrados en la web se descargaron automáticamente 25626 imágenes de aves de Bogotá con etiquetado de clasificación taxonómica, estas imágenes fueron adquiridas de la web de grupos sociales de ornitología en la fuente Flickr.

Tabla 3. Enlaces de descarga por página.

Nombre de la web de adquisición	Enlace de descarga	Cantidad de imágenes descargadas
Flickr	https://www.flickr.com/	12626
Wikiaves de Colombia	https://www.flickr.com/groups/wikiavesdecolombia/	13000

El primer programa usado fue Jdownloader2, del cual se obtuvieron 13000 imágenes de la red social Flickr del usuario Wikiaves de Colombia, este primer grupo de imágenes fue descargado con la función de servir como apoyo o reserva de cualquier inconveniente a presentar. El segundo programa que se usó como recurso tiene el nombre de 55Photos, este programa tenía la facilidad de descargar todas las imágenes de la plataforma de Flickr del nombre taxonómico del ave que se ponga en el buscador propio del programa, de aquí se descargaron 12626 las cuales fueron separadas para ser usadas en la base de datos principal (Anexo 1. Archivos: Base de datos). En la siguiente Tabla 3 se explica de forma más clara el número exacto de aves adquiridas de 55Photos.

Tabla 4.Imágenes descargadas y utilizadas para la base de datos.

Especie	Cantidad de imágenes descargadas	Cantidad de imágenes útiles
ArdeaCinerea	1497	1019
BulbucusIbis	110	111
ColibriCoruscans	466	464
CoragypsAtratus	1498	1471
DiglossaCyanea	234	224
DiglossaSittoides	99	99
ElanusLeucurus	1599	1509
MolothrusBonariensis	599	573
PasserinaCyanea	294	288

PolydimbusPodiceps	1200	1189
ThraupisPalmarum	860	845
TurdusFuscater	303	298
TyrannusMelancholicus	1600	1244
ZenaidaMacroura	1145	1118
ZonotrichiaCapensis	1122	1084
TOTAL	12626	11536

Como se aprecia en la Tabla 3 en total se descargaron 12626 imágenes distribuidas por especie, dado que 55Photos no posee ningún tipo de filtro de veracidad en etiquetas y comentarios correspondan con la imagen publicada no se usaron completamente todas ya que no son de utilidad para el entrenamiento y evaluación de las redes neuronales, dado esto la clasificación de las imágenes útiles descargadas que conforman la base de datos de entrenamiento se realizó de forma manual, esto quiere decir que nosotros determinamos si la imagen es funcional, observando si pertenece a la especie de ave que corresponde a la descarga y también si el tamaño de RoI (Region of Interest) cumple con la mínima establecida.

Cabe resaltar que no se tuvo en cuenta para formar la base de datos si la imagen era icónica, es decir que en la imagen solo se observe al ave en específico para poder extraer con mucha facilidad las características de forma, color y textura. por lo tanto, se pueden encontrar fotografías con el ave como foco principal o imágenes en las que presencian muchas aves, animales, árboles y paisajes.

La creación de esta base de datos tomo un tiempo de 3 meses y 20 días, desde la búsqueda de fuentes de descarga, revisión manual de cada imagen para determinar su utilidad y corrección de etiquetado. De estas imágenes de la base de datos se usarán el 80% para el entrenamiento de 2 redes neuronales convolutivas y el restante 20% pondrá a prueba el funcionamiento y detección del orden y familia del ave en la CNN.

4.2 ADECUACION DE LA TOPOLOGIA YOLOv3

En este apartado se explicará más a fondo la tecnología utilizada y el proceso que demuestra el cumplimiento de los objetivos específicos 2 y 3 definidos en el capítulo 3.

4.2.1 FASE DE ENTRENAMIENTO

En lo que concierne a la implementación de algoritmos de aprendizaje, es casi un estándar usar unidades gráficas de procesamiento GPU, ya que cuando se entrena una red neuronal es necesario realizar complejos y largas operaciones matemáticas con matrices, siendo las GPU un aliado ideal dado al alto rendimiento y potencia de cálculo que estas poseen gracias a su procesamiento paralelo, como se indica en [25].

Por otra parte, la autora del documento [26] afirma que las CPU son capaces de entrenar redes neuronales profundas, pero les toma una gran cantidad de tiempo debido a que esta requiere una gran cantidad de ciclos de reloj por su baja cantidad de núcleos y forma de trabajo secuencial.

Para el presente trabajo se utilizó el servidor de computo "CRATOS" propiedad de la Pontificia Universidad Javeriana, se decidió usar esta máquina debió a que posee un hardware muy robusto, principalmente en las 8 unidades de procesamiento gráficas de 12GB en RAM. Por lo tanto, se tiene instalado la versión 8.0 de CUDA, como su página oficial [27] indica, es la arquitectura de cálculo paralelo desarrollada para NVIDIA para aumentar el rendimiento de los núcleos y procesar datos en forma paralela y dado a que se van a trabajar con redes neuronales, tiene que encontrarse instalada la biblioteca cuDNN (versión instalada 5.1), según su página oficial [28] esta es una librería escrita en CUDA para redes neuronales profundas que aceleran la GPU optimizando rutinas estándar como capas de convolución, agrupación, normalización y activación hacia adelante y atrás de las redes neuronales.

Para el entrenamiento en YOLOV3 hay que primero preparar el conjunto de datos, para esto es necesario que cada una de las imágenes posea un archivo con formato ".txt" con el mismo nombre del archivo de la imagen, en la siguiente Tabla 5 se puede observar el formato interno del archivo.

Tabla 5. Información dentro del archivo de texto para cada imagen.

	Clase ave	X normalizada	Y normalizada	W normalizado	H normalizado
Ave N° 1	6	0.555	0.76878	0.37	0.21621
Ave N°2	6	0.574	0.2897	0.396	0.4294
Ave N°3	6	0.382	0.3753	0.392	0.37837

Cada línea del archivo corresponde a cada objeto encontrado en la imagen, la primera columna <clase-objeto> es la clase a la que pertenece esa especie. Las siguientes columnas corresponden al rectángulo contenedor del pájaro, <x> <y> estas son las coordenadas centrales del rectángulo, <ancho> <alto> es el tamaño que está ocupando el pájaro en la imagen. Todos los valores excepto la clase están entre 0 y 1, ya que se normalizan con el tamaño de toda la imagen.

Tabla 6. Tabla de identificación de la clase de cada ave, es un numero de 0 a 14 para especificar cada especie.

