

# Aplicación de Redes Convolucionales al procesamiento de imágenes

## Data Set sobre la Fauna Silvestre en Colombia

Brahyan Camilo Marulanda Muñoz  
Universidad Autónoma de Occidente  
Yumbo, Valle del Cauca, Colombia  
[brahyan.marulanda@uao.edu.co](mailto:brahyan.marulanda@uao.edu.co)

Daniel Alejandro Tobar Alvarez  
Universidad Autónoma de Occidente  
Cali, Valle del Cauca, Colombia  
[daniel.ale\\_tobar@uao.edu.co](mailto:daniel.ale_tobar@uao.edu.co)

Henry Carmona Collazos  
Universidad Autónoma de Occidente  
Cali, Valle del Cauca, Colombia  
[henry.carmona@uao.edu.co](mailto:henry.carmona@uao.edu.co)

Diego Ivan Perea Montealegre  
Universidad Autónoma de Occidente  
Cali, Valle del Cauca, Colombia  
[diego.perea@uao.edu.co](mailto:diego.perea@uao.edu.co)

**Abstract**— This paper will be mainly focused on the training of a Convolutional Neural Network (CNN) to solve a problem related to the detection of objects using images as input information (Dataset). For the development of the project, the Deep Learning tools Keras and/or TensorFlow will be considered. In the procedure there will be information related to the conformation of the dataset (which will be information related to Wildlife) and the explanation of this based on the definition of the images and classes that will be used in the project (which will serve as the basis for a Monitoring System). In addition to this, the labeling of the images for the training process of the network and, subsequently, the training of the CNN model to be used, its validation and the explanation of the neural network architecture used.

**Keywords**—CNN; Dataset Wildlife; Detection; Keras; Monitoring; Neural Networks; System; Tensorflow.

**Resumen**—El desarrollo del presente documento estará enfocado principalmente al entrenamiento de una Red Neuronal Convolutiva (CNN, por sus siglas en Inglés) que permita resolver un problema relacionado con la detección de objetos usando imágenes como información de entrada (Dataset). Para el desarrollo del proyecto, se tendrán en cuenta las herramientas para Deep Learning Keras y/o TensorFlow. En el procedimiento se tendrá información relacionada con la conformación del dataset (el cual será información relacionada con Fauna Silvestre) y la explicación de este basado en la definición de las imágenes y clases que se va a usar en el proyecto (las cuales servirán como base a un Sistema de Monitoreo). Además de ello, el etiquetado de las imágenes para el proceso de entrenamiento de la red y, posteriormente, el entrenamiento del modelo de CNN a utilizar, su validación y la explicación de la arquitectura de red neuronal usada.

**Palabras Clave**—CNN; Dataset Fauna Silvestre; Detección; Keras; Monitoreo; Redes Neuronales; Sistema; Tensorflow.

### I. INTRODUCCIÓN

Diferentes usos que ha proporcionado la Inteligencia Artificial junto con la Ingeniería han permitido que esta sinergia sea utilizada para abordar problemáticas relacionadas con la medicina, la biología animal, el seguimiento de los patrones de fauna y flora y el control de estos modelos. El uso de la

Inteligencia Artificial (IA) y el Internet de las Cosas (IoT) para ayudar a diversas industrias y sectores a ser más eficientes y productivos, hoy en día se ha convertido en una de las aplicaciones ‘boom’ que es de gran ayuda para proteger a animales en peligro de extinción e impedir que la caza furtiva siga creciendo en muchas partes del mundo [1]. Según Telcel (misma referencia anterior), el laboratorio Teamcore en el Centro de Inteligencia Artificial de la Sociedad de la Universidad del Sur de California desarrolló una aplicación impulsada por IA llamada PAWS (abreviatura de Assistant for Wildlife Security), que ayuda a equipar a los defensores de la vida silvestre con rutas de patrulla optimizadas. Lo anterior debido a que el comercio ilegal de vida silvestre se considera la cuarta empresa criminal más rentable del mundo, después de las drogas, las armas y el tráfico de personas. Un ejemplo más preciso de ello es el mencionado en NatGeo, donde se hace uso de PAWS para controlar la caza furtiva de elefantes africanos todos los días mediante el uso de algoritmos de predicción y toma de decisiones a partir de la información obtenida del monitoreo [2]. Por tal motivo y, teniendo el panorama anterior, el desarrollo del presente documento se enfocará en el entrenamiento de una Red Neuronal Convolutiva en Google Colab utilizando TensorFlow2-Keras que permita resolver el problema de detección de fauna silvestre de los principales animales afectados en Colombia por el tráfico y la caza ilegal; siendo este reto del curso de Redes Neuronales y Deep Learning, la forma en la que el equipo de trabajo podrá adquirir conocimientos experimentales acerca del uso de las Redes Neuronales Convolucionales en los problemas de detección de objetos a partir de la recolección de imágenes y el etiquetado de las mismas, lo cual es útil para la implementación de Sistemas de Monitoreos que controlan los procesos del estado de salud y el seguimiento a especies en vía de extinción o también, sirviendo como la base teórica para llevar a cabo la solución a problemas como la predicción de la caza furtiva de animales, los modelos o patrones del comportamiento de las cadenas tróficas o en sí, el seguimiento de la dinámica de los biomas y la prestación de servicios ecosistémicos.

## II. MARCO TEÓRICO

Existen diferentes conceptos y fundamentos que son necesarios para llevar a cabo el desarrollo que van desde la importación de librerías para el desarrollo del Notebook en Google Collaboratory y finalizan en la realización de la validación del modelo. Por tal motivo, el presente marco teórico contará con conceptos teóricos y las librerías utilizadas en el Notebook de Google Collaboratory. Para el caso del Notebook, se definirán las librerías que se utilizarán para el desarrollo de la codificación:

**pandas** es un paquete de Python que proporciona estructuras de datos rápidas, flexibles y expresivas, diseñadas para el trabajo con datos "relacionales" o "etiquetados" sea fácil e intuitivo. Para este caso específico, su uso es de forma secundaria [3].

**numpy** proporciona un potente objeto array de N dimensiones además de funciones sofisticadas (broadcasting) y útiles funciones de álgebra lineal, transformada de Fourier y números aleatorios [4].

**json** especificado por RFC 7159 (que hace obsoleto a RFC 4627) y por ECMA-404, es un formato de intercambio de datos ligero inspirado en la sintaxis literal del objeto JavaScript. Expone una API familiar a los usuarios de los módulos de la biblioteca estándar `marshal` y `pickle` [5].

El módulo **shutil** ofrece varias operaciones de alto nivel en archivos y colecciones de archivos. En particular, provee funciones que dan soporte a la copia y remoción de archivos. Para operaciones en archivos individuales, véase también el módulo `os` [6].

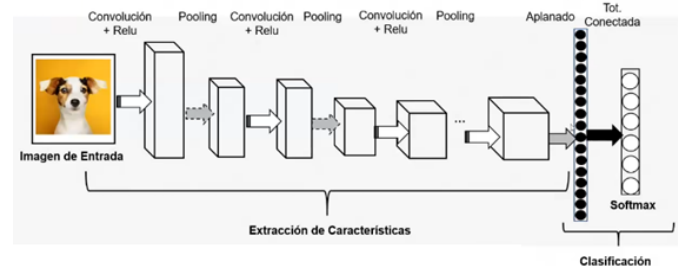
**matplotlib** es una biblioteca completa para crear visualizaciones estáticas, animadas e interactivas en Python. Básicamente, produce figuras de calidad de publicación en una variedad de formatos impresos y entornos interactivos en todas las plataformas. Matplotlib se puede utilizar en scripts de Python, el shell de Python e IPython, servidores de aplicaciones web y varios kits de herramientas de interfaz gráfica de usuario [7].

Además de ello, con la intención de cumplir a cabalidad con los ítems solicitados en la guía, se tiene como base los modelos preentrenados de la API Object detection de Tensorflow presentes en el GitHub Tensorflow 2 Detection Model Zoo, en el cual se encuentran una gran variedad de modelos a partir de arquitecturas, backbouns y tamaño de imágenes de entrada diferentes. Con estos modelos pre-entrenados se realiza el entrenamiento a partir un data set personalizado, correspondiente a un conjunto de imágenes etiquetadas, el cual se requiere se encuentre en archivos TFRecord y Label Maps.

Los **TFRecord** son un tipo de archivo personalizado de TensorFlow cuya estructura es un formato binario sencillo orientado a registros que muchas aplicaciones de TensorFlow usan para entrenar datos. Básicamente, son usados porque facilitan la combinación de múltiples conjuntos de datos y se integra perfectamente con la funcionalidad de importación y

preprocesamiento de datos proporcionada por la biblioteca. Especialmente para los conjuntos de datos que son demasiado grandes para almacenarse completamente en la memoria, esto es una ventaja, ya que solo los datos que se requieren en ese momento (por ejemplo, un lote) se cargan desde el disco y luego se procesan [8]. En complemento a ello, se tiene que el **Label Map** es un archivo con extensión ptxt (que son grafos de texto), el cual mapea las etiquetas de cada una de las clases para los procesos de entrenamiento y predicción, por medio de los labels o etiquetas de cada una de las clases junto con los id asignados en el TFRecord.

Las **Redes Neuronales Convolucionales** (CNN en Ingles), son el enfoque principal del desarrollo del documento. Estas redes son principalmente usadas en procesamiento de imágenes. Las capas que las componen pueden dividirse en dos bloques: el primer bloque, formado principalmente por capas convolucionales y de pooling, cuyo objetivo es la identificación de patrones gráficos, mientras que el segundo bloque tiene como objetivo la clasificación de los datos que reciben. Esto se logra a partir de datos de entrenamiento, los cuales van modificando parámetros intrínsecos (pesos sinápticos y bias) con el objetivo de entrenar mejor la red neuronal y reducir la función de pérdida (loss), es decir, el error de precisión final. Por otra parte, una de las ventajas que ofrece estos tipos de arquitecturas es el hecho de tratarse de un sistema concurrente, es decir, que trabaja en paralelo, ya que varias neuronas pueden estar procesando al mismo tiempo, lo que reduce considerablemente el tiempo de cómputo. En base a lo anterior, se considera que las redes neuronales de cualquier tipo ofrecen grandes resultados gracias a los cuales se continúa invirtiendo tiempo y estudio en estas arquitecturas. Estructuralmente, hacen la convolución y el pooling, luego se aplica un proceso de aplanamiento (flattering) para pasar de una forma matricial a una forma vectorial, esto con el fin de tener una cantidad computacional viable y admisibles para ser introducidos como entradas a redes neuronales que están en capas densas [9].



### A. Redes Preentrenadas en GitHub

Con el objetivo de analizar las diferentes posibilidades de selección de Redes Pre-entrenadas, se hace una búsqueda analítica en la plataforma de GitHub donde se proporciona una colección de modelos de detección previamente entrenados con el dataset COCO 2017 [10].

