UNIVERSIDAD DEL AZUAY

FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
ESCUELA DE INGENIERÍA DE SISTEMAS Y TELEMÁTICA

Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con smartphone Android y TensorFlow Lite.

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE INGENIERO DE SISTEMAS Y TELEMÁTICA

Autor: Wilson Andrés Campoverde Quito

Director: PhD. Juan Gabriel Barros Gavilanes

Cuenca, Julio, 2019

ECUADOR

DEDICATORIA

A mis padres y abuelos quienes siempre me guiaron y alentaron para que siguiera adelante, a mis hermanos y amigos que me acompañaron en todo momento.

AGRADECIMIENTO

A Dios, a mis padres y profesores en especial al Doctor Gabriel Barros por su apoyo incondicional.

ÍNDICE

Índice de tablas

Índice de figuras

RESUMEN

Generalmente, se utilizan personas para conteo manual de vehículos y actores de la movilidad, lo cual resulta costoso. Actualmente, el avance de la tecnología permite utilizar métodos basados en redes neuronales convolucionales a la visión por computadora. El objetivo de este trabajo es conocer el rendimiento de las técnicas actuales en un dispositivo *smartphone* Android, estas medidas son conocidas en la literatura como *precision* y *recall*. Consecuentemente se analiza la creación de un sistema automático a bajo costo que permita clasificar y contar estos actores del espacio público, usando TensorFlow Lite. Adicionalmente, se reentrena un modelo basado en *Single Shot Detector* para comparar 2 modelos: el primero por efecto y el segundo re-entrenado. Los resultados reportan un incremento significativo en rendimiento mAP para el modelo, como también mejoran las medidas de *precision* y *recall*.

Analysis of the performance to detect and classify vehicles and pedestrians in a continuous time through an Android Smartphone and TensorFlow Lite.

Abstract

Generally, people are used to manually count vehicles and mobility actors, which is expensive. Currently, the technological advance allows to use methods based on convolutional neural networks for vision by computer. The objective of this work is to determine the performance of current techniques in an Android smartphone device, these measures are known as precision and recall. The creation of an automatic low-cost system that allows the classification and counting of these elements in the public space by using TensorFlow Lite is analyzed. Additionally, a model based on the Single Shot Detector is re-trained to compare 2 models: the first by default and the second by retraining. The results report a significant increase in the mAP performance for this model and an improvement in the precision and recall measures.

Gabriel Barros
Thesis Director

Andrés Campoverde
Author

UNIVERSIDAD DEL AZUAY
Dpto. Idiomas

Translated by
Ing. Paúl Arpi

Capítulo I

INTRODUCCIÓN

Los datos de movilidad urbana son importantes para resolver problemas como: congestión de tráfico, espacios inadecuados para bicicletas, peatones y la falta de opciones de transporte público urbano. Existen datos que indican que el espacio de las veredas está sobrepoblado y además son inseguras debido a su tamaño inadecuado [1]. Contar con datos de movilidad es importante para la investigación, como la medición de ruido [2], simular flujo vehicular [3], conteo de personas [4], análisis de impacto del uso de bicicleta [5]. La falta de datos sobre movilidad en la ciudad de Cuenca es evidente, existen datos publicados, pero el periodo desde que los datos son capturados hasta que son publicados es muy prolongado, alrededor de seis meses. Sin embargo, la recolección de los datos manualmente es costoso, extenso en el tiempo; por otro lado, para la automatización de estos procesos se necesita hardware y software específicos a un precio muy elevado. La sociedad consumista cada año genera desperdicios tecnológicos. ¿Podrá un *smartphone* servir como sensor ya, que cuenta con cámara y antenas de comunicación? En la literatura no se encuentra variedad