Representación por clase de Ave	Nombre clase Ave
0	ArdeaCinerea
1	BulbucusIbis
2	ColibriCoruscans
3	CoragypsAtratus
4	DiglossaCyanea
5	DiglossaSittoides
6	ElanusLeucurus
7	MolothrusBonariensis
8	PasserinaCyannea
9	PodilymbusPodiceps

10	ThaupisPalmarum
11	TurdusFuscater
12	TyrannusMelancholicus
13	ZenaidaMacroura
14	ZonotrichiaCapensis

Para generar cada uno de los archivos y coordenadas de los rectángulos de todas las imágenes utilizamos un Script con interfaz gráfica llamado “bbox label tool multi class” diseñada por el ingeniero Francisco Calderón. Manualmente se dibujó un rectángulo a cada pájaro que aparece en la imagen siempre y cuando este corresponda a alguna de las 15 clases, en los demás casos el archivo de texto queda vacío. Entendiendo que cuando YOLO analice este tipo de archivos concluye que no hay ninguna clase y después del entrenamiento al presentarle imágenes similares su vector de salida con las probabilidades de que los objetos pertenezcan a alguna de las clases serán muy bajas.

Tabla 7. Archivo para el entrenamiento.

aves.data	
clases: clases a identificar	15
train: archivo de entrenamiento	Train.txt
valid: archivo de prueba	Test.tx
names: nombres de las clases	aves.names
backup: archivo donde se guardarán los pesos	backup

En la Tabla 7 se coloca la ubicación de algunos archivos que son necesarios para el entrenamiento como se entiende a continuación, la primera fila es el número de clases, la siguientes líneas son la ubicación de los archivos que contiene los path de las imágenes de entrenamiento y validación, la siguiente línea es la ruta del archivo que contiene el nombre de las clases y la última fila es la ruta de la carpeta donde se guardaran los pesos y puntos de recuperación a medida que se entrena. Cabe recalcar que las rutas mostradas son rutas relativas de linux.

El archivo yolov3.cfg contiene toda la arquitectura de las capas de la red neuronal de YOLO, por tal razón se realiza una copia del archivo con el nombre de yolov3_aves.cfg. Y es necesario modificar las lineas del comienzo del archivo definiendo el “batch” igual a 64, siendo esta la cantidad de imágenes que se cargan para cada paso de entrenamiento, permitiendo calcular un gradiente y actualizar el valor de los pesos. Muy de la mano al valor anterior se define el valor de “subdivisions” en 32, este parámetro busca disminuir los parámetros de la VRAM de la GPU, dividiendo las 64 imágenes en 32 grupos. Se probaron distintos valores más pequeños para hacer que el entrenamiento fuese más rápido, pero con estos valores se saturaron las memorias de las GPU.

En el mismo archivo es necesario adecuar algunos parámetros de las entradas y salidas de tres de las capas, siendo estas las modificaciones que se hacen a la arquitectura de la red, y se colocaron los siguientes valores filters=60 y classes=15. El valor de “filters” indica la cantidad de núcleos convolucionales que hay en una capa y se calculó con la ecuación “filters = (clases + 5) * 3”, donde el 5 equivale al número de coords y el 3 a número de máscaras donde se encuentran los valores de anclaje. El valor de los coords, máscaras, la ecuación del filtro y las 3 capas que se modificaron son las recomendaciones realizadas por el autor [29]. El valor de “classes” permite calcular la probabilidad para cada una de las clases se encuentre en el rectángulo contenedor.

El último paso que seguir antes de entrenar, es dividir la base de datos al azar en los conjuntos de entrenamiento y de validación. Estos 2 conjuntos son importantes ya que permiten realizar los cálculos del valor promedio de acierto para cada una de las clases siendo esta la métrica mAP e IoU, como se explicó en el capítulo 2 sección 2.3. Sabiendo el valor del mAP después de cada iteración, permite escoger el mejor archivo de pesos sinápticos y calcular el punto conveniente de detención del entrenamiento evitando el sobre-entrenamiento; esto sucede cuando la red neuronal ya no es capaz de detectar objetos en nuevas imágenes que se le presenten.

Se realizó un script (Anexo 2. Script1) que dividió el conjunto de entrenamiento en 80% para entrenamiento y 20% para validación, y generar 2 archivos "train.txt" y "test.txt" que contienen los path a cada una de las imágenes. Se escogen estos valores para dejar al menos 80 imágenes en la clase DiglossaSittoides para el entrenamiento, y dejar imágenes suficientes para el cálculo del mAP.

Para empezar el entrenamiento es necesario descargar y utilizar los pesos convolucionales que están pre-entrenados en Imagenet ("darknet53conv.74"). Se parte el entrenamiento desde este punto debido a que 12000 imágenes es muy poco para partir desde cero y además es lo recomendado por el autor [29]. Con lo anterior se inicia el entrenamiento ejecutando desde el terminal con la siguiente línea:

"./darknet detector train ArchivosYolo/aves.data cfg/yolov3_aves.cfg darknet53conv.74 -gpus 3,5,6,7 -map | tee resultadosentrenamiento.txt".

Todo lo que se muestra en el terminal se guardará en el archivo de texto "resultadosentrenamiento.txt" para graficar los resultados.

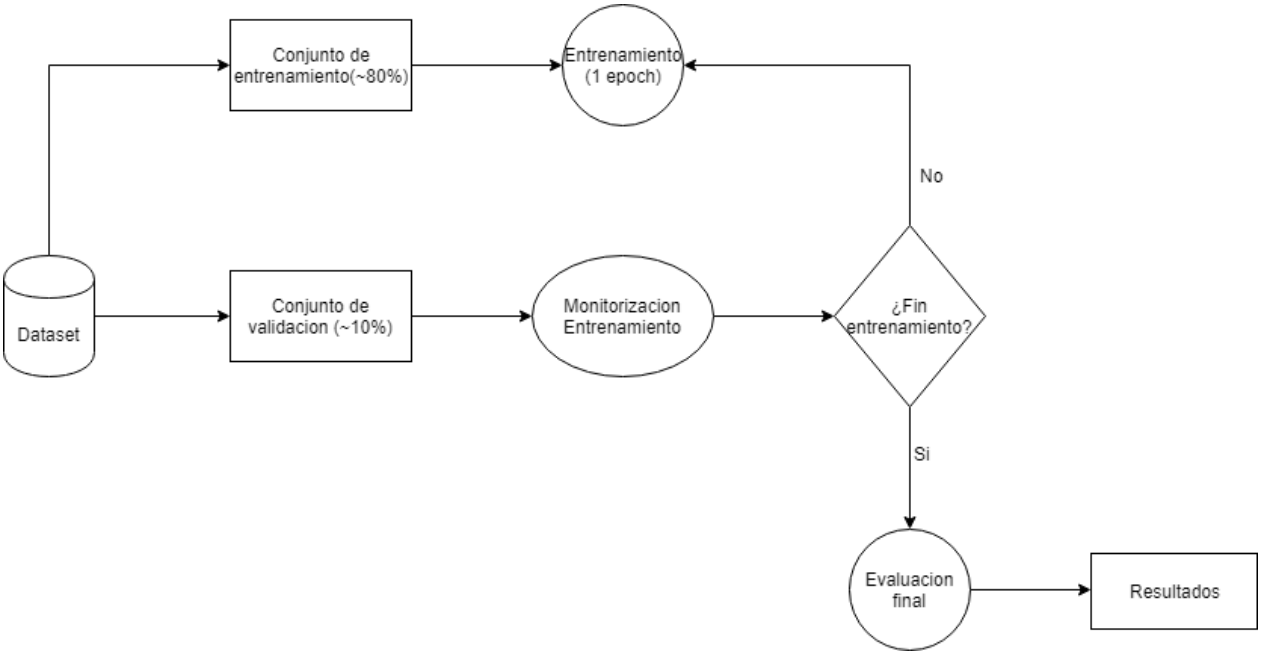


Ilustración 16. Funcionalidad de los conjuntos de entrenamiento y Validación en YOLO.