Según la fuente de referencia [11], COCO es un conjunto de datos de detección, segmentación y subtitulación de objetos a gran escala cuyas características son:

- Segmentación de objetos
- Reconocimiento en contexto
- Segmentación de cosas super pixeladas
- 330K imágenes (>200K etiquetadas)
- 1,5 millones de instancias de objetos
- 80 categorías de objetos
- 91 categorías de objetos
- 5 leyendas por imagen
- 250.000 personas con puntos clave

Tres de las tareas principales de la Visión por Computadora que son tratadas con algoritmos de codificación y que buscan resolver problemas del mundo real desde la extracción de información a partir de imágenes, son la *Clasificación*, *Detección* y *Segmentación*. Para el caso del proyecto, se trae a colación el concepto de **Detección de Objetos**, el cual es el proceso realizado para además de clasificar los objetos en una imagen, saber dónde se encuentran dentro de la misma. Este proceso puede ser llevado a cabo tanto para un único objeto como para múltiples de ellos, los cuales se encuentran o se enmarca a través de bounding boxes. La RN es entrenada con esos bounding boxes y esta devuelve/retorna la localización de las cajas y la probabilidad del label que representa la pertenencia del objeto detectado a una clase denominada. Los *bounding boxes* son entonces los cuadros delimitadores dentro de las imágenes que enmarcan objetos de diferentes clases. En esencia, es un rectángulo que rodea un objeto, especifica su posición, clase y confianza (cuán probable es que esté en esa ubicación). Una *clase* puede ser un grupo semántico representativo de un objeto, animal, humano o cosas ya establecidos (por ejemplo, perro, mazana, gato, casa, coche, serpiente, tenedor, etc.)

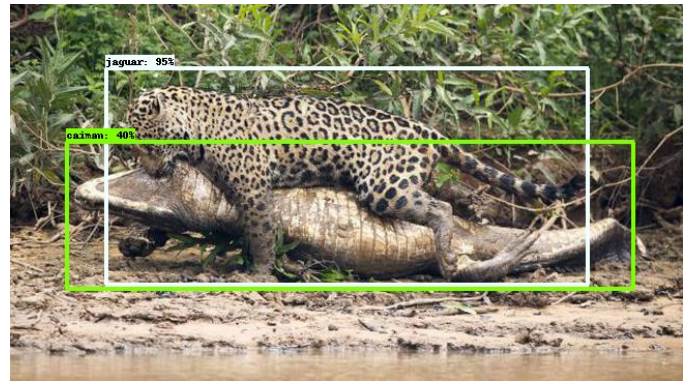
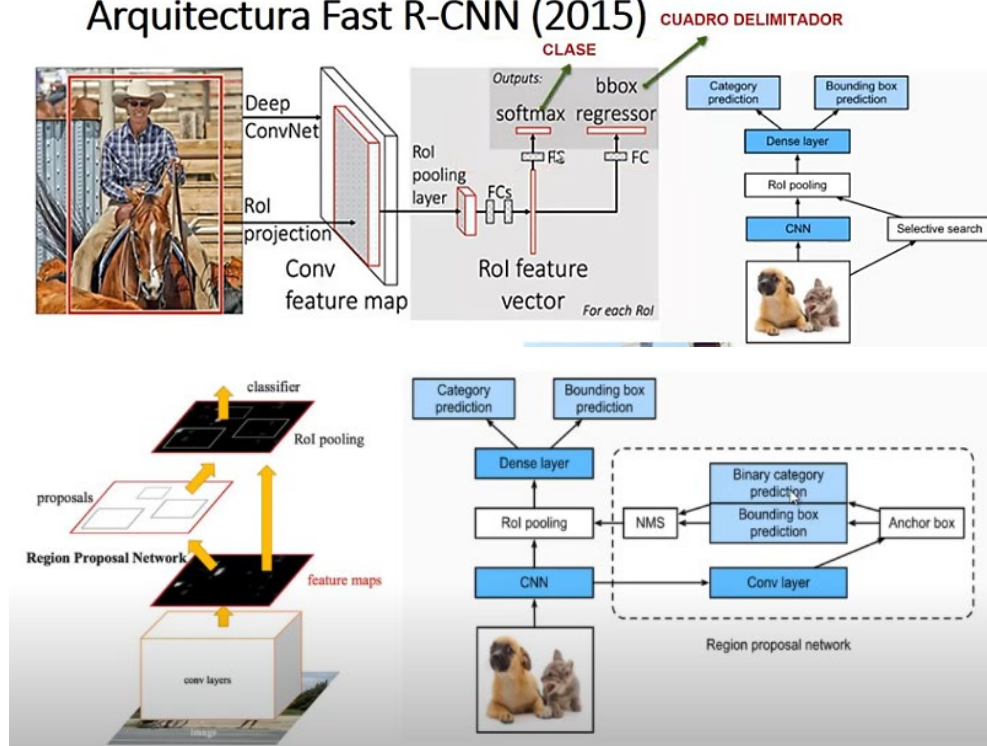


Ilustración 1. Clasificación + Localización = DETECCIÓN DE OBJETOS.

Uno de los modelos recomendados en la guía para el desarrollo del Miniproyecto con el propósito de hacer Detección de Objetos de un Dataset Personalizado es el Modelo **Faster R-CNN**. Esta arquitectura utiliza la búsqueda selectiva para unir las propuestas de región, para ello, utiliza un Módulo de Agrupación de las Regiones de Interés que es conocido en Inglés como RoI Pooling Layer. Este modelo extrae una ventana de tamaño fijo de todo el mapa de características y utiliza dichas características para obtener la etiqueta de la clase final cuyo principal beneficio es que la red neuronal es posible entrenarla de un extremo a otro. Todo lo anterior teniendo como información de entrada unos bounding boxes. El RoI Pooling permite obtener un *vector de características* de tamaño fijo cuya información va a una capa densa [12].

## Arquitectura Fast R-CNN (2015)



<sup>1</sup> Imágenes tomadas de <https://www.youtube.com/watch?v=C-kBqPIZGo8>.



Dentro de su Arquitectura, se tiene el Regional Proposal Network que básicamente son unas redes de propuestas de región que con la implementación de la RPN se evita el uso de algoritmos de búsqueda selectiva. Esto hace que la red sea posible operarla de 7 a 10 fps y con ello, se pueda hacer Detección de Objetos en tiempo real con Deep Learning. Un ejemplo de cómo hacer detecciones es el mencionado en Microsoft donde es destacada la rapidez de la red [13].

Así como esta red, también se cuenta con **ResNet**. En general, en una red neuronal convolucional profunda, varias capas se apilan y se entrenan para la tarea en cuestión. La red aprende varias características de nivel bajo/medio/alto al final de sus capas. En el aprendizaje residual, en lugar de tratar de aprender algunas características, se aprenden algunos residuos. *Residual* puede entenderse simplemente como la sustracción de la característica aprendida de la entrada de esa capa. ResNet hace esto usando conexiones de acceso directo (conectando directamente la entrada de la enésima capa a alguna capa  $(n + x)$ ). Se ha demostrado que entrenar esta forma de redes es más fácil que entrenar redes neuronales convolucionales profundas simples y también se resuelve el problema de la precisión degradante [14]. De sus modelos más destacados, se tiene a *ResNet50* es una variante del modelo ResNet que tiene 48 capas de convolución junto con 1 capa MaxPool y 1 capa de Average Pool layer. Cuenta con  $3.8 \times 10^9$  operaciones de puntos flotantes [15]. *ResNet-101* es una red neuronal convolucional de 101 capas de profundidad. Puede cargar una versión previamente entrenada de la red entrenada en más de un millón de imágenes de la base de datos ImageNet [1]. La red previamente entrenada puede clasificar imágenes en 1000 categorías de objetos, como teclado, mouse, lápiz y muchos animales. Como resultado, la red ha aprendido ricas representaciones de características para una amplia gama de imágenes. La red tiene un tamaño de entrada de imagen de 224 por 224 [16]. Ambos modelos son posibles verlos discriminados en cuanto a código en: <https://keras.io/api/applications/resnet/>.

Estos modelos pueden ser útiles para la inferencia inmediata si el interés es para categorías que ya están en esos conjuntos de datos. También son útiles para modelos de dataset nuevos para la red. Esto se logra a través del uso de un concepto denominado *Transfer Learning*.

## B. Transfer Learning

Este concepto hace referencia a la acción de hacer la transferencia del aprendizaje de un modelo pues cuando se entrenan los modelos ya sea de regresión, redes convolucionales o MLP, se entrenan desde cero, donde se tiene unos valores que se inician aleatoriamente, existe una fase de entrenamiento, se definen la cantidad de épocas, hasta obtener una solución. Con su uso, se aprovecha el aprendizaje que ya se tiene en varios modelos y no se inicia desde cero, sino desde un modelo que ya ha aprendido a reconocer un patrón en específico. Se puede aplicar en modelos propios y modelos desarrollados por otras compañías como Nvidia. Dichos modelos ya creados se usan para clasificar las imágenes propias y solucionar el problema.

De una forma más técnica, según DataScientest [17], el Transfer Learning, o aprendizaje transferido en español, se refiere al conjunto de métodos que permiten transferir

conocimientos adquiridos gracias a la resolución de problemas para resolver otros problemas. Frecuentemente, los modelos utilizados en este campo necesitan grandes tiempos de cálculo y muchos recursos. Sin embargo, utilizando como punto de partida modelos pre-entrenados, el Transfer Learning permite desarrollar rápidamente modelos eficaces y resolver problemas complejos de Computer Vision o Natural Language Processing.

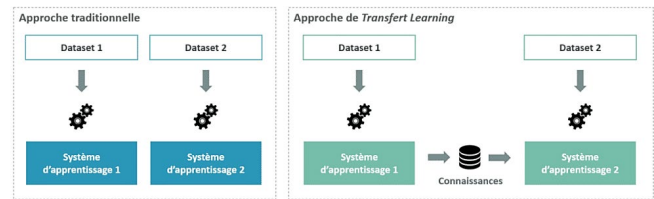


Ilustración 2. Enfoque tradicional vs. enfoque de Transfer Learning.