sobre la recolección de datos urbanos con celulares, usando TensorFlow. Existe un sistema para detectar, seguir y clasificar vehículos el cual está implementada sobre una computadora [6], y otro con un Raspberry Pi [7]. Se ha medido el rendimiento de redes neuronales convolucionales en dispositivos *smartphone* [8] y se compara en varios *smartphone* con redes neuronales para clasificación determinando que *hardware* tiene mejor tiempo de inferencia [9], se midió la velocidad y precisión de tres meta arquitecturas que detectan objetos [10]. Algo que no se ha realizado todavía, es usar TensorFlow Lite en un *smartphone* para clasificar vehículos, bicicletas, motocicletas y peatones como contribución a bases de datos. Contar con estos datos a precio bajo y que directamente los datos estén disponibles para realizar investigaciones sobre la movilidad urbana tendrá mucho impacto sobre la calidad de vida de las personas en una ciudad. Si se obtienen las mediciones y monitoreo para la ciudad de Cuenca, será replicable a ciudades en el mundo. Se propone como objetivo para este trabajo de titulación conocer el rendimiento de estas técnicas con respecto a las métricas de: *precision* y *recall*, tiempo de inferencia, media de precisión promedio (mAP) y promedio logarítmico de tasa de pérdida (*log-average miss rate*) en diferentes escenarios.

1.1 Objetivos

1.1.1 Objetivo general

Medir y analizar en términos de precisión y rendimiento al menos dos modelos usando TensorFlow Lite en un *smartphone* para clasificar: vehículos, bicicletas,

motocicletas y pedestres.

1.1.2 Específicos

1. Revisar la literatura: modelos de redes neuronales profundas que tengan un mejor desempeño en *smartphones*, conocer si existen técnicas de rastreo de objetos específicas para *smartphones*, indagar sobre métricas para evaluar el rendimiento.

2. Seleccionar métodos compatibles con los *smartphones* Android usando TensorFlow Lite.

3. Preparar y evaluar los modelos de aprendizaje profundo seleccionados, documentar y comparar los resultados en términos de precisión y rendimiento.

1.1.3 Estructura de tesis

El documento tendrá la siguiente estructura: el Capítulo 2 presenta el estado del arte, el cual contiene un contraste de técnicas clásicas usadas para el conteo de actores urbanos con las técnicas actuales, en el Capítulo 3 se detallan los métodos utilizados y los experimentos, Capítulo 4 describe los resultados obtenidos y se genera una discusión y por último la Capítulo 5 contiene la conclusión de este trabajo.

Capítulo II

TRABAJOS RELACIONADOS

2.1 Introducción

Durante el transcurso de los últimos cinco años, se ha visto un incremento importante en el uso de la inteligencia artificial en la vida cotidiana; para aplicaciones como: asistentes inteligentes, desbloqueo facial en dispositivos móviles, como también para la banca en línea. Sistemas recomendadores con dos claros ejemplos en áreas diferentes: el primero está en el área de *retail* y es Amazon, ya que el 35% de sus ingresos provienen gracias a estos sistemas y segundo en el área de entretenimiento es Netflix, ya que el 75% del contenido que consumen los usuarios son recomendados por sus algoritmos [11]. Cámaras con inteligencia artificial en diferentes contextos como el uso para emular funciones de hardware, vigilancia [12], etc. Esto es gracias a la mejora en capacidad de procesamiento en CPUs y la nueva tendencia de procesamiento

en GPUs. Cada año los GPUs tienen un número mayor de núcleos, por lo tanto, son miles de hilos que pueden correr paralelamente. Aplicaciones que usan detección de objetos para mejorar la calidad de vida de los habitantes de la tierra son usadas cada vez más, por ejemplo: para detectar enfermedades de las plantas en los cultivos [12], en la medicina hay varias ramas para las que se puede utilizar por ejemplo la lucha contra el cáncer [13], escanear texto para ser traducido, detectar actores de movilidad urbana; un claro ejemplo es el "Autopilot de Tesla" [14]. Para las aplicaciones que usan detección de objetos y/o clasificación, sus modelos son entrenados en computadoras con componentes significativos. Según el director de *School of AI*, Siraj Raval, quién recomienda hardware para aprendizaje de máquina, ver tabla 2.1. Existen servidores que ofrecen servicios propiamente para entrenar estos modelos como son: Google, Amazon y existen también páginas web que son de ayuda para quién incursiona en aprendizaje de máquina, por ejemplo ofrecen el servicio para entrenar modelos y etiquetar como: Luminoth. La importancia de etiquetar objetos se explica más adelante en la sección 3.2.2.

AÑO	CPU	RAM	GPU	RAM de GPU
2019	Intel Core i9-9980XE Extreme Edition 3.0	16GB DDR4	GeForce RTX 2080 Ti	11GB GDDR6
2018	Intel Core i7-8700K	16GB DDR4	GeForce GTX 1080 Ti	11GB GDDR5X
2017	Intel Core i9-7900X X-series Processor	8GB DDR4	GeForce GTX 1080	8GB GDDR5X

Tabla 2.1: Características recomendadas, basado en [15].