4.3 ADECUACION DE LA TOPOLOGIA MobileNet

En este apartado se explicará más a fondo la tecnología utilizada y el proceso que demuestra el cumplimiento de los objetivos específicos 2 y 3 definidos en el capítulo 3.

4.3.1 FASE DE ENTRENAMIENTO

Como se recalcó antes en YOLO entrenar desde cero una red neuronal para el reconocimiento de imágenes requiere una mayor cantidad de datos etiquetados y mucha potencia de cómputo (cientos de horas de GPU). Por lo tanto, para esta aplicación es suficiente usar la técnica de aprendizaje por transferencia, que logra entrenar modelos con un conjunto más limitado datos y acelerar el entrenamiento. Señalando que los resultados después de usar el aprendizaje por transferencia nunca van a ser tan buenos como entrenar el modelo completo.

En TensorFlow es necesario traer la biblioteca "TensorFLow Hub" que como su página oficial lo indica [30], es una biblioteca para publicación, descubrimiento y uso de modelos reutilizables de aprendizaje automáticos. Para esta aplicación se comenzó usando el módulo de extracción en vectores las características de la imagen con la arquitectura Inception V3 entrenada en Imagenet [31].

Los autores de TensorFlow desarrollaron el código "retrain.py" [21] el cual carga el módulo pre-entrenado y pasa a entrenar la capa superior de la red para identificar nuestras clases de aves, por lo que implementa funciones que agregan las operaciones correctas al gráfico de la red. Ninguna de las especies de aves se encontraba en las clases de Imagenet en las que se formó la red completa, por lo tanto, el aprendizaje por transferencia reutiliza las capas inferiores que han sido entrenadas para detectar ciertos objetos.

La biblioteca de Keras le aporta un nivel de abstracción alto a TensorFlow, que se utiliza para crear y entrenar los modelos, haciendo que TensorFlow sea una herramienta más fácil de usar sin sacrificar flexibilidad y rendimiento. Mas adelante se explica cómo se ingresa la base de datos de imágenes y el porcentaje de estas que se usan para entrenamiento y validación en el código "retrain.py", pero es necesario resaltar que este código detecta la cantidad de clases y con esto modifica la última capa de la red, creando la misma cantidad de neuronas como clases, aplicando en ellas la función "softmax" para que las neuronas de salida tomen valores entre 0 y 1, esta información fue adquirida trabajando con la documentación de Keras [24].

YOLO se diferencia de la biblioteca de TensorFlow en la forma en la que realiza el análisis de la imagen ya sea para entrenamiento o identificación de objetos, dado que YOLO hace todo un recorrido por la imagen y calcula las probabilidades de los rectángulos no solo buscando características para cada clase, si no también encuentra la ubicación del objeto en la imagen, dando la ventaja de encontrar múltiples clases en una misma imagen. TensorFlow, por lo contrario, no funciona de esta manera, el solo ajusta sus pesos para encontrar características en cada una de las clases y luego detectarlas, esto quiere decir que la imagen que se le presenta si por ejemplo tiene 3 o 4 aves de la misma o diferente clase es necesario hacer el recorte de la imagen para cada una, siendo esta la entrada a la red para detectar a qué clase pertenece cada una. Por lo tanto, fue necesario adecuar el conjunto de entrenamiento que presentamos para YOLO.

Para YOLO fue necesario crear un archivo de texto que tiene las coordenadas de cada uno de los objetos presentes en cada imagen como se evidencia en la tabla 5, por lo tanto, se reutilizo en el desarrollo de un script (Anexo 3. Script 2) que tiene la función de abrir la imagen y su archivo de texto correspondiente, tomar cada una de las coordenadas de los objetos encontrados y hacer el recorte de la imagen, quiere decir que se creara un archivo de imagen para cada ave encontrada.

Se creo una carpeta llamada "Imágenes" y esta contiene 15 carpetas con el nombre de cada una de las clases, y en cada una de ellas se almacenan los recortes realizados para esa clase, En la siguiente Ilustración 17 se encuentra el ejemplo de la distribución de las carpetas.

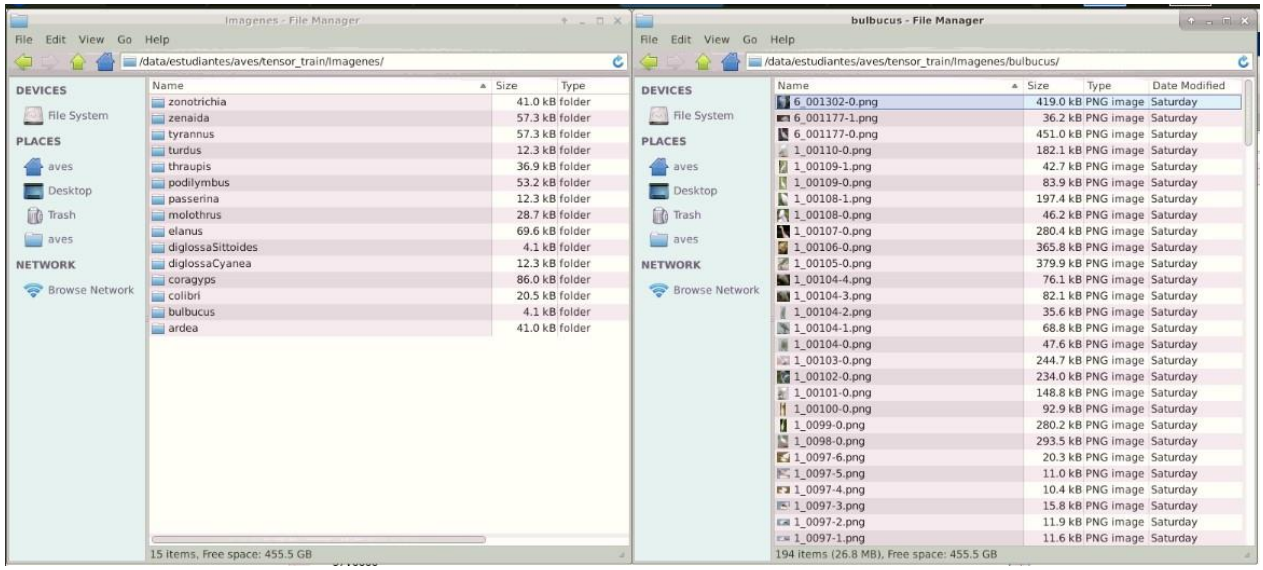


Ilustración 17. Distribución de carpetas.