## C. Selección del Modelo

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
CenterNet MobileNetV2 FPN 512x512	6	23.4	Boxes
CenterNet MobileNetV2 FPN Keypoints 512x512	6	41.7	Keypoints
EfficientDet D0 512x512	39	33.6	Boxes
Faster R-CNN ResNet101 V1 1024x1024	72	37.1	Boxes
Faster R-CNN ResNet101 V1 800x1333	77	36.6	Boxes
Faster R-CNN ResNet152 V1 640x640	64	32.4	Boxes
Faster R-CNN ResNet152 V1 1024x1024	85	37.6	Boxes
Faster R-CNN ResNet152 V1 800x1333	101	37.4	Boxes
Faster R-CNN Inception ResNet V2 640x640	206	37.7	Boxes
Faster R-CNN Inception ResNet V2 1024x1024	236	38.7	Boxes
Mask R-CNN Inception ResNet V2 1024x1024	301	39.0/34.6	Boxes/Masks
ExtremeNet (deprecated)	--	--	Boxes
ExtremeNet	--	--	Boxes

Para hacer la selección del modelo ya entrenado se tienen en cuenta tres parámetros importantes. *Speed* corresponde a la Velocidad de Predicción. *COCO mAP* corresponde al mean Average Precision, el cual es una métrica utilizada para evaluar los modelos de detección de objetos como Fast R-CNN, YOLO, Mask R-CNN, etc. La media de los valores de precisión media (AP) se calcula sobre los valores de recall de 0 a 1 [18]. El *Output* simplemente es la salida del modelo, cuyo interés para la Detección de objetos son los boxes.

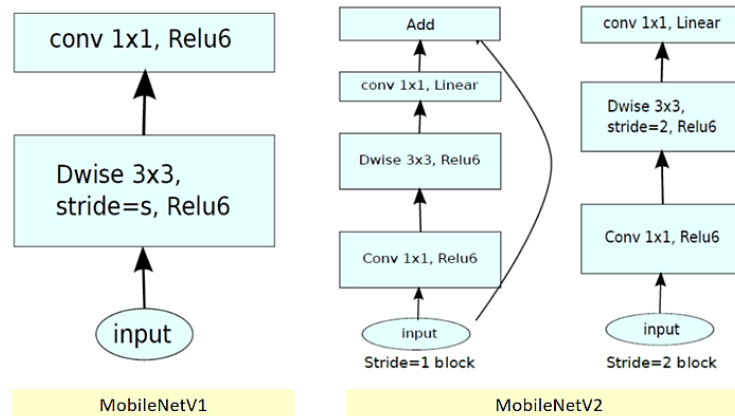
Teniendo presente que nuestra necesidad se ajusta a que se tenga buena precisión, pero no con un alto consumo computacional, como equipo de trabajo se toma la decisión de operar con el modelo **Faster R-CNN ResNet101 V1 640x640**, la cual se encuentra en el repositorio de [http://download.tensorflow.org/models/object\\_detection/tf2/20200711/faster\\_rcnn\\_resnet101\\_v1\\_640x640\\_coco17\\_tpu-8.tar.gz](http://download.tensorflow.org/models/object_detection/tf2/20200711/faster_rcnn_resnet101_v1_640x640_coco17_tpu-8.tar.gz). Sin embargo, se presentó un error en el momento de implementarla, del cual no se tuvo la capacidad de corregirlo debido a que requería configuración de versiones desde el *Pipeline*.

```
--model_dir={model_dir} \
--num_train_steps={num_steps}

2022-10-19 00:01:46.945099: E tensorflow/stream_executor/cuda/cuda_blas.cc:2981] Unable to register cuBLAS factory: Attempting to register factory for plugin cuBLAS when one has already been registered.
2022-10-19 00:01:47.931452: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer.so.7'; dlopen error: libnvinfer.so.7: cannot open shared object file: No such file or directory
2022-10-19 00:01:47.931630: W tensorflow/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer_plugin.so.7'; dlopen error: libnvinfer_plugin.so.7: cannot open shared object file: No such file or directory
2022-10-19 00:01:47.931652: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libraries. If you would like to use Nvidia GPU with TensorRT, you will have to install the current set of dependencies (https://docs.nvidia.com/deeplearning/tensorrt/quickstart.html#prerequisites)
2022-10-19 00:01:52.690290: W tensorflow/core/common_runtime/gpu/gpu_bfc_allocator.cc:42] Overriding orig_value setting because the TF_FORCE_GPU_ALLOW_GROWTH environment variable is set to true.
INFO:tensorflow:Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:GPU:0',)
I1019 00:01:52.709777 140613284411264 mirrored_strategy.py:374] Using MirroredStrategy with devices ('/job:localhost/replica:0/task:0/device:GPU:0',)
INFO:tensorflow:Maybe overwriting train_steps: 1000
I1019 00:01:52.714498 140613284411264 config_util.py:552] Maybe overwriting train_steps: 1000
INFO:tensorflow:Maybe overwriting use_bfloat16: False
I1019 00:01:52.714695 140613284411264 config_util.py:552] Maybe overwriting use_bfloat16: False
Traceback (most recent call last):
  File "/content/models/research/object_detection/model_main_tf2.py", line 114, in <module>
    tf.compat.v1.app.run()
  File "/usr/local/lib/python3.7/dist-packages/tensorflow/python/platform/app.py", line 36, in run
    _run(main=main, argv=argv, flags_parser=_parse_flags_tolerate_undef)
  File "/usr/local/lib/python3.7/dist-packages/absl/app.py", line 308, in run
    _run_main(main, args)
  File "/usr/local/lib/python3.7/dist-packages/absl/app.py", line 254, in _run_main
    sys.exit(main(argv))
  File "/content/models/research/object_detection/model_main_tf2.py", line 111, in main
    record_summaries=FLAGS.record_summaries)
  File "/content/models/research/object_detection/model_lib_v2.py", line 549, in train_loop
    add_summaries=record_summaries)
  File "/content/models/research/object_detection/builders/model_builder.py", line 1253, in build
    add_summaries)
  File "/content/models/research/object_detection/builders/model_builder.py", line 402, in _build_ssd_model
    _check_feature_extractor_exists(ssd_config.feature_extractor.type)
  File "/content/models/research/object_detection/builders/model_builder.py", line 270, in _check_feature_extractor_exists
    'tensorflow'.format(feature_extractor_type, tf_version_str))
ValueError: is not supported for tf version 2. See 'model_builder.py' for features extractors compatible with different versions of TensorFlow
```

Este mismo error ocurrió con los modelos Faster R-CNN ResNet152 V1 640x640, Faster R-CNN ResNet50 V1 640x640 y Faster R-CNN Inception ResNet V2 640x640, inclusive haciendo uso de otros [tutoriales](#). Debido a ello, se decide hacer uso de los modelos **SSD MobileNet v2 320x320** y **SSD MobileNet V1 FPN 640x640**, con el objetivo de utilizar un *backbount* o extractor de características diferente a los utilizados en las arquitecturas mencionadas anteriormente. Estos modelos a pesar de no ser los más precisos y veloces, tienen un

rendimiento relativamente bueno, al considerar que el ultimo brinda una precisión de 29.1, la cual se logra asemejar a algunos de los modelos basados en R-CNN. Las arquitecturas a utilizarse para el desarrollo son las del modelo base **MobileNet**, el cual cuenta con un módulo mejorado con estructura residual invertida, permitiéndole eliminar las no linealidades en las capas estrechas. Este modelo, como columna vertebral para la extracción de funciones, logra prestaciones de última generación para la detección de objetos y la segmentación semántica [19].



El funcionamiento de la arquitectura de MobileNetV2 se da de la siguiente manera desde la caracterización de sus parámetros:

Input	Operator	$t$	$c$	$n$	$s$
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Los parámetros ilustrados en la anterior imantes corresponde a  $t$ : factor de expansión,  $c$ : número de canales de salida,  $n$ : número repetido,  $s$ : zancada. Los núcleos  $3 \times 3$  se utilizan para la convolución espacial.

Por lo general, la red principal (multiplicador de ancho 1,  $224 \times 224$ ), tiene un costo computacional de 300 millones de multiplicaciones y utiliza 3,4 millones de parámetros. Las compensaciones de rendimiento se exploran más a fondo, para resoluciones de entrada de 96 a 224 y multiplicadores de ancho de 0,35 a 1,4. El coste computacional de la red asciende a 585 millones de Madds, mientras que el tamaño del modelo varía entre 1,7 millones y 6,9 millones de parámetros. Para entrenar la red, se utilizan 16 GPU con un tamaño de lote de 96. Una forma visual de los procesos en MobileNetV2 es:

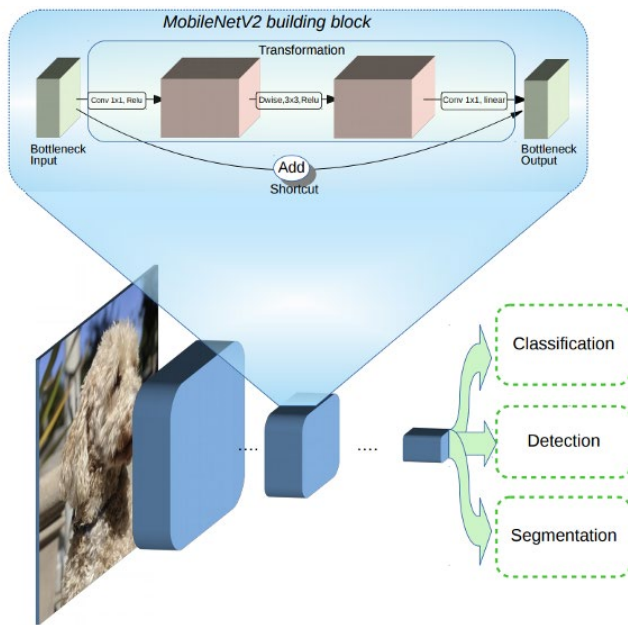


Ilustración 3. Visualización del Modelo en funcionamiento.

La arquitectura MobileNetV2 refleja la buena detección de objetos esto se evidencia añadiéndole SSDlite en la MobileNetV2 debido a que realizando esta combinación en el conjunto de datos de MS COCO frente a otras arquitecturas logra una precisión competitiva con muchos menos parámetros y menor complejidad computacional.

Network	mAP	Params	MAdd	CPU
SSD300[34]	23.2	36.1M	35.2B	-
SSD512[34]	26.8	36.1M	99.5B	-
YOLOv2[35]	21.6	50.7M	17.5B	-
MNet V1 + SSDLite	22.2	5.1M	1.3B	270ms
MNet V2 + SSDLite	22.1	<b>4.3M</b>	<b>0.8B</b>	200ms

En la anterior figura se muestra que el tiempo de inferencia es más rápido que el de MobileNetV1. En particular, MobileNetV2 + SSDLite es 20 veces más eficiente y 10 veces más pequeño, mientras que aún supera a YOLOv2 en el conjunto de datos COCO.

### III. DESCRIPCIÓN DEL PROBLEMA

De forma sintetizada, el equipo de trabajo toma la decisión de trabajar la implementación de la red convolucional relacionada con la Fauna Silvestre en Colombia. Esta noción es definida como el conjunto de organismos vivos de especies animales terrestres y acuáticas, que no han sido objeto de domesticación, mejoramiento genético, cría regular o que han regresado a su estado salvaje (Ley 611 de 2000, Ministerio de Medio Ambiente, Artículo 1) [20]. De acuerdo con lo anterior, existen diferentes características propias de este ecosistema en Colombia, las cuales son:

- No han sufrido procesos de manipulación genética o reproductiva por parte del ser humano.