2.2 Definición de sistema

Según los autores [7] y [6], un sistema *Intelligent Transportation Systems* (ITS) debe contener lo siguiente: un detector, un clasificador y un rastreador. Los mismos que se describen en la siguiente lista:

1. Detector: El éxito de este componente es determinar en que posición se encuentra un objeto partiendo de una imagen, en nuestro caso detectar objetos en un *frame* o cuadro proveniente de un video.

2. Clasificador: El objetivo de este componente es predecir que clase es un objeto que se detectó en un *frame*.

3. Rastreador: Trata de determinar los cambios en la posición de los objetos entre dos cuadros subsecuentes de un video grabado.

2.2.1 Medidas

2.2.2 Medidas para el detector

La medida que se usa para evaluar el modelo de detección de objetos es conocida como *Mean Average Precision* (mAP) descrita en [16]. Se recalca que ésta medida es usada para las técnicas actuales de aprendizaje de máquina. Con redes neuronales profundas mas no para las técnicas clásicas que se describen en 2.3.1. Para calcular el mAP se siguen los siguientes 2 pasos: primero, obtener la precisión promedio (AP), la misma se obtiene considerando la intersección sobre unión (IoU) mayor que el 50%, ver ecuación 2.1, para cada clase que

se presenta en su *ground truth*. Para el segundo paso se procede a calcular el promedio de todos los valores de AP para cada clase.

$$IoU = \frac{AreadeSolapamiento}{AreadeUnión} \tag{2.1}$$

También se aplicó *log-average miss rate*, una segunda medida de rendimiento la cual se propone en [17] para comparar dos detectores diferentes, haciendo uso de los falsos positivos por imagen (FFPI) para medir el espacio uniforme en un espacio logarítmico.

2.2.3 Medidas para el clasificador

Para evaluar la precisión del clasificador se usaron las siguientes medidas: *precision* (PRE), *recall* (REC) y una comparación de falsos positivos con verdaderos positivos. Las medidas son definidas en las ecuaciones 2.2, 2.3, respectivamente. En dónde (VP) es verdadero positivo, (FP) significa falso positivo y (FN) corresponde a falso negativo.

$$PRE = \frac{VP}{VP + FP} \tag{2.2}$$

$$REC = \frac{VP}{VP + FN} \tag{2.3}$$

Según lo investigado se dividió en dos secciones, la sección 2.3.1 en dónde se describen las técnicas clásicas de visión por computadora para ITS, se tomaron estos trabajos ya que se usaron como referencia base para la construcción de esta tesis. En la actualidad ya contamos con la tecnología

necesaria para emplear aprendizaje profundo descritas en la sección 2.3.2. Se encontró que no existen trabajos orientados a la movilidad urbana, sin embargo, existen tesis y artículos de clasificación y/o detección, que fueron una guía para el cumplimiento de los objetivos planteados.

2.3 Implementaciones existentes

2.3.1 Técnicas clásicas de aprendizaje de máquina

Los sistemas de detección y clasificación de vehículos fueron desarrollados e implementados en computadoras [6], dispositivos embarcados como Raspberry PI [7]; ambos siguen un flujo muy similar sin hacer uso de aprendizaje profundo. Para la clasificación se usó *Support Vector Machine* (SVM) también conocida como *Support-Vector Networks* publicada en el año de 1995 [18], con técnicas que ayudan a clasificar como son *Measured Based Features* (MBF), como también la técnica *Intensity Pyramid- based Histogram Oriented Gradients* (IPHOG) y para detección de vehículos se usó *Gaussian Mixture Model* (GMM) para substraer el fondo; para el rastreo se usó el filtro de Kalman, ver imágenes 2.1 y 2.2.