La primera fase del entrenamiento analiza todas las imágenes y calcula y almacena los valores de cuello de botella (vector de características de imagen). Esta penúltima capa ha sido entrenada para generar un conjunto de valores para que el clasificador distinga entre todas las clases, siendo este un resumen de las imágenes que le permite al clasificador realizar una buena elección en un conjunto muy pequeño de valores. Cada imagen se reutiliza varias veces durante el entrenamiento y el cálculo del cuello de botella toma un tiempo considerable, es importante que si se vuelve a ejecutar el script el reutilizara estos valores.

Después de la primera fase, continua la capacitación de la capa superior de la red. Para empezar, se establecieron 10000 pasos de entrenamiento, cada paso elije 10 imágenes del conjunto de entrenamiento, encuentra su cuello de botella almacenados y las introduce en la capa final para obtener predicciones. Esas predicciones se comparan con las etiquetas reales para actualizar los pesos de la capa final usando el proceso de propagación hacia atrás.

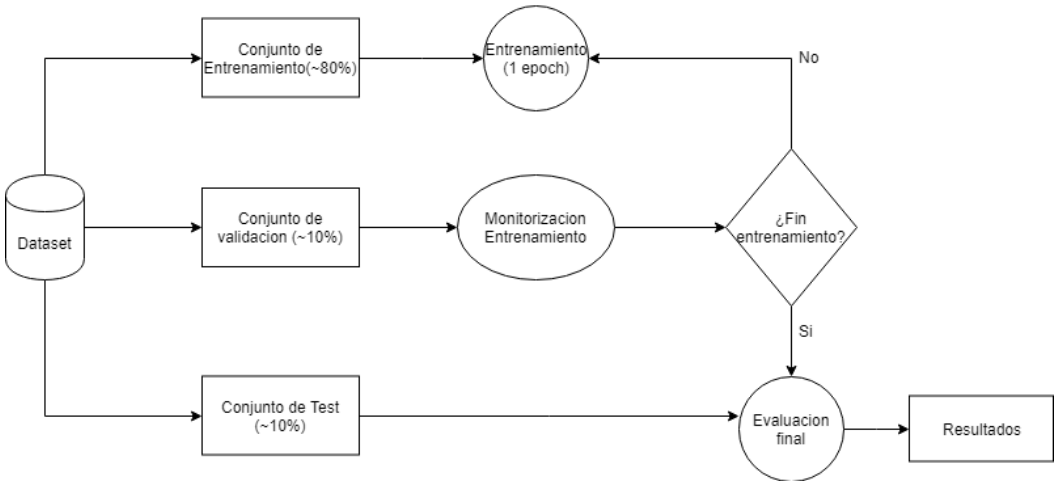


Ilustración 18. Funcionalidad de los Conjuntos Entrenamiento, Validación y testing de MobileNet.

5. PROTOCOLO DE PRUEBAS

5.1 FASE DE ENTRENAMIENTO Y PRUEBAS YOLOv3

Durante el entrenamiento de YOLO utilizamos los pasos recomendados por el autor, que se basan en las mismas métricas basadas en el índice Jaccard o intersección sobre la unión (IoU), [32]. Todo el proceso de entrenamiento consiste en 30000 iteraciones de un lote de proceso de 64 imágenes y 32 subdivisiones. Como una métrica de entrenamiento usamos la pérdida total del lote actual y una perdida promedio de ejecución para los lotes recientes, calculada así:

$$Promedio_{perdida} = Promedio_{perdida} * 0.9 + perdidas * 0.1 \quad (10)$$

El mAP nos permite evaluar el rendimiento de la red neuronal, primero se calcula la precisión media (AP), para cada una de las clases y luego el valor de media de precisión media (mAP). Desde las 30000 iteraciones del algoritmo, guardamos el estado actual de la red cada 1000 iteraciones y ejecutamos el mAP promedio para todas las imágenes y clases como un indicador de rendimiento.

El mAP se calculará para cada 4 epocas,

Esta métrica se graficó en tiempo real mientras la red estaba entrenando debido a que en la línea de código que se mostró en el item 4.2.1, con la que se inició el entrenamiento mostrada en y se puede visualizar en la Grafica de la Ilustración 20.

5.2 FASE DE ENTRENAMIENTO Y PRUEBAS MobileNet

En el terminal se mostrará para cada uno de los pasos la precisión de validación, precisión de entrenamiento y la entropía cruzada. La precisión de entrenamiento muestra que porcentaje de las imágenes utilizadas en el lote de entrenamiento actual se etiquetaron en la clase correcta. Estos valores también son almacenados en la carpeta temporal del disco duro, y usando la biblioteca llamada TensorBoard (viene incluida en TensorFlow) permite visualizar las estadísticas del entrenamiento gráficamente y en tiempo real. La precisión de validación es la precisión de un grupo de imágenes seleccionadas al azar de un conjunto diferente. La diferencia de la precisión de validación y entrenamiento radica en que las imágenes que la red ha usado para aprender pueden adaptarse al ruido de los datos del entrenamiento, entonces si por ejemplo la precisión de entrenamiento es alta pero la de validación es baja entonces significa que la red esta sobre ajustando y memorizando características particulares en las imágenes de entrenamiento, lo cual no es nada útil. La entropía cruzada es una función de perdida que permite determinar que tan bien está progresando el proceso de aprendizaje.

El script "retrain.py" divide la base de datos de imágenes en tres conjuntos diferentes, el de entrenamiento, validación y pruebas, tomando los porcentajes para cada división respectivamente 80%, 10% y 10% del conjunto total.

Se escogió "accuracy" como método para evaluar el modelo de la red neuronal, haciendo referencia al porcentaje de observaciones correctas clasificadas.

$$accuracy(\%) = \frac{inst\_correctas}{N\_eval} * 100 \quad (11)$$

Con N_eval el número de observaciones sobre el que estamos evaluando esta métrica, e inst_correctas el número de observaciones para las que la clase predicha y la clase con la que se etiqueta la observación son iguales.

Estos porcentajes se calculan y se grafican para el conjunto de validación y el de entrenamiento. Al comparar sus resultados se determina en qué momento la red empieza a

sobrentrenarse, por lo tanto, se aprende características específicas de las imágenes y cuando se le presenta una nueva no es capaz de determinar la clase.

Como se dijo anteriormente es necesario una función de coste que permita conocer con un valor real, que tan precisas son las predicciones de una red neuronal para un conjunto de observaciones dadas. La función de coste se denomina "cross entropy" relaciona una matriz X con N observaciones, T la matriz de etiquetas de X y W un conjunto con los pesos y sesgos "b".

6. ANALISIS DE RESULTADOS

6.1 YOLO

El correcto entrenamiento de los clasificadores es sin duda alguna la etapa base para el funcionamiento óptimo de un sistema de detección de objetos de. De esta manera, obtener de las curvas de aprendizaje, el mAP y el promedio de perdidas en la arquitectura de Yolo, además del porcentaje de acierto y entropia cruzada en la arquitectura de TensorFlow, permitió a través de su análisis verificar y evaluar aspectos fundamentales de cada modelo entrenado.