- Su desarrollo evolutivo y adaptaciones al medio ambiente no han tenido intervención por parte del ser humano.
- Su sistema inmunológico se desarrolla de forma natural sin que sea estimulado por el ser humano (no requieren vacunas).
- Su reproducción se da de forma natural en su hábitat sin ninguna intervención humana.
- Su comportamiento está condicionado por genética y aprendizaje según las características de su entorno, sin ninguna intervención del ser humano.
- Su supervivencia no depende del ser humano, así se hayan acostumbrado a estar en contacto continuo con él.

En aras de tener información relevante relacionada con la situación de la fauna silvestre en Colombia, se consulta una noticia en El Tiempo que indica cuales son las formas de atentar o algunos de los delitos ambientales más comunes del país y sus penalizaciones [21]:

**Aprovechamiento ilícito.** Quien se apropie, acceda, capture, mantenga, introduzca, extraiga, explote, aproveche, exporte, transporte, comercie, explore, trafique o de cualquier otro modo se beneficie de los especímenes, productos o partes de los recursos faunísticos, forestales, florísticos, hidrobiológicos, corales, biológicos o genéticos de la biodiversidad colombiana, incurrirá en prisión de sesenta (60) a ciento treinta y cinco (135) meses, y multa de ciento treinta y cuatro (134) a cuarenta y tres mil setecientos cincuenta (43.750) salarios mínimos legales mensuales vigentes. La pena se aumentará de una tercera parte a la mitad cuando la conducta se cometa a través de la práctica de cercenar aletas de peces cartilaginosos (tiburones, rayas o quimeras), y descartar el resto del cuerpo al mar.

**Tráfico de fauna.** El que trafique, adquiera, exporte o comercialice sin permiso de la autoridad competente o con incumplimiento de la normatividad existente los especímenes, productos o partes de la fauna acuática, silvestre o especies silvestres exóticas, incurrirá en prisión de sesenta (60) a ciento treinta y cinco (135) meses, y multa de trescientos (300) hasta cuarenta mil (40.000) salarios mínimos legales mensuales vigentes. La pena se aumentará de una tercera parte a la mitad cuando la conducta se cometa a través de la exportación o comercialización de aletas de peces cartilaginosos (tiburones, rayas o quimeras).

**Deforestación.** El que sin permiso de autoridad competente o con incumplimiento de la normatividad existente tale, queme, corte, arranque o destruya áreas iguales o superiores a una hectárea continua o discontinua de bosque natural, incurrirá en prisión de sesenta (60) a ciento cuarenta y cuatro (144) meses, y multa de ciento treinta y cuatro (134) a cincuenta mil (50.000) salarios mínimos legales mensuales vigentes. En los siguientes casos, la pena se aumentará a la mitad:



1. Cuando la conducta se realice para acaparamiento de tierras, para cultivos de uso ilícito o para mejora o construcción de infraestructura ilegal.
2. Cuando la conducta afecte más de 30 hectáreas contiguas de extensión o cuando en un periodo de hasta seis meses se acumule la misma superficie deforestada.

**Dañon en los recursos naturales y ecocidio.** Quien destruya, inutilice, haga desaparecer o cause un impacto ambiental grave o de cualquier otro modo dañe los recursos naturales, incurrirá en prisión de sesenta (60) a ciento treinta y cinco (135) meses, y multa de ciento sesenta y siete (167) a dieciocho mil setecientos cincuenta (18.750) salarios mínimos legales mensuales vigentes.




**Contaminación ambiental.** El que contamine, provoque o realice directa o indirectamente emisiones, vertimientos, radiaciones, ruidos, depósitos o disposiciones al aire, la atmósfera o demás componentes del espacio aéreo, el suelo, el subsuelo, las aguas superficiales, marítimas o subterráneas o demás recursos naturales en tal forma que contamine o genere un efecto nocivo en el ambiente, que ponga en peligro la salud humana y los recursos naturales, incurrirá en prisión de sesenta y nueve (69) a ciento cuarenta (140) meses, y multa de ciento cuarenta (140) a cincuenta mil (50.000) salarios mínimos legales mensuales vigentes.




De todos los aspectos anteriormente mencionados, el desarrollo del mini proyecto estará enfocado en la Caza y el Tráfico ilegal de fauna silvestre; esto debido a que se han afectado 234 especies de aves, 76 de mamíferos, 27 de reptiles

y 9 de anfibios, siendo Colombia el país que ocupa el segundo lugar en tráfico de especies a nivel global [22]. Por otra parte, es el principal responsable de una considerable disminución en las poblaciones de muchas especies, lo cual es impulsado en el hecho que, a escala mundial, más de 6.000 millones de animales silvestres se comercializan anualmente para atender la demanda de animales vivos para mascotas en hogares, zoológicos y para laboratorios [23].

Con el objetivo de Controlar la red trófica y monitorear los ecosistemas y el desarrollo de los animales en peligro de extinción, se realizará un **Modelo de Detección de Animales**, el cual podrá entregar la ubicación del bounding box y a partir del cálculo de distancias entre ellos, tomar decisiones en dos aspectos. Uno de ellos, es para el *Control de Animales en vía de extinción*, mediante el cual se tenga una vigilancia no nociva ni armada de los animales en sus hábitats naturales con un sistema de cámaras y el otro derivado del primero, corresponde al *Control de la Cadena alimenticia* de estos animales y cuál es la interrelación entre ellos y la mano del hombre. Para ambos casos, es de suma importancia que el ser humano pueda, desde lo racional, llevar a cabo una idea de prevenir y reducir la alteración de las cadenas alimenticias, tomando acciones como evitar la tala de árboles, reducir la contaminación de fuentes hídricas, no ocasionar incendios en áreas verdes y respetar los tiempos de animales acuáticos. No obstante, se recalca que este sistema de monitoreo no será realizado en el curso, sino que se alcanzará a tener el algoritmo que permita detectar los diferentes animales silvestres, subdivididos en 6 clases, relacionados así:

Tabla 1. Dataset.

Animal	Rol	Descripción	#Img
<b>Caimán Negro</b> 	Depredador/ Presa (Jaguar)	Vive en lagos, ríos y otros hábitats de agua dulce, principalmente en la cuenca del Amazonas entre Colombia, Ecuador y Perú. Es el mayor depredador de estos humedales. Gracias a su potente vista y audición, caza durante la noche. La caza ilegal y la pérdida y fragmentación de su hábitat son sus mayores amenazas [24].	63
<b>Guacamaya Roja</b> 	Presa (Ocelote)	Las guacamayas se distribuyen en las tierras bajas de la planicie del Caribe y en toda la región Orinoco-Amazónica; también en los valles interandinos del Cauca y el Magdalena, en la región pacífica del Darién colombiano y en las estribaciones de las Cordilleras. Se alimentan principalmente de semillas y dependen de la existencia de árboles o palmas huecas en el bosque para poderse reproducir [25].	57
<b>Iguana</b> 	Presa (Ocelote)	En Colombia la iguana verde habita en bosques, siempre y cuando estén cerca de espejos de agua. Pueden encontrarse no solo en el bosque húmedo tropical, igualmente en el bosque seco, bosque de galería, sabanas con poca vegetación arbórea, y hasta islas o costas con vegetación exclusivamente arbustiva. La iguana es una de las especies mayormente traficadas en Colombia para tenencia o comercialización, muchas iguanas han sido comercializadas para consumo humano, no solo por su carne, sino también por sus huevos (entre 30-50 huevos) para consumo, utilizan sus partes como carnada para la pesca o para peletería [26].	66

<b>Jaguar</b> 	Depredador	Su población en el mundo se ha reducido en un 45% y en países como Estados Unidos, El Salvador y Uruguay está declarado como extinto. En Colombia, las poblaciones más grandes de jaguares se encuentran en el Amazonas, la Orinoquia, Chocó y el Caribe. Esta especie necesita grandes extensiones de tierra para sobrevivir, por eso la pérdida del hábitat, el desarrollo de infraestructura, la expansión de las actividades agrícolas y ganaderas, y la pérdida de vegetación son sus grandes amenazas [24].	84
<b>Ocelote</b> 	Depredador	Habita ecosistemas por debajo de los 2,000 msnm a lo largo de su distribución en Colombia. El ocelote comparte los hábitats de baja altura con jaguares, pumas y margays. Prefieren usar caminos, senderos y carreteras, pero marca menos frecuentemente con rascas que el puma y el jaguar. Esta especie estuvo muy amenazada por la cacería para suplir la demanda de pieles manchadas en la década de los 60's y principios de los 70's. Hubo miles de individuos que fueron cazados por indígenas y colonos principalmente en las regiones de la Orinoquia y la Amazonia.  Información tomada de: <a href="https://www.cvc.gov.co/sites/default/files/Planes_y_Programas/Plan-de-accion-felino.pdf">https://www.cvc.gov.co/sites/default/files/Planes_y_Programas/Plan-de-accion-felino.pdf</a> .	81
<b>Puma</b> 	Depredador	Este hermoso felino de color leonado se encuentra en bosques secos y húmedos y en el páramo en altitudes de hasta 4100 msnm. La especie puede ocupar prácticamente todos los hábitats, a excepción de aquellos inundables. Por lo general, habitan bosques maduros e intervenidos. La intervención humana ha ocasionado una reducción en sus principales presas que son los roedores, lo cual ha logrado cercanía de estos con los cazadores ilegales o traficantes [27].	79

#### A. Desarrollo del Etiquetado en LinkedAI

LinkedAI es una plataforma web que proporciona una solución integral para la anotación de imágenes con herramientas de etiquetado rápidas, generación de datos sintéticos, gestión de datos, funciones de automatización y servicios de anotación bajo demanda con herramientas integradas para acelerar y finalizar proyectos de visión artificial. El equipo de trabajo hizo uso de este aplicativo para la implementación de los etiquetados de los diferentes animales que se tienen en el dataset.

*Etiquetado.* Cada uno de los animales cuenta con una variable semántica (observar Ilustración 5) que se encuentra acompañada de un recuadro denominado bounding box, con el que se trata de enmarcar el área que recubra la mayor cantidad de información de cada uno de los seis animales.



Ilustración 4. Etiquetado rápido con Bounding Box.

*Cantidad de Datos.* Para que la CNN quede en lo posible, competentemente entrenada, se toma la decisión de proporcionar un enfoque preciso en los felinos, que son los depredadores de las redes tróficas que se están manipulando para el proyecto y por tal motivo, cuentan con una cantidad promedio de recolección de imágenes de 81 imágenes.