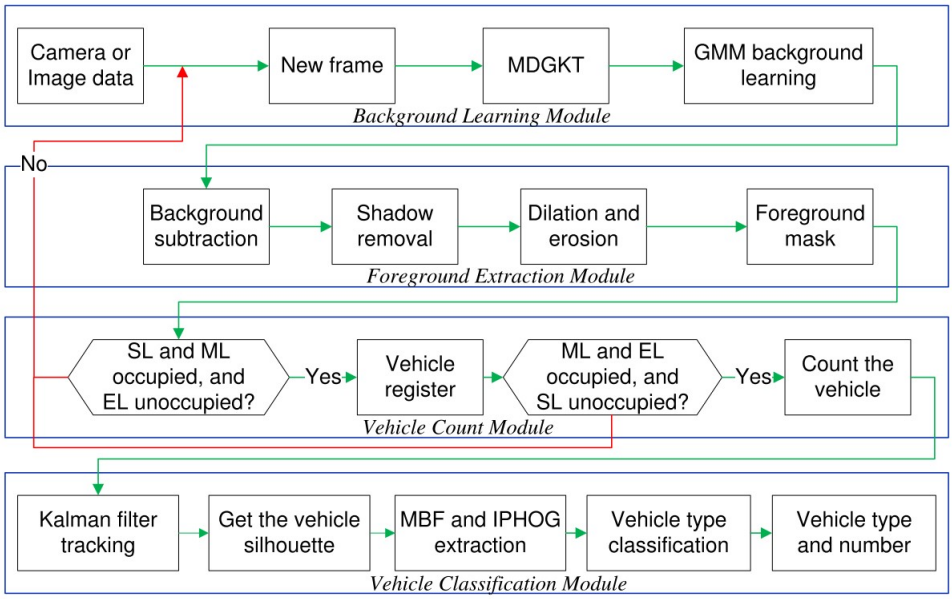


Figura 2.1: Flujo de trabajo con MATLAB, tomado de [6].

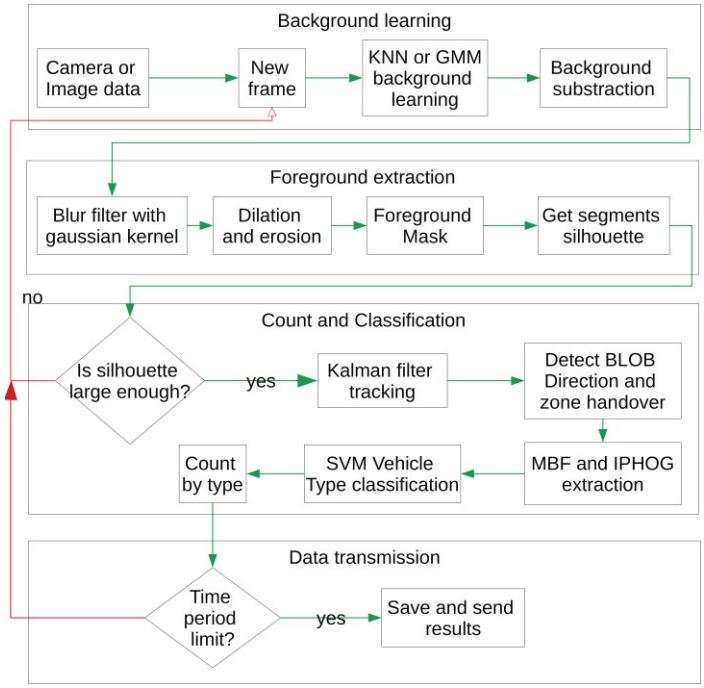


Figura 2.2: Flujo de trabajo con C++, tomado de [7].

Como se puede observar en la tabla 2.2 en la que se detalla el *hardware* usado para estas técnicas clásicas, también se describen las medidas de evaluación a sus trabajos, la forma de evaluar en dichos sistemas y por último el lenguaje de programación que fue utilizado para estos trabajos.

	Hardware	Librerías	Medidas de evaluación	Forma de evaluar	Lenguaje de Programación
[6]	Computadora	OpenCV	PRE / REC Matriz de confusión puntaje F1	Videos propios	Matlab
[7]	Raspberry Pi 3	OpenCV	Conteo manual Conteo automático	Videos propios	C++

Tabla 2.2: Comparación de trabajos clásicos entre [6] y [7].

2.3.2 Técnicas actuales de aprendizaje de máquina

Se encontró, que hay una brecha de investigación para aplicaciones móviles en Android que hagan detección, clasificación y rastreo haciendo uso de aprendizaje profundo con el framework TensorFlow Lite (TF Lite), aplicado a *Intelligent Transportation Systems*.