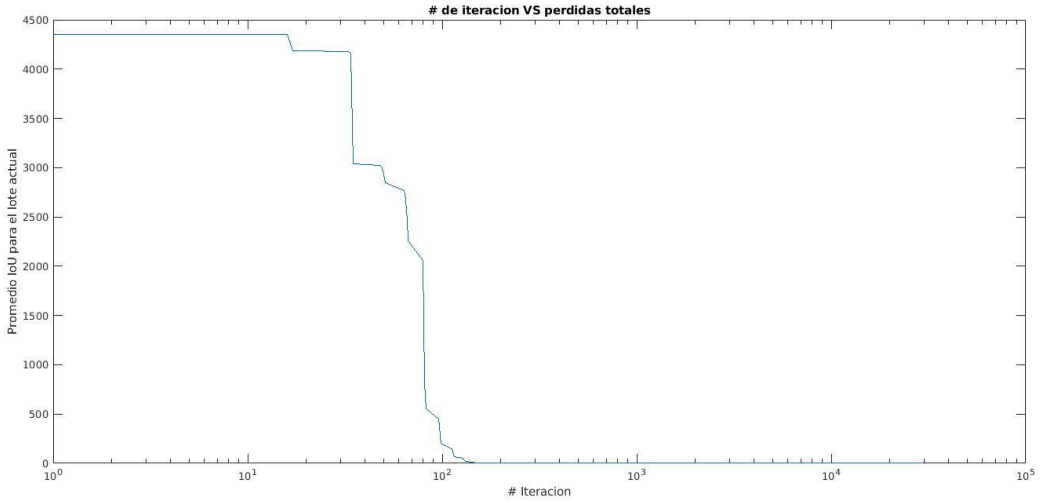


Ilustración 19. Iteración pérdidas totales.

La ilustración 19 indica la velocidad de entrenamiento de la arquitectura de la red, a través de mostrar el error en el cálculo de IoU (Intersección sobre la unión) sobre las imágenes que se usaron como entrenamiento en la red para cada una de las iteraciones. Al final de cada iteración la red tomaba un porcentaje de las imágenes que se usaron para entrenamiento en ese punto y predice el rectángulo contenedor del objeto en la imagen y lo compara con el rectángulo real establecido en la sección 4.2.1, la idea de calcular este valor es superar en todas las imágenes el umbral del 50% de área compartida entre los rectángulos.

La grafica muestra como el error en el cálculo de IoU para cada iteración comienza en un valor muy alto y a medida que avanzan las iteraciones este valor tiende a 0, principalmente cuando casi llega a la iteración número 100 se puede decir que la red ya entrenó.

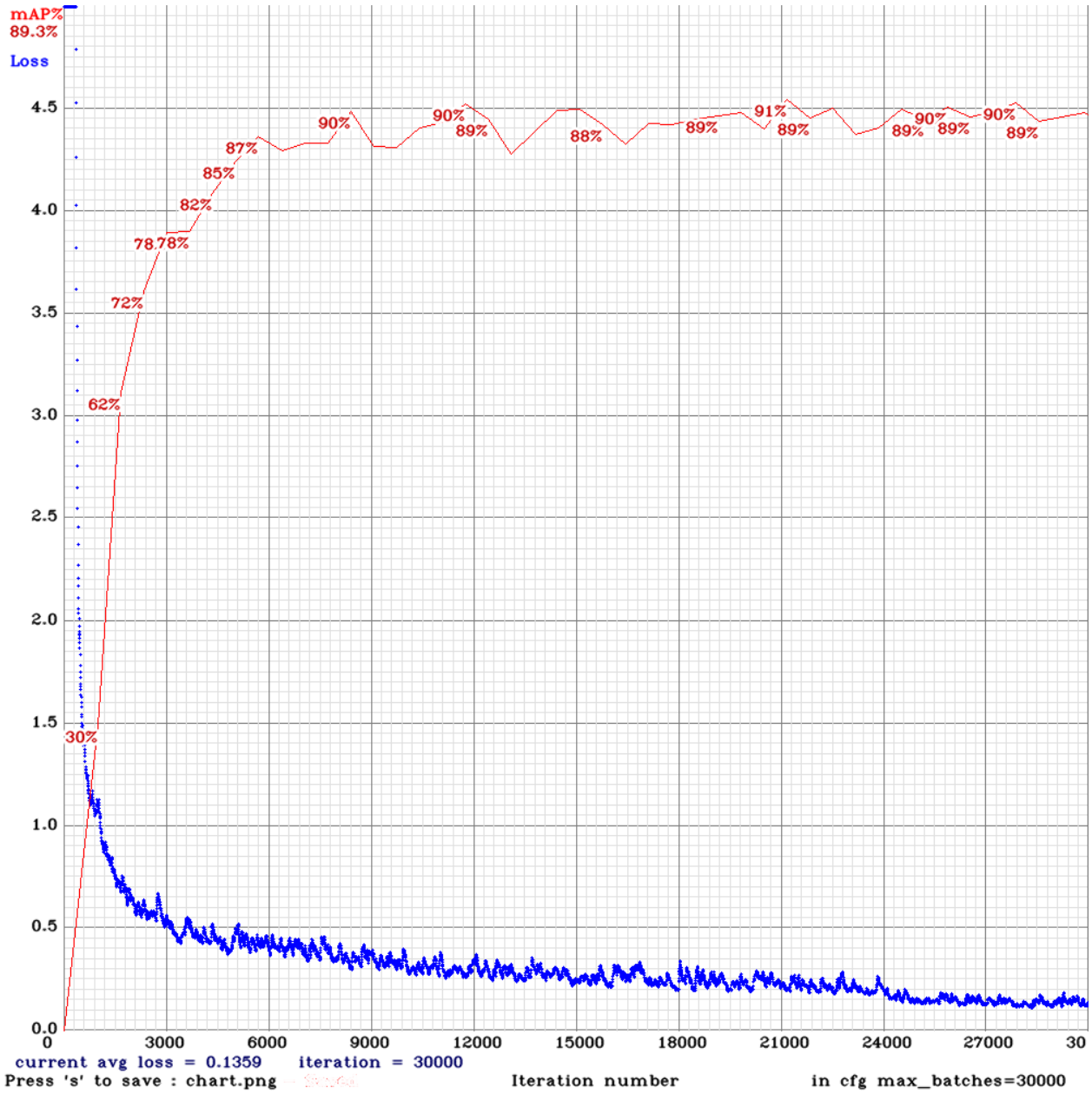


Ilustración 20. Maptrain.

Los valores del mAP y el error promedio para cada iteración se graficaron en tiempo real, a medida que la red se entrenaba se visualizaban estos valores permitiendo establecer en qué momento detener el entrenamiento dado que la red se encuentra en sobre entrenamiento. Al analizar la gráfica permite dar una idea de cuál es el mejor archivo de pesos para usarlo como definitivo para una implementación en una aplicación.