Asimismo, para el caso de las presas, que también son clases dentro del conjunto de datos de fauna silvestre, se tuvo en cuenta un enfoque secundario de un valor promedio de 62 imágenes.

Labels (543)			ALL	SAVED
Categories	Count	%		
Jaguar	117	22%		
Ocelote	105	19%		
Guacamaya	92	17%		
Caiman	84	15%		
Puma	76	14%		
Iguana	69	13%		
_delete	10	2%		

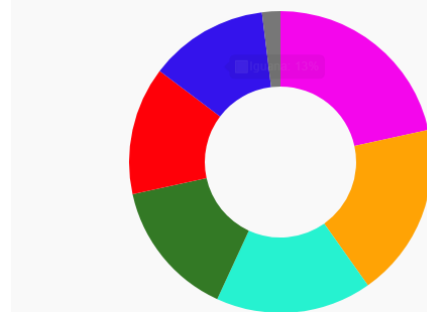


Ilustración 5. Gráfico resumen. Jaguar fue la clase con más etiquetas dentro del proceso, 22%.



Luego de realizar todo el etiquetado, surge una dificultad en el momento de utilizar la información proporcionada por esta herramienta debido a que la plataforma entrega un archivo con extensión Json y el requerido por los modelos pre entrenados de TensorFlow es TFRecord. Para esto, se contaba con un código en Colab que, en el momento de ejecutarse, arrojaba errores y por tal motivo no se hace uso de él. Por eso, se tomó la decisión de utilizar la herramienta de Roboflow utilizando la misma cantidad de imágenes.

#### B. Desarrollo del Etiquetado en Roboflow

“Con Roboflow, los clientes pueden anotar imágenes mientras evalúan la calidad de los conjuntos de datos para prepararlos para la capacitación. (La mayoría de los algoritmos de visión artificial requieren etiquetas que esencialmente «enseñan» al algoritmo a clasificar objetos, lugares y personas). La plataforma permite a los desarrolladores experimentar para generar nuevos datos de entrenamiento y ver qué configuraciones conducen a un mejor rendimiento del modelo. Una vez que se completa el entrenamiento, Roboflow puede implementar el modelo en la nube, el borde o el navegador y monitorear el modelo en busca de casos extremos y degradación con el tiempo” [28].

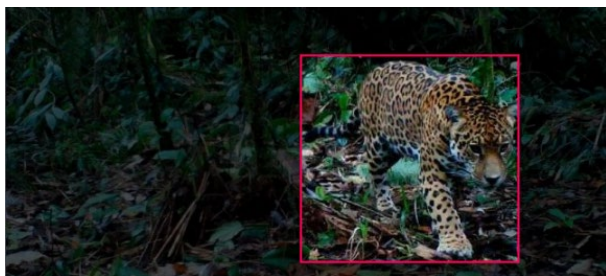


Ilustración 6. Etiquetado.

Al igual que con LinkedAI, se tuvieron en cuenta la misma cantidad de imágenes, pero aquí, ya se obtuvo un archivo TFRecord. Además, de las 424 imágenes que se utilizaron para hacer el etiquetado y el entrenamiento, hubo cierto porcentaje dividido en lo que corresponde el Entrenamiento como tal, pero también una Validación y Testeo, así:

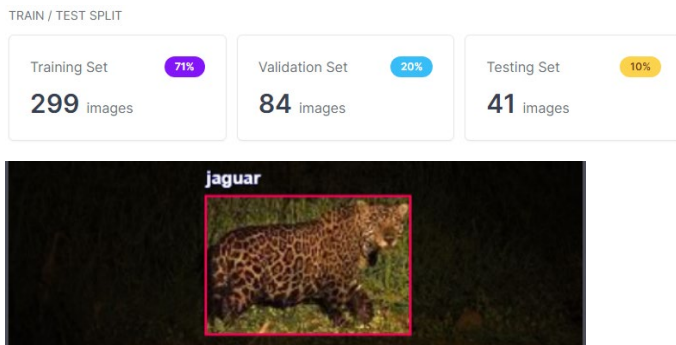


Ilustración 7. Validación.

Otro aspecto importante de esta herramienta es que permite hacer un preprocesamiento de las imágenes correspondiente a un redimensionamiento (resize with edge) de las mismas, dejándolas tomas de forma unánime en el tamaño requerido por cada modelo de MobileNet.

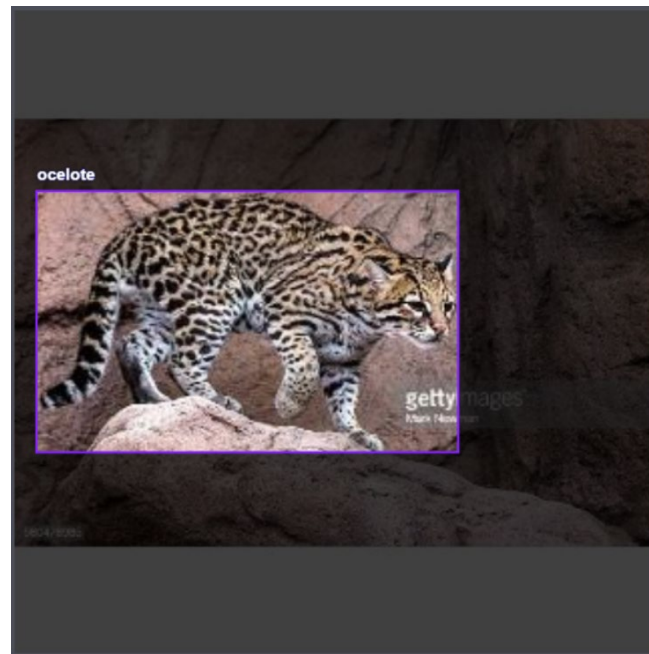


Ilustración 8. Redimensionamiento.

#### C. Planteamiento del Enunciado

La Guía del Miniproyecto del Curso, orientado por el docente Andrés Felipe Escobar<sup>2</sup> propone entrenar una Red Neuronal Convocucional que permita resolver un problema relacionado de detección de objetos, de segmentación (semántica o de instancia) o de estimación de postura usando imágenes como información de entrada. Para ello, se recomienda usar las herramientas Deep Learning Keras y/o TensorFlow.

#### D. Metodología

Para la realización del proyecto, primero se realizó una búsqueda considerable de los modelos que se encuentran en la API de TensorFlow. Tal como se mencionó en el marco teórico, cuando se tuvo la plena intención de implementar 3 de los modelos que se tenían, ninguno dio resultados ni siquiera presentables para evidenciarlos debido a conflictos que existían entre Colab y la versión de Tensorflow que tenían los modelos. Por tal motivo, haciendo uso del modelo SSD MobileNet V1 FPN 640x640. Se entrenó el modelo con imágenes de 640x640 obtenidas por el redimensionamiento de Roboflow. De acuerdo con ello, la metodología con sus respectivos pasos necesarios que se tuvieron en cuenta en el desarrollo, fueron:

<sup>2</sup> Docente de la asignatura de Redes Neuronales y Deep Learning, aescobaro@uao.edu.co.

1. Definición de las imágenes y clases que se va a usar en el proyecto.
2. Etiquetado de las imágenes para el proceso de entrenamiento de la red.
3. Entrenamiento del modelo de red neuronal a utilizar, utilizando alguno de los múltiples tutoriales en internet.
4. Validación del modelo de red neuronal utilizado.
5. Explicación de la arquitectura de red neuronal usada.

#### IV. DESARROLLO DE LA PROPUESTA

Para la realización del proyecto, se tuvieron en cuenta dos Notebooks en Google Colab proporcionados por el docente a cargo, los cuales son:

##### ENTRENAMIENTO DEL MODELO.

[Model training detection SSD MobileNet\\_V1\\_FPN\\_640x640.ipynb](#).

Inicialmente, se importan e instalan las librerías necesarias para el reentrenamiento de los modelos y la manipulación de componentes como directorios y archivos del sistema operativo.

```
import os
import pandas as pd
import json
import shutil

!pip install tf_slim
```

Seguido a ello, se realiza la obtención del repositorio models de Tensorflow y se realiza el desplazamiento hasta la carpeta research.

```
%cd /content
!git clone --
quiet https://github.com/tensorflow/models.git
%cd /content/models/
%cd /content/models/research
!protoc object_detection/protos/*.proto --
python_out=.
os.environ['PYTHONPATH'] += ':/content/models/rese
arch:/content/models/research/slim/'
```

Posteriormente se realiza la obtención del modelo, para lo cual, se obtiene el enlace del GitHub Tensorflow 2 Detection Model Zoo y se almacena en content.

```
!wget --no-check-
certificate http://download.tensorflow.org/models/
object_detection/tf2/20200711/ssd_mobilenet_v1_fpn
_640x640_coco17_tpu-8.tar.gz \
```

```
_
O /content/ssd_mobilenet_v1_fpn_640x640_coco17_tpu
-8.tar.gz
```

Con el modelo descargado, se procede a extraer o descomprimir los archivos del mismo en una carpeta dentro de research con el nombre del modelo.

```
!tar -
zxvf /content/ssd_mobilenet_v1_fpn_640x640_coco17_
tpu-8.tar.gz
output_path = 'ssd_mobilenet_v1_fpn_640x640_coco17
_tpu-8'
output_path = os.path.join(os.getcwd(), output_pat
h)
print("La carpeta se almacenó en {}".format(output
_path))
```

Se crea una carpeta o directorio en el que se almacenaran los archivos de configuración del pipeline.

```
path_training = '/content/ssd_mobilenet'
os.mkdir(path_training)
```

Se mueven los archivos de configuración a la carpeta ssd\_mobilenet

```
source_config = "{}pipeline.config".format(output
_path)
target_config = "{}pipeline.config".format(path_t
raining)
shutil.copyfile(source_config, target_config)
```

Ahora, se realiza la importación de ciertas librerías necesarias para la ejecución del pipeline y el entrenamiento.

```
import tensorflow as tf
from object_detection.utils import config_util
from object_detection.protos import pipeline_pb2
from google.protobuf import text_format
```

Se almacena el archivo de configuración en la variable config con la cual se procede a enseñarla, en la cual se puede observar que el modelo contaba con 90 clases y había sido entrenado con 25000 steps.

```
config = config_util.get_configs_from_pipeline_fil
e(target_config)
```

Se realiza la configuración del archivo de configuración para que pueda ser editado posteriormente con el nuevo número de clases, datos de entrenamiento y demás.

```
pipeline_config = pipeline_pb2.TrainEvalPipelineCo
nfig()
with tf.io.gfile.GFile(target_config, "r") as f:
```

```
proto_str = f.read()
text_format.Merge(proto_str, pipeline_config)
```