Existen investigaciones que hacen uso de aprendizaje profundo con TensorFlow Lite en *smartphones* Android. Sin embargo, la gran mayoría realiza solo clasificación. En consecuencia, se tiene una red neuronal que solamente es capaz de clasificar; por lo que únicamente se evalúan medidas usadas en la clasificación de objetos como son: *Precision, Recall* y puntaje F1. Estas evaluaciones se realizan solamente a fotografías. Como se puede observar en la tabla 2.3, se detallan los modelos que se usaron para cada trabajo, es preciso denotar que los tiempos de inferencia en los dos trabajos detallados son para modelos de clasificación. Se espera que los modelos que clasifican y detectan a su vez devuelvan tiempos de inferencia mayores.

Clasificación				
	Hardware	Medida de evaluación	Modelo	Tiempo de inferencia
[8]	Nvidia Jetson TX2 Nokia 8	Throughput (fps) Tiempo inferencia (ms)	MobileNet V1	56ms
[9]	10.000 distintos *smartphones* (OnePlus 6)	Tiempo inferencia (ms)	MobileNet V1	24ms

Tabla 2.3: Comparación de implementaciones en *smartphones*.

Se encontró una tesis de pre-grado en donde se realizó *Transfer Learning* [19], hacia tres modelos diferentes *Single Shot Detection* (SSD) [20], *Faster Recurrent Convolutional Neural Network* (Faster R-CNN) [21] y por último *Tiny You Only Look Once* (Tiny YOLO) [22]. Estos modelos re-entrenados se convirtieron a TF Lite para que el *hardware* del dispositivo sea mejor aprovechado. El *smartphone* Android que se utilizó para las pruebas fue un OnePlus 5T. Los modelos usados por *Alsing*, sirven para clasificar y detectar un objeto POST-IT.

Los resultados que Alsing [19] quiso evidenciar se detallan en la figura 2.3, en la primera columna se describe el modelo utilizado, en la segunda columna se detalla el tiempo de inferencia en micro-segundos y por último en la tercera columna se ha detallado los resultados de la medida mAP para cada modelo utilizado.

Model	Inference (ms)	mAP
Tiny Yolo V2	515	87.57%
SSD MobileNet V1	454	91.16%
SSD MobileNet V2	438	91.90%
SSD Inception V2	716	96.82%
Faster RCNN Inception V2	4105	96.69%
Faster RCNN ResNet50	20018	99.33%

Figura 2.3: Tabla de tiempos de inferencia y mAP, tomada de [19].

Capítulo III

MÉTODOS Y EXPERIMENTOS

3.1 Definición del sistema

Como se indicó en el capítulo II, este trabajo define e implementa un sistema detector y clasificador basado en video para un dispositivo con sistema operativo Android. Por esta razón, se planteó un flujo de trabajo que invoclucre, el aprendizaje profundo haciendo uso del framework TensorFlow por ser una herramienta disponible para los dispositvos objetivo. Adicionalmente, entrenar un modelo utilizando la red neuronal SSD con Mobilenet en su versión número 2 y TensorFlow Lite permite aprovechar los recursos del *smartphone* Android. El objetivo del reentrenamiento es mejorar el rendimiento final de la aplicación.

En la figura 3.1 se presenta un diagrama de bloques con las partes principales del sistema. A continuación se describen cada una de las secciones.

La sección de la **cámara** hace referencia al hardware del dispositivo. Cada

frame o cuadro del video capturado en tiempo continuo es la entrada para el sistema planteado. En este caso se requiere examinar el software o paquetes de software disponibles en el dispositivo que permitan la lectura del video y que sean compatibles con nuestro sistema.

Siguiendo con el flujo planteado sigue la sección **procesador de frames**, la cual prepara o re-dimensiona el *frame* a una medida específica para cada imagen, en este caso 300x300 *pixeles*. Este detalle es importante por que cada modelo solo recibe como entrada un *frame* con medidas específicas.

Después de haber re-dimensionado el *frame*, este pasa a la sección de **detector**. La tarea de este componente es detectar la posición de objetos. La salida del detector es el *bounding box* o rectángulo envolvente con coordenadas en la imagen. Este rectángulo puede ser un objeto en formato JSON y contiene 4 coordenadas, las mismas que indican en donde está ubicado el objeto detectado en el *frame* procesado.