El cálculo de los valores de mAP para cada iteración es algo complejo, dado que para obtener los valores más altos es necesario que la red realice la predicción correcta de la clase para cada uno de los objetos de las imágenes y al mismo tiempo poseer un porcentaje alto de IoU, su punto de referencia son los archivos y adecuaciones realizadas en la sección 4.2.1.

El principal uso de la ilustración 20 es tener una idea de cuál es el mejor archivo pesos convolucionales entrenados por la red neuronal, este archivo se guarda cada 1000 iteraciones de la red y el cálculo del mAP es la mejor forma de evaluar la predicción para cada uno de los 30 archivos que se generaron.

Los puntos claves en el análisis de la gráfica de la ilustración 20 es encontrar la iteración en la cual el valor de mAP deja de crecer y empieza a volverse constante, luego se busca la iteración donde se consigue el valor más alto y donde deja sus fluctuaciones no sean mayores al 2%, ocurriendo en la iteración 21000, también es necesario observar la curva del porcentaje de error, dado que si este valor es muy alto el entrenamiento no sirve. Y cabe resaltar que es necesario escoger el archivo de pesos cuando el valor del mAP comience a volverse constante, no decrece y sea el más alto, dado que si se escoge un archivo de pesos de una iteración mayor la red esta sobrentrenándose y esto afecta principalmente en que solo sería capaz de detectar objetos en las mismas imágenes de entrenamiento y no en las nuevas imágenes que se le presenten.

El porcentaje de error se calculó con todas las imágenes en las que no pudo detectar correctamente el objeto ya sea porque no supero el umbral del 50% en IoU o no predijo la clase correcta para el objeto.

El 91% de porcentaje de mAP alcanzado y un porcentaje de error menor al 0.5% indica que fue un entrenamiento exitoso, y que el archivo de pesos convolucionales está listo para usarse en la implementación de una aplicación. Usar este archivo de pesos traza con bastante precisión los rectángulos contenedores de los objetos y al mismo tiempo predice con altos porcentajes la clase a la que pertenece dicho objeto.

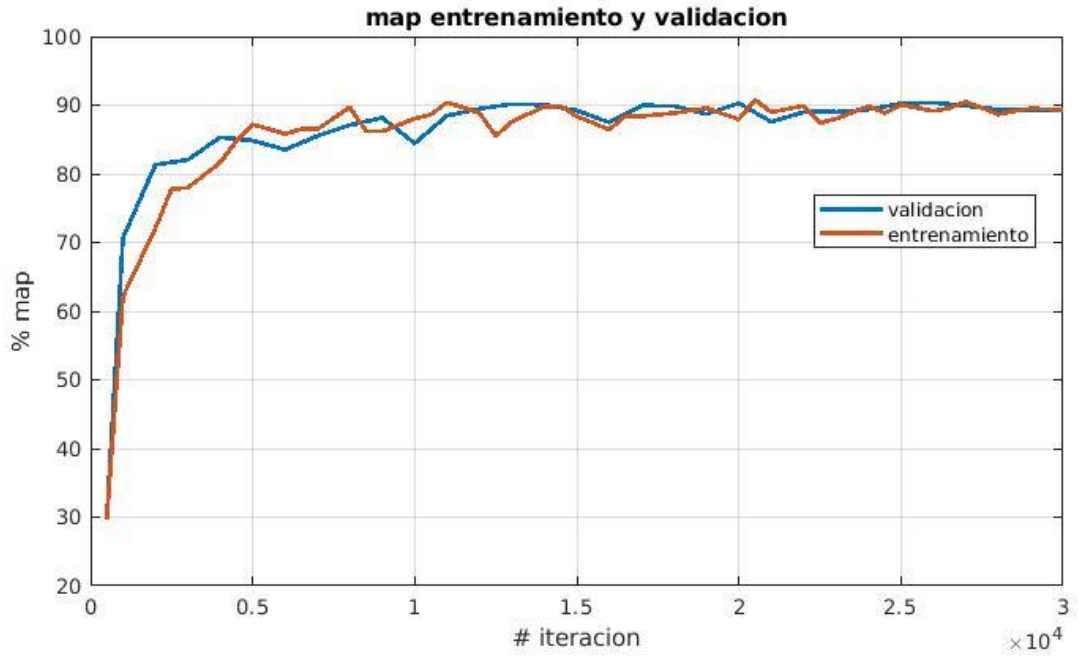


Ilustración 21. Maptrain entrenamiento y validacion.

En la ilustración 21 se puede observar las gráficas de mAP calculado con los archivos de pesos de cada una de las iteraciones en los conjuntos de entrenamiento (naranja) y validación (azul). Como se dijo anteriormente la mejor forma de evaluar los archivos de pesos es con el cálculo del mAP por eso fue necesario realizarlo con el conjunto de imágenes de validación, indicando que no hubo sobre entrenamiento

y permite confirmar y saber que tan útil es el archivo cuando se le presentan nuevas y distintas imágenes a las del conjunto de entrenamiento.

En la gráfica de la ilustración 21 nos permite confirmar que el archivo de pesos de la iteración 21000 es útil para implementarlo en una aplicación real, si bien no fue el valor más alto alcanzado por el mAP para el conjunto de imágenes de validación, se logró un buen porcentaje.

6.2 MobileNet

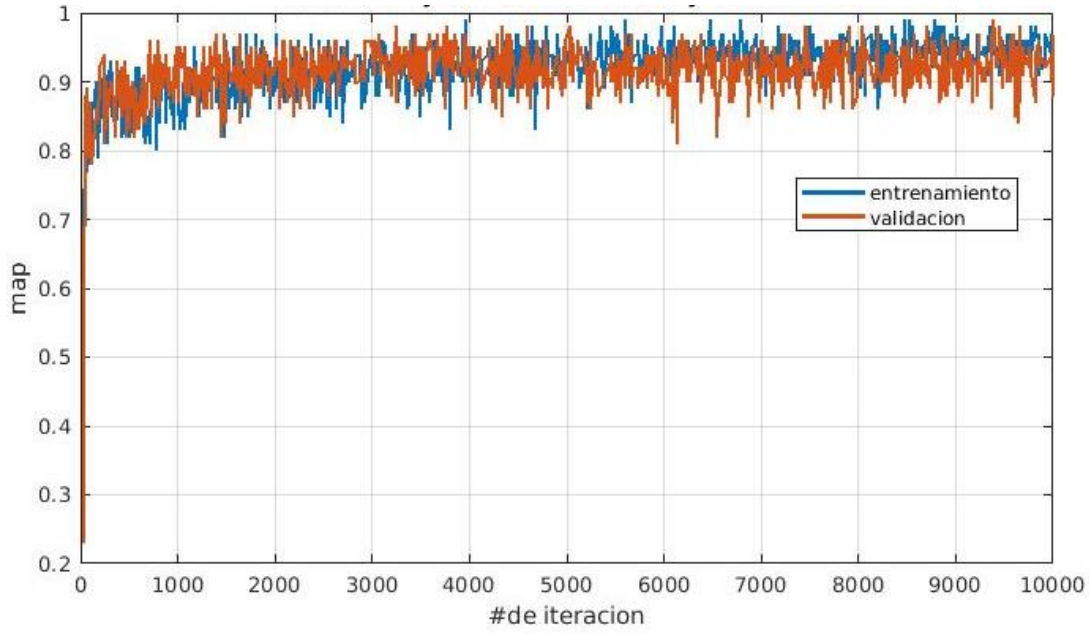


Ilustración 22. Accuracy.