Se almacena la ruta de los archivos de entrenamiento y validación, correspondiente a los TFRecord y Label Map.

```
label_map_pbtxt_fname = "/content/drive/MyDrive/Parcial2/TFRecord/animals_label_map.pbtxt"
train_record_fname = "/content/drive/MyDrive/Parcial2/TFRecord/train/animals.tfrecord"
test_record_fname = "/content/drive/MyDrive/Parcial2/TFRecord/valid/animals.tfrecord"
```

Se procede a editar los datos del archivo de configuración del pipeline, con el cual se indica que se hará uso de 6 clases, correspondiente a Jaguar, Ocelote, Puma, Caimán, Guacamayo e Iguana, también se indica el tamaño del lote de los datos, los checkpoint del modelo y por último, se indica la ruta de los archivos de validación y entrenamiento, correspondiente a los TFRecord y label Map.

```
pipeline_config.model.ssd.num_classes = 6
pipeline_config.train_config.batch_size = 4
pipeline_config.train_config.fine_tune_checkpoint = "{}checkpoint/ckpt-0".format(output_path)
pipeline_config.train_config.fine_tune_checkpoint_type = "detection"
pipeline_config.train_input_reader.label_map_path = label_map_pbtxt_fname
pipeline_config.train_input_reader.tf_record_input_reader.input_path[0] = train_record_fname
pipeline_config.eval_input_reader[0].label_map_path = label_map_pbtxt_fname
pipeline_config.eval_input_reader[0].tf_record_input_reader.input_path[0] = test_record_fname
```

Se realiza otros ajustes a los archivos de configuración.

```
config_text = text_format.MessageToString(pipeline_config)
with tf.io.gfile.GFile(target_config, "wb") as f:
    f.write(config_text)
```

Se instalan otras librerías necesarias para el reentrenamiento del modelo.

```
!pip install lvis
!pip install tf-models-official
!pip install tensorflow-io
```

Con la personalización de los archivos de configuración, se procede a reentrenar el modelo con 5000 épocas, en el cual

se guardan los checkpoints del entrenamiento en una carpeta dentro del drive.

```
num_steps = 5000
model_dir = "/content/drive/MyDrive/Parcial2/Models_saved"

!python /content/models/research/object_detection/
model_main_tf2.py \
--pipeline_config_path={target_config} \
--model_dir={model_dir} \
--num_train_steps={num_steps}
```

Con el modelo reentrenado, se procede a cargar los datos obtenidos del entrenamiento en Tensorboard. Resultado presentado en la Ilustración 9, en la cual se puede observar como el modelo en tan solo 5000 épocas logra llegar a valores de pérdida en la clasificación y localización, relativamente pequeños alrededor de 0.05 y 0.02 respectivamente.

```
%load_ext tensorboard
%tensorboard --
logdir "/content/drive/MyDrive/Parcial2/Models_saved"
```

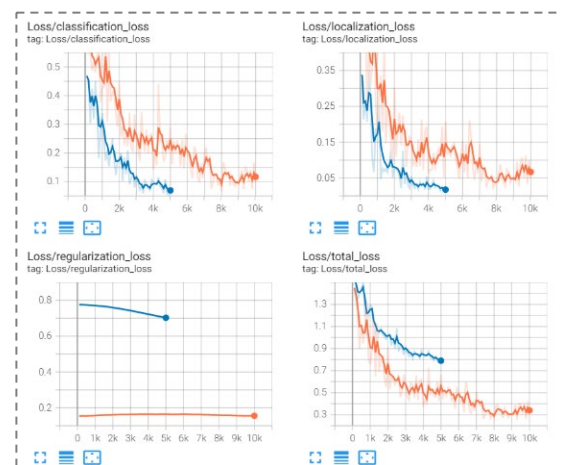


Ilustración 9. Evolución de las pérdidas entregada por Tensorboard. La curva azul es la del Modelo SSD MobileNet V1 FPN 640x640 y la curva naranja es la del modelo SSD MobileNet V2 320x320.

Con el modelo ya entrenado, se procede a salvarlo o guardarlo con el fin de poder utilizarlo en las etapas de inferencia y posteriores aplicaciones de ser necesario.

```
model_dir = "/content/drive/MyDrive/Parcial2/Models_saved/v3_640X640/checkpoint1"
output_directory = "/content/drive/MyDrive/Parcial2/Models_saved/v3_640X640/Model_3"

!python /content/models/research/object_detection/
exporter_main_v2.py \
--input_type image_tensor \
```



```
--pipeline_config_path {target_config} \
--trained_checkpoint_dir {model_dir} \
--output_directory {output_directory}
```

Finalmente, se comprime el modelo sintonizado o afinado (fine tuned) en formato zip.

```
!zip -
r /content/drive/MyDrive/Parcial2/Models_saved/v3_
640X640/Model_3/fine_tuned_model3.zip /content/dri
ve/MyDrive/Parcial2/Models_saved/v3_640X640/Model_
3
```

## INFERENCIA.

[Inference detector model SSD MobileNet V1 FPN 640x640.ipynb](#).

Inicialmente, se instalan las librerías necesarias para el proceso de inferencia del modelo reentrenado.

```
!pip install tf_slim
!pip install lvis
!pip install tf-models-official
```

De igual forma, se instalan ciertas librerías, repositorios y se realiza el desplazamiento a la carpeta de research.

```
%cd /content
!git clone --
quiet https://github.com/tensorflow/models.git
%cd /content/models/
#!git checkout 58d19c67e1d30d905dd5c6e5092348658fe
d80af
!apt-get update && apt-get install -y -
qq protobuf-compiler python-pil python-
lxml python-tk
!pip install -
q Cython contextlib2 pillow lxml matplotlib
!pip install -q pycocotools
%cd /content/models/research
!protoc object_detection/protos/*.proto --
python_out=.
os.environ['PYTHONPATH'] += ':/content/models/rese
arch:/content/models/research/slim/'
!python object_detection/builders/model_builder_te
st.py
```

En esta sección, se importan las librerías necesarias para la inferencia del modelo y la visualización de imágenes.

```
import cv2
import zipfile
```

```
from object_detection.utils import label_map_util
from object_detection.utils import visualization_u
tils as viz utils
import tensorflow as tf
import numpy as np
```

```
from PIL import Image
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
```

Posteriormente, se descomprime el modelo sintonizado o afinado y se ubica en el directorio de content.

```
path = '/content/drive/MyDrive/Parcial2/Models_sav
ed/v3_640X640/Model_3/fine_tuned_model3.zip'
zip_ref = zipfile.ZipFile(path, "r")
zip_ref.extractall("/content/fine_tuned_model")
zip_ref.close()
```

Se carga el modelo reentrenado.

```
PATH_TO_MODEL_DIR = "/content/fine_tuned_model/con
tent/drive/MyDrive/Parcial2/Models_saved/v3_640X64
0/Model_3"
```

```
PATH_TO_SAVE_MODEL = PATH_TO_MODEL_DIR + "/saved_m
odel"
```

```
detect_fn = tf.saved_model.load(PATH_TO_SAVE_MODEL
)
```

Se almacena en una variable la ruta del label map y se obtienen los índices de las clases a partir de este archivo.

```
PATH_TO_LABELS = "/content/drive/MyDrive/Parcial2/
TFRecord/animals_label_map.pbtxt"
category_index = label_map_util.create_category_in
dex_from_labelmap(PATH_TO_LABELS, use_display_name
=True)
```

Se crea una función encargada de leer la imagen ubicada en la ruta que se le entregue como argumento, posteriormente, con la lectura en forma de arreglo, se convierte a tensor y se realiza la detección.

```
def read_image(path):
    image_path = path

    image_np = np.array(Image.open(image_path))

    input_tensor = tf.convert_to_tensor(image_np)
    input_tensor = input_tensor[tf.newaxis, ...]

    detections = detect_fn(input_tensor)
```

```
num_detections = int(detections.pop('num_detections'))
```

```
return detections, num_detections, image_np
```

Se crea otra función, la cual se encarga de hacer posible la visualización de la detección realizada, ubicando los bounding boxes en los respectivos objetos.

```
def show_detection(detections, num_detections, image_np):
    detections2 = {key: value[0:num_detections].numpy() for key, value in detections.items() }
```

```
detections2['num_detections'] = num_detections
```

```
detections2['detection_classes'] = detections2['detection_classes'].astype(np.int64)
```

```
# Copy image and draw bounding-boxes
```

```
image_np_with_detections = image_np.copy()
```

```
# print(np.array(image_np_with_detections).shape
```

```
)
```

```
# Use object detection library to show the classification and boundingboxes
```

```
viz_utils.visualize_boxes_and_labels_on_image_array(
```

```
    image_np_with_detections,
```

```
    detections2['detection_boxes'],
```

```
    detections2['detection_classes'],
```

```
    detections2['detection_scores'],
```

```
    category_index,
```

```
    max_boxes_to_draw=15,
```

```
    min_score_thresh=0.40,
```

```
    use_normalized_coordinates = True
```

```
)
```

```
# Detection result
```

```
image_np_with_detections = cv2.cvtColor(image_np_with_detections, cv2.COLOR_BGR2RGB)
```

```
return image_np_with_detections
```

Finalmente, a partir de un conjunto de imágenes ubicada en una carpeta, indicada como `folder_path`, se realiza la concatenación del nombre de 50 imágenes y la dirección de la carpeta, se realiza la detección y finalmente se enseña en un `figure`.

```
folder_path = "/content/drive/MyDrive/Imagenes"
```

```
#path = "/content/images/frame_1-4_3516.jpg"
```

```
count = 0
```

```
for im in os.listdir(folder_path):
```

```
    path = folder_path + '/' + im
```

```
    print(im)
```

```
    (detections, num_detections, image_np) = read_image(path)
```

```
    image_np_with_detections = show_detection(detections, num_detections, image_np)
```

```
    plt.figure()
```

```
    cv2.imshow(image_np_with_detections)
```

```
if count == 50:
```

```
    break
```

```
count += 1
```