La sección **clasificador** permite hacer una estimación de la clase de cada uno de los objetos detectados. Los valores de salida del clasificador son los siguientes: el *label* o la etiqueta del objeto detectado, la precisión del objeto detectado. El valor de precisión o pertenencia a una clase es generalmente representado con un valor de probabilidad entre 0 y 1.

Por último con los resultados obtenidos por el detector y el clasificador, pasan al **rastreador**. Se una la implementación llamada *Object MultiBox Tracker* una aproximación a [23], desarrollado por @*andrewharp* de Google. Los objetos detectados en cada *frame* son asignados con un color diferente por ejemplo: un carro con color azul y un bus con color verde, si se detecta otro

carro se asignará un color diferente rojo por ejemplo, de esa manera se sabe que son el objeto carro pero son dos carros diferentes.

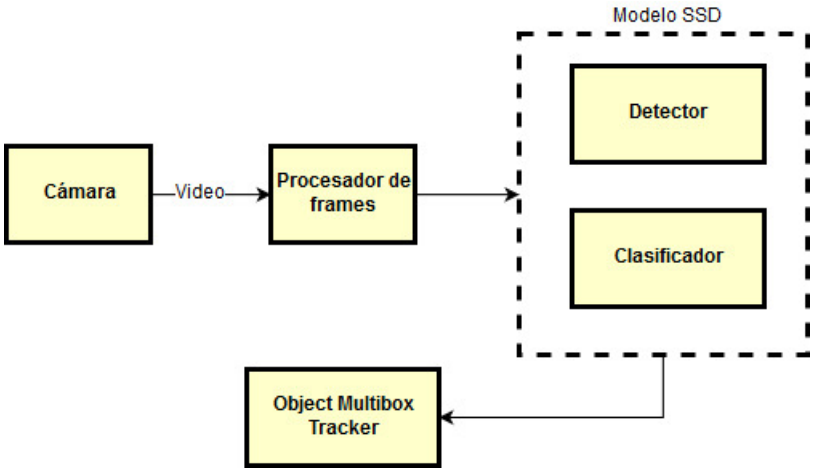


Figura 3.1: Flujo de trabajo utilizado.

Como se aprecia en la figura 3.1, se usa la red neuronal profunda denominada Modelo SSD para los pasos de detección y clasificación. En este caso, se comparan dos modelos SSD: el primero es un modelo entrenado por defecto. Además, es accesible para descargar en internet, solamente se requiere el nombre, versión y *dataset* usado para el entrenamiento para identificarlo sin equivocación. El modelo entrenado de red neuronal que se considera por defecto es: *coco ssd mobilenet v1 quantized*. Se trata de un modelo lanzado el 2018-06-29. El segundo es un modelo entrenado en el proceso de este trabajo, al que se refiere como modelo de red neuronal re-entrenado. Se utiliza como base el modelo *coco ssd mobilenet v2 quantized*, lanzado el 2019-01-03. El entrenamiento permite comparar si el tiempo de inferencia varía dependiendo del número de clases del clasificador. Siendo una misma arquitectura como es SSD, el modelo denominado "por defecto" con 80 clases [24] y el modelo denominado "entrenado" con 6 clases. En los siguientes párrafos, se detallan

los pasos usados para realizar el entrenamiento de la red neuronal usada para detectar y clasificar objetos.

3.2 Entrenamiento

3.2.1 Contexto

Se propuso investigar la respuesta a las siguientes interrogantes, ¿Cómo se entrena una red neuronal? ¿Con el hardware que dispongo es suficiente para entrenamiento? ¿Cuánto tiempo dura entrenar un modelo de detección de objetos? Se procedió a re-entrenar la red neuronal con las siguientes clases: carro, persona, bicicleta, moto, bus, camión. Para esto se hizo uso de las siguientes especificaciones en cuanto a *hardware* se refiere:

Hardware

- 8th Generation Intel® Core™ i7-8750H.
- 16GB DDR4.
- NVIDIA GeForce GTX 1060 with Max-Q Design 6GB.

En cuanto a lo que a *software* se refiere, se hizo la prueba en dos sistemas operativos: Windows 10 y Ubuntu 16.04. El primero presentó conflictos con las versiones de las librerías y