El valor de accuracy indica que tan precisa es la red para detectar objetos, es el equivalente al mAP para medir la precisión de los modelos entrenados en la arquitectura de YOLO, pero se diferencia de esta en que no se tiene en cuenta el IoU. Además, en a topología de TensorFlow se utiliza una función llamada softmax, esta calcula las probabilidades de que el objeto pertenezca a diferentes clases, y puntuando con una probabilidad mayor la que se considera la clase a la que pertenece, cabe decir que en esta función las clases son mutuamente excluyentes por lo tanto si el objeto pertenece a una clase entonces no puede pertenecer a ningún otra, la suma de las probabilidades calculadas por la función en el objeto suman 1.

En los casos en los que no hay objeto en la imagen, o el objeto es difícil de distinguir la función softmax aun así calcula probabilidades para cada clase, siendo esta una de sus desventajas en la implementación.

De la ilustración 22 se calcula el valor de precisión para todas las iteraciones, es fácil de calcular ya que si la predicción es correcta (coincide con la clase verdadera) y supera el umbral del 0.5 calculado con la función softmax entonces es un acierto, para los demás casos es un error. Este porcentaje de acierto se calcula para el 20% de las imágenes usadas en el entrenamiento y un 20% de las imágenes de validación escogidas al azar. Se puede observar en la gráfica que se obtuvo aproximadamente un 90% de acierto en las imágenes de entrenamiento y validación, lo que quiere indicar el entrenamiento en esta arquitectura es un éxito y se puede adecuar en MobileNet para una implementación en una aplicación móvil.

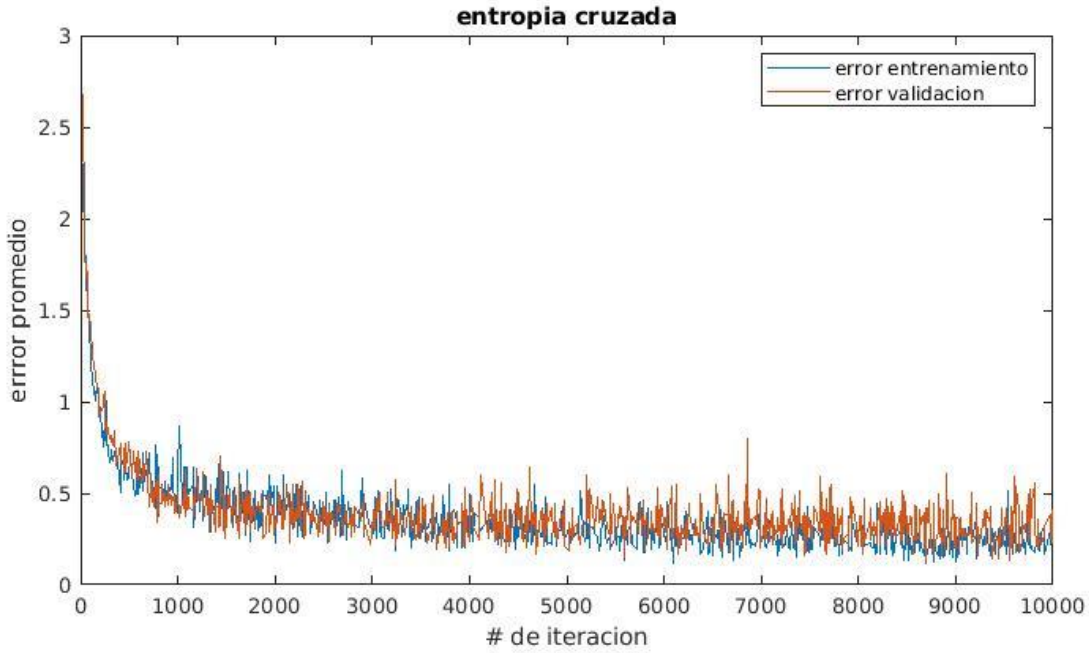


Ilustración 23. Entropía cruzada.

La función de entropía cruzada nos permite cuantificar la diferencia entre dos o mas distribuciones de probabilidad. La distribución verdadera que es la que esperamos que el algoritmo iguale se expresa con una probabilidad absoluta de 1, lo que quiere decir que pertenece únicamente a una clase. Esta función va muy de la mano con la función de Softmax, lo que indica que las predicciones de salida del algoritmo son por ejemplo 0.228 para la "Clase 0", 0.619 para la "Clase 4" y 0.153 para la "Clase 14", la función de entropía cruzada indica que tan cerca esta la predicción predicha de la predicción real y se calcularía para el ejemplo asi "- (0.0*ln(0.228) + 1.0*ln(0.619) + 0.0*ln(0.153))=0.479", se multiplica por 1 porque es la clase verdadera, y en los demás casos se multiplica por 0.

En la ilustración 23 se observa el error promedio calculado con la función de entropía cruzada observando en la gráfica como este valor disminuye a un porcentaje menor al 0.5% siendo un error muy pequeño y confirmando el éxito en el entrenamiento de la red.

Para entrenar YOLO, "Cratos" se gastó 48 horas entrenando con 4 GPU a 12GB, a diferencia de TensorFlow que se demoró 4 horas entrenándose en la CPU de Cratos que posee 256GB de memoria RAM. Comparar ambos modelos no es posible porque tiene enfoques totalmente distintos, TensorFlow está diseñado para correr en procesadores sin mucha potencia, y necesita que se le presente el objeto. En Yolo se le presenta la imagen y él es capaz de detectar la ubicación y la clase del objeto, pero esto se ve reflejado en la necesidad más potencia en los recursos de la máquina.

Para que TensorFlow pueda igualar lo que hace Yolo sería necesario la implementación de métodos de ventaneo exhaustivo que recorran la imagen y de esta forma el modelo de red de TensorFlow pueda predecir las clases, pero esto se ve reflejado en el tiempo que tardaría en reconocer la clase del objeto.

7. CONCLUSIONES Y RECOMENDACIONES

Una vez realizada la identificación de las aves con las redes neuronales convolucionales se aprecia un procesamiento de cómputo rápido y eficaz, capaz de clasificar el orden y la familia taxonómica de las 15 especies de aves de la Saba de Bogotá en segundos con un porcentaje mayor al 70% de acierto en su predicción.

Para la implementación de trabajos futuros, se recomienda ampliar la base de datos con aves, no solo en incluir aves de Bogotá, sino, de todo el país y aumentar el número de muestras por cada dato de ejemplo, Puesto que es la detección por especie en ciertos casos s más fácil detectar que el orden y familia, como, por ejemplo, identificar un Colibrí Aladino y un Colibrí Coruscans tiene un nivel de complicidad alto debido a sus extensas características fisiológicas que a simple vista no se ven o no conocemos dado que no es el área de conocimiento base.

Por otra parte, para el desarrollo de la fase de entrenamiento toma importancia debido a la base de datos ya que la cantidad de patrones a clasificar tiene un valor fundamental para el correcto aprendizaje de la máquina. Igualmente, la idea de usar imágenes con relación de aspecto 4:3 se puede cambiar usando imágenes recortadas y así perfeccionando la precisión en la detección. En este punto, es importante tener en cuenta que para la transferencia de conocimiento no era necesario descargar las imágenes en alta resolución, pues se ahorra bastante tiempo de descarga y se sigue teniendo como salida una etiqueta adecuada. Reducir el ruido de las imágenes también puede ser un factor crucial para mejorar el aprendizaje de las redes, y por consiguiente, para mejorar las medidas de desempeño del sistema. En ese sentido, se sugiere para futuros trabajos implementar estrategias de filtrado más elaboradas.

Adicionalmente, el entrenamiento de las CNN tiene la característica particular de que gracias a la adecuación de su capa convolucional evita que se le tenga que hacer un procesamiento previo a las imágenes a clasificar y seguido a esto, disminuye el número de conexiones en el sistema, lo que genera que el procesamiento sea más rápido.

Reducir el ruido de las imágenes de entrenamiento es un factor crucial para mejorar el aprendizaje de los clasificadores y, por consiguiente, para mejorar las medidas de desempeño del sistema. En ese sentido, se sugiere para futuros trabajos implementar estrategias de filtrado más elaboradas a las que se hicieron en el presente proyecto.

8. BIBLIOGRAFIA

[1] S. Ransbotham, D. Kiron, P. Gerbert, and M. Reeves, "Reshaping business with artificial intelligence.," *MIT Sloan Mangement Rev.*, 2017.

[2] M. Misas A., E. A. López-Enciso, and P. Querubín-Borrero, "La inflación en Colombia : una aproximación desde las redes neuronales," *Ensayos sobre Política Económica*, 2018.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "Yolov2(Cvpr)," *Proc. 2016 IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR 2016)*, 2015.

[4] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2," *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, 2018.

[5] L. M. Renjifo, A. M. Franco-Maya, J. D. Amaya-Espinel, G. H. Kattan, and B. Lopez Lanus, *Libro Rojo de Aves de Colombia*. 2002.

[6] A. Polaszek, "A universal register for animal names," *Nature*. 2005.

[7] B. G. Freeman, S. L. Hilty, C. F. Diego, T. Ellery, and L. E. Urueña, "New and noteworthy bird records from central and northern Colombia," *Cotinga*, 2012.

[8] Audubon and Cornell Lab of Ornithology, "eBird," *eBird*, 2018. .

[9] R. E. Uhrig, "Introduction to Artificial Neural Networks," *Proc. IECON'95-21st Annu. Conf. IEEE Ind. Electron. Vol. 1, IEEE*, 1995.

[10] H. Debes, K., Koenig, A., Gross, "Transfer Functions in Artificial Neural Networks," *Neuroinformatics*, 2005.

[11] S. Haykin, "Neural networks: a comprehensive foundation by Simon Haykin," *The Knowledge Engineering Review*. 1999.

[12] E. N. Sánchez Camperos, A. Y. Alanís García, and E. N. Sánchez Camperos Alanís García,Alma Yolanda., *Redes neuronales :conceptos fundamentales y aplicaciones a control automático*. 2006.
[13] I. Sutskever, G. Hinton, A. Krizhevsky, and R. R. Salakhutdinov, "Dropout : A Simple Way to Prevent Neural Networks from Overfitting," *J. Mach. Learn. Res.*, 2014.
[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *ImageNet Classification with Deep Convolutional Neural Networks*, 2012.
[15] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.
[16] J. Redmon and A. Farhadi, "YOLOv3," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
[17] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.
[18] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2015.
[19] J. Nagi *et al.*, "Max-pooling convolutional neural networks for vision-based hand gesture recognition," in *2011 IEEE International Conference on Signal and Image Processing Applications, ICSIPA 2011*, 2011.
[20] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018.
[21] N. R. Gavai, Y. A. Jakhade, S. A. Tribhuvan, and R. Bhattad, "MobileNets for flower classification using TensorFlow," in *2017 International Conference on Big Data, IoT and Data Science, BID 2017*, 2018.
[22] Y. Leviathan and Y. Matias, "Google AI Blog: Google Duplex: An AI System for Accomplishing Real-World Tasks Over the Phone," *Google Blog*, 2018. .
[23] Y. N. Dauphin *et al.*, "TensorFlow Debugger: Debugging Dataflow Graphs for Machine Learning," *None*, 2016.
[24] Keras, "Keras Documentation," *Loss functions*, 2017. .
[25] L. Du and Y. Du, "Hardware Accelerator Design for Machine Learning," in *Machine Learning - Advanced Techniques and Emerging Applications*, 2018.
[26] C. Gregg and K. Hazelwood, "Where is the data? Why you cannot debate CPU vs. GPU performance without the answer," in *ISPASS 2011 - IEEE International Symposium on Performance Analysis of Systems and Software*, 2011.
[27] NVIDIA, *Cuda C Programming Guide*. 2017.
[28] nvidia, "NVIDIA cuDNN | NVIDIA Developer," *nvidia*, 2019. .
[29] A. Dosovitskiy and T. Brox, "Inverting visual representations with convolutional networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.
[30] F. Nelli and F. Nelli, "Deep Learning with TensorFlow," in *Python Data Analytics*, 2018.
[31] T. Xiao, L. Liu, K. Li, W. Qin, S. Yu, and Z. Li, "Comparison of Transferred Deep Neural Networks in Ultrasonic Breast Masses Discrimination," *Biomed Res. Int.*, 2018.
[32] R. Real and J. M. Vargas, "The probabilistic basis of Jaccard's index of similarity," *Systematic Biology*. 1996.
[33] http://www.icesi.edu.co/wiki_aves_colombia.

9. ANEXOS

Anexo N°1

Archivos base de los datos de aves.

Enlace:
https://drive.google.com/drive/folders/1Bfmb7s2s3MaO3hzNKHaGlcJO9D7aGPyf?usp=sharing

Anexo N°2

Este Script 1 tiene la función de dividir el conjunto de entrenamiento en 80% para entrenamiento y 20% validación.

Enlace:
https://drive.google.com/drive/folders/1U8BQtAXMvaDCz1Kbh5Tx8yh591s47ZR4?usp=sharing

Anexo N° 3

Este Script 2 que tiene la función de abrir la imagen y su archivo de texto correspondiente, tomar cada una de las coordenadas de los objetos encontrados y hacer el recorte de la imagen, quiere decir que se creara un archivo de imagen para cada ave encontrada.

Enlace:
https://drive.google.com/drive/folders/19QZF2Bkp2U1H1Hbqn-2MPxb424-Elz9c?usp=sharing