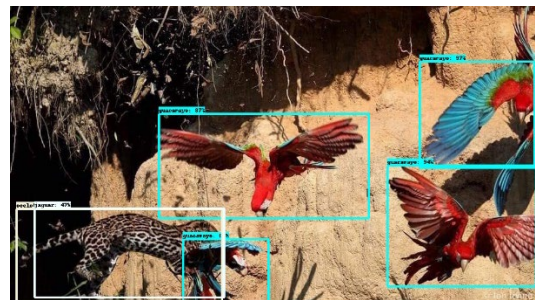


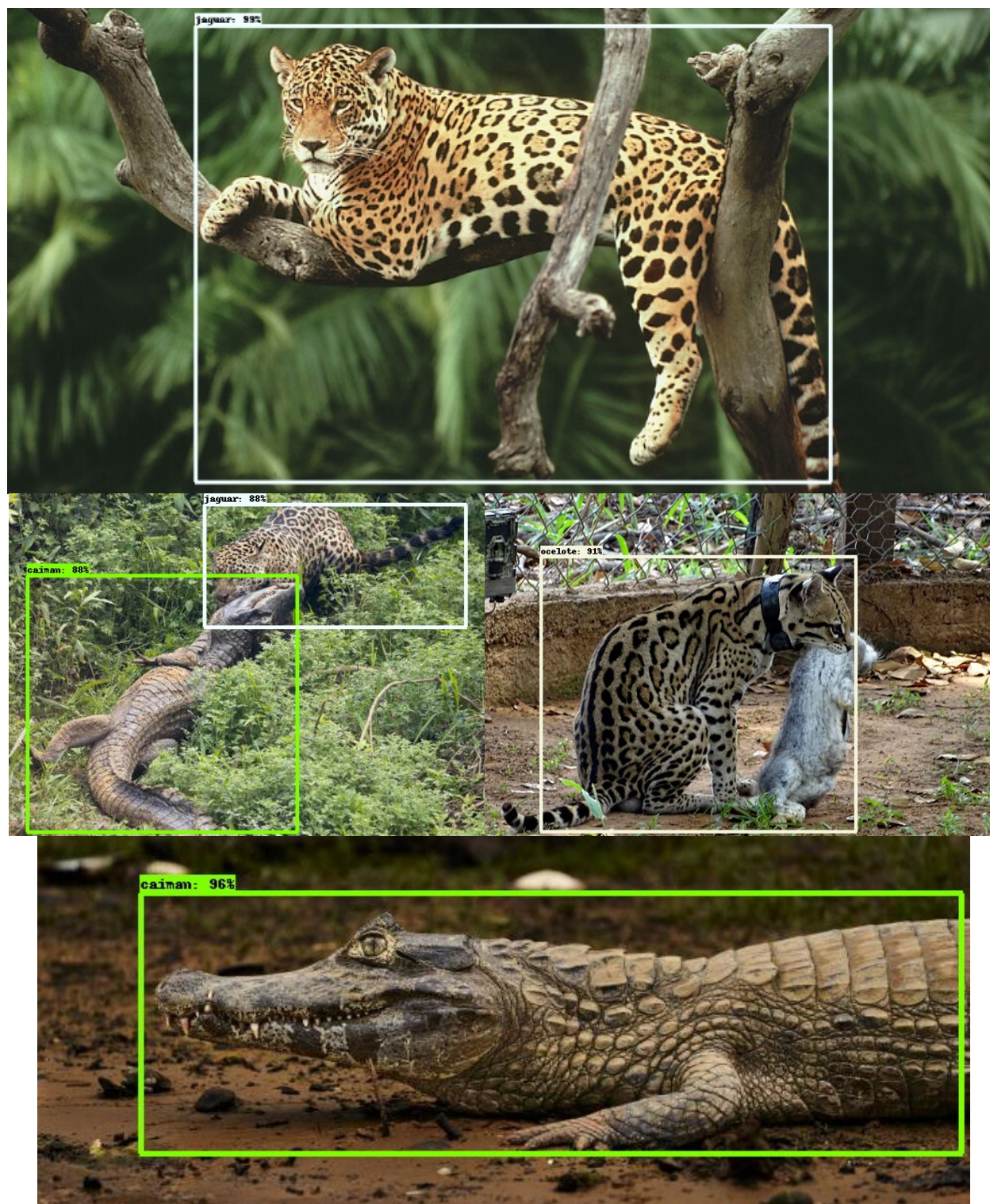
Ilustración 10. Detección realizada por el modelo reentrenado.

## V. RESULTADOS

Para la evaluación de los resultados, se extraerán diferentes imágenes obtenidas ya con la inferencia generada por el modelo y se evaluará desde una perspectiva ligera, cuál fue la precisión del modelo con la cantidad de imágenes que se usaron para entrenamiento, validación y testeo. Además de ello, se ilustrarán los casos de confusión, focos nulos o muertos y escenas donde el modelo presenta los mayores fallos.

*Clases bien detectadas.* Para considerar que la detección estuvo bien realizada, se tomó en cuenta un criterio de que la precisión de la inferencia fuera mayor que el 80%, por tal motivo, algunas de las clases detectadas que tuvieron esta magnitud, fueron:

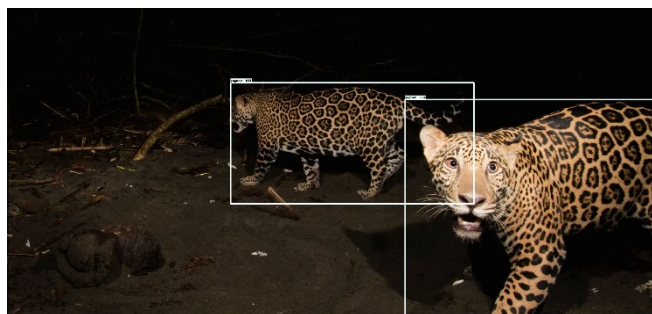




Un caso particular de las detecciones bien realizadas fue la evaluación e interpretación del modelo respecto al animal Jaguar en condiciones no óptimas pues se presenta un foco distorsionado en la imagen, oscuridad y colores un poco más contrastados. Sin embargo, el modelo se ajusta y predice que se tiene en un 91% de precisión. Esto sería un dato muy interesante si se tratase del procesamiento de imágenes en tiempo real de un sensor ubicado en la red sensorial de un

Robot Móvil para monitoreo. Lo anterior debido a que la manipulación de estos datos al ser recolectados por un LiDAR y posteriormente procesados, podrían presentar fallas en no saber realmente que animal se encuentra pues los felinos presentan rasgos similares entre sí.





*Clases regularmente detectadas.* En este caso, se trae a colación ciertas imágenes en las que existe una detección considerablemente buena para una de las clases presentes y para la otra, menor que el 80% en cuanto a magnitud. Por eso se considera que una de las razones podría ser que para los felinos existió mayor enfoque en lo que son sus características físicas y corporales en muchas imágenes, siendo este aspecto algo que no ocurrió de igual forma ni en la Iguana ni con el Caimán. Sin embargo, se obtuvieron buenos resultados haciendo la salvedad de que podría mejorarse el nivel de inferencia pues lo importante no es que solo indique la Localización y la Clasificación, sino que pueda predecir con un alto nivel de *Accuracy*.

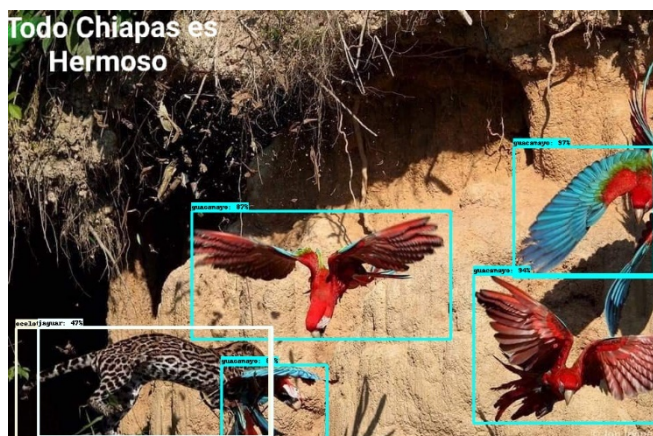


Ilustración 11. Guacamayos con una Detección mayor al 85%. Se detecta Ocelote y Jaguar (Incorrecto).



Ilustración 12. Diferencias entre las magnitudes de predicción.



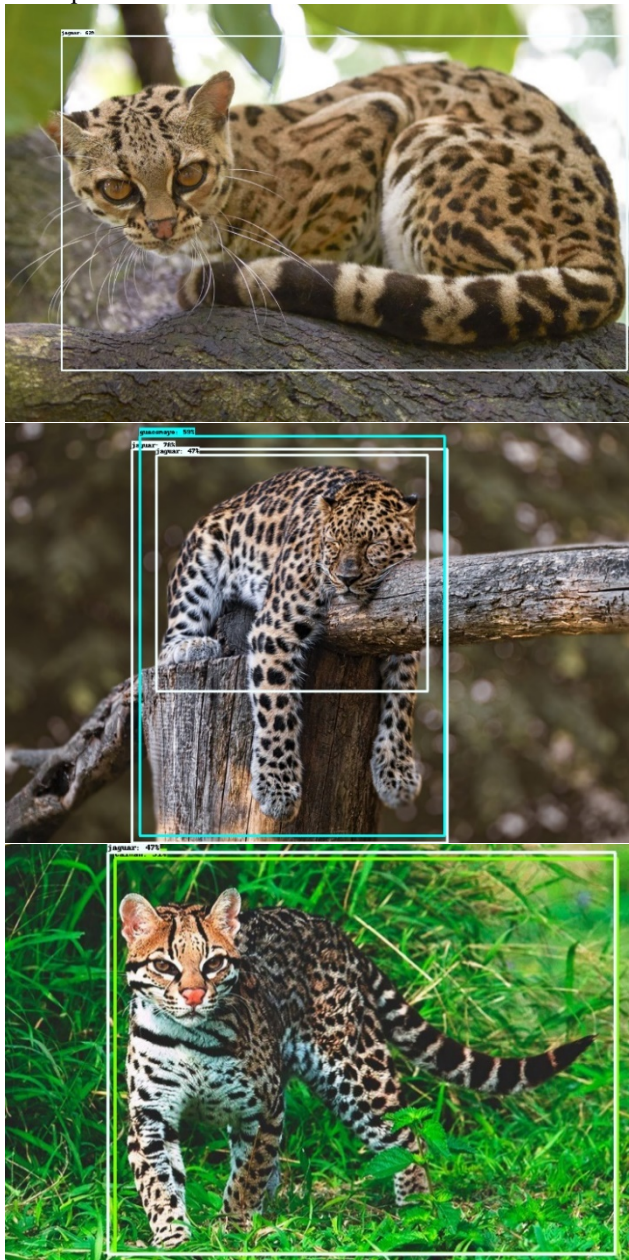
Ilustración 13. Nuevamente se presenta el caso de que detecta Ocelote y Jaguar (Incorrecto).

*Anomalías del Modelo.* En este caso, el equipo de trabajo logra identificar ciertas fallas del modelo que corresponden a tres fenómenos específicos. En la primera imagen se tiene una Reflexión del Jaguar con el agua y esto ocasiona que se detecte como si existieran 3 jaguares en total, aspecto que no es muy viable ni confiable cuando se desee procesar imágenes y tomar decisiones porque se consideran más elementos de los que no se tienen. En la segunda, se detecta un tallo de un árbol como si fuese un Caimán, aspecto que tampoco es muy amigable con el monitoreo de las redes tróficas porque podría correlacionar el jaguar con el caimán y es absurdo desde lo racional humano que un “jaguar trepe un caimán”.





**Detección Nula.** Se consideran aquellas detecciones en las que el modelo no logra identificar ningún objeto en específico, es decir, cuando se indica la presencia de múltiples objetos no existentes en una detección dada o cuando el modelo se equivoca en su totalidad y etiqueta a un objeto de una clase como perteneciente a otra.



### CONCLUSIÓN

Con la realización del presente documento, el equipo logra adquirir más experticia experimental relacionada con el manejo, manipulación y diseño de redes neuronales artificiales para la solución de problemas asociados a la Detección de Objetos remontados a situaciones particulares de la vida real, siendo en este caso, la forma en la que, mediante el modelo pre-entrenado de una red, se predice la localización y la clasificación de ciertas clases de animales pertenecientes al ecosistema de la Fauna Silvestre con sus

biomas terrestre. Por otro lado, a pesar de que no se logra implementar el modelo solicitado en la guía, correspondiente a los modelos basados en R-CNN, debido a los conflictos entre las versiones de Tensorflow, con la selección y reentrenamiento del modelo SSD MobileNet V1 FPN 640x640, se logra entrenar una red que lograra realizar la detección de objetos, a partir de un data set personalizado.

Principalmente, existieron ciertos factores que se debieron tener en cuenta durante el proceso para que no fuese solo “hacer por hacer” sino que se desarrollara de la forma más correcta. Uno de ellos, fue la cantidad de imágenes por cada clase, esto se hizo pensando en que animales darles una relevancia dentro de la red trófica y ver cuál podría ser su correlación tanto de presa como depredador. A pesar de no pensarse como un factor determinante, poder tener un dataset enriquecido y heterogéneo por cada clase, permite que se evalúen todas las condiciones posibles del animal en su hábitat y sea más cómodo poderlo detectar. Esto logró evidenciarse en que, para la mayoría de los casos, la detección de los animales en la validación fue satisfactoria debido a que se realizó con exactitud la detección del animal específico, que aunque que la magnitud de la precisión no fue del 100% se alcanzan valores mayores al 80%, logrando diferenciar entre cada clase de animal indicado en el data set. Así mismo, es importante resaltar la dificultad presentada en el proceso de clasificación, al realizar la diferenciación entre animales tan semejantes como lo son los felinos, debido especialmente a las características de apariencia que comparten entre ellos. Todo lo anterior amplió el panorama del problema y eso, a su vez, destacó más elementos y escenas esenciales dentro de las redes tróficas para que los datos en bruto del sistema de monitoreo puedan ser más robustos. Comparar diferentes modelos para un uso en específico, ayudó a conocer las capacidades que tiene cada uno de ellos, como sus ventajas y desventajas, además de los aspectos en los que se destaca cada modelo teniendo presente su tamaño y velocidad de procesamiento. Con el tamaño se puede realizar diferentes aplicaciones en sistemas embebidos si este no es computacionalmente costoso; asimismo, si es muy rápido puede que la precisión disminuya, por lo que analizar estos campos es de importancia al implementarlos en el problema.

Por tal motivo, fue inspirador poder realizar todo el entregable y adquirir conocimientos relacionados sobre cómo abordar problemas de Detección desde el Deep Learning. Los resultados muestran la eficacia de la aplicación en la problemática planteada porque en una imagen, el modelo es capaz de detectar diferentes animales y clasificarlos, por lo que el modelo puede desarrollar tareas complejas como la detección de múltiples animales en una misma imagen, lo que se remota o extrapola a que se evalúen dinámicas tróficas dentro de una red o hacerle un seguimiento a ciertas especies cuando se expongan a límites humanos o de desequilibrio biológico dentro de su hábitat. Todo lo anterior es de suma importancia pues hoy en día ya existen aplicaciones realizando un monitoreo robusto de las especies y con estos trabajos prácticos, se adquieren conocimientos al campo proporcionando así más herramientas al educando para enriquecer su perfil y ser aún más competente en el mercado.

## REFERENCIAS

- [1] “Salvar animales con Inteligencia Artificial | Telcel Empresas”. <https://www.telcel.com/empresas/tendencias/notas/inteligencia-artificial-salva-animales-peligro-de-extincion.html#> (consultado oct. 15, 2022).
- [2] “Rangers Use Artificial Intelligence to Fight Poachers”. <https://www.nationalgeographic.com/animals/article/paws-artificial-intelligence-fights-poaching-ranger-patrols-wildlife-conservation> (consultado oct. 15, 2022).
- [3] “pandas · PyPI”. <https://pypi.org/project/pandas/> (consultado sep. 04, 2022).
- [4] “numpy · PyPI”. <https://pypi.org/project/numpy/> (consultado sep. 04, 2022).
- [5] “json — Codificador y decodificador JSON — documentación de Python - 3.10.8”. <https://docs.python.org/es/3/library/json.html> (consultado oct. 19, 2022).
- [6] “shutil — Operaciones de archivos de alto nivel — documentación de Python - 3.10.8”. <https://docs.python.org/es/3/library/shutil.html> (consultado oct. 19, 2022).
- [7] “matplotlib · PyPI”. <https://pypi.org/project/matplotlib/> (consultado sep. 04, 2022).
- [8] “¿Tensorflow Records? Qué son y cómo usarlos | por Thomas Gamauf | Principalmente IA | Medio”. <https://medium.com/mostly-ai/tensorflow-records-what-they-are-and-how-to-use-them-c46bc4bbb564> (consultado oct. 19, 2022).
- [9] “Deep Learning por Jesús Alfonso López Sotelo - 9789587786866 - Libros Técnicos Universitarios”. <https://www.alpha-editorial.com/Papel/9789587786866/Deep+Learning> (consultado oct. 20, 2022).
- [10] “models/tf2\_detection\_zoo.md at master · tensorflow/models · GitHub”. [https://github.com/tensorflow/models/blob/master/research/object\\_detection/g3doc/tf2\\_detection\\_zoo.md](https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md) (consultado oct. 16, 2022).
- [11] “COCO - Common Objects in Context”. <https://cocodataset.org/#home> (consultado oct. 16, 2022).
- [12] “(590) Que es R-CNN, Fast R-CNN, Faster R-CNN y Mask R-CNN - Explicación en español - YouTube”. <https://www.youtube.com/watch?v=C-kBqPIZGo8> (consultado oct. 18, 2022).
- [13] “Detección de objetos mediante Fast R-CNN - Cognitive Toolkit - CNTK | Microsoft Learn”. <https://learn.microsoft.com/es-es/cognitive-toolkit/object-detection-using-faster-r-cnn> (consultado oct. 18, 2022).
- [14] “Residual Network (ResNet)”. <https://iq.opengenus.org/resnet/> (consultado oct. 18, 2022).
- [15] “Understanding ResNet50 architecture”. <https://iq.opengenus.org/resnet50-architecture/> (consultado oct. 18, 2022).
- [16] “ResNet-101 convolutional neural network - MATLAB resnet101”. <https://www.mathworks.com/help/deeplearning/ref/resnet101.html?jsessionid=337e04a0a16dec2d11c6e5c43a96> (consultado oct. 18, 2022).
- [17] “¿Qué es el Transfer Learning?”. <https://datascientest.com/es/que-es-el-transfer-learning> (consultado oct. 16, 2022).
- [18] “Mean Average Precision (mAP) Explained: Everything You Need to Know”. <https://www.v7labs.com/blog/mean-average-precision#h1> (consultado oct. 16, 2022).
- [19] “Review: MobileNetV2 — Light Weight Model (Image Classification) | by Sik-Ho Tsang | Towards Data Science”. <https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c> (consultado oct. 18, 2022).
- [20] “Fauna Silvestre - Secretaría Distrital de Ambiente”. <https://www.ambientebogota.gov.co/fauna-silvestre> (consultado oct. 15, 2022).
- [21] “Conozca algunos de los delitos ambientales y sus penas en Colombia - Medio Ambiente - Vida - EL TIEMPO.COM”. <https://www.eltiempo.com/vida/medio-ambiente/conozca-algunos-de-los-delitos-ambientales-y-sus-penas-en-colombia-643364> (consultado oct. 15, 2022).
- [22] “Tráfico ilegal de especies en Colombia - Fundación ProAves - por las aves y su hábitat en Colombia”. <https://proaves.org/trafico-ilegal-de-especies-en-colombia/> (consultado oct. 15, 2022).
- [23] “Tráfico ilegal - Secretaria Distrital de Ambiente”. <https://www.ambientebogota.gov.co/trafico-ilegal> (consultado oct. 15, 2022).
- [24] “Estas son las 12 especies animales amenazadas en Colombia | WWF”. [https://www.wwf.org.co/?300414/122Despecies2Dmas2Damenazadas2DColombia&ads\\_cmpid=17330292229&ads\\_adid=139945893511&ads\\_matchtype=b&ads\\_network=g&ads\\_create=600210239224&utm\\_term=especies%20animales&ads\\_targetid=kwd-297116747730&utm\\_campaign=&utm\\_source=adwords&utm\\_medium=ppc&ttv=2&gclid=Cj0KCQjw166aBhDEARIsAMEyZh5VSIou-67GLCcWzbmB\\_6Jd6pll5SY7uXZukpYTjNBtN2Mz1UPtsUcaArIcEALw\\_wcB](https://www.wwf.org.co/?300414/122Despecies2Dmas2Damenazadas2DColombia&ads_cmpid=17330292229&ads_adid=139945893511&ads_matchtype=b&ads_network=g&ads_create=600210239224&utm_term=especies%20animales&ads_targetid=kwd-297116747730&utm_campaign=&utm_source=adwords&utm_medium=ppc&ttv=2&gclid=Cj0KCQjw166aBhDEARIsAMEyZh5VSIou-67GLCcWzbmB_6Jd6pll5SY7uXZukpYTjNBtN2Mz1UPtsUcaArIcEALw_wcB) (consultado oct. 15, 2022).
- [25] “Fundación ProAves - por las aves y su hábitat en Colombia”. <https://proaves.org/las-guacamayas-de-colombia/> (consultado oct. 15, 2022).
- [26] “Llamado a cuidar las iguanas”. <https://www.cali.gov.co/dagma/publicaciones/152858/llamado-a-cuidar-las-iguanas/> (consultado oct. 15, 2022).
- [27] “Colombia: las comunidades rurales y su papel clave para la conservación del puma”. <https://es.mongabay.com/2017/02/colombia-las-comunidades-rurales-papel-clave-la-conservacion-del-puma/> (consultado oct. 15, 2022).
- [28] “La plataforma de desarrollo de visión artificial Roboflow recauda \$ 20 millones - Noticias Movil”. <https://noticiasmoviles.com/la-plataforma-de-desarrollo-de-vision-artificial-roboflow-recauda-20-millones/> (consultado oct. 18, 2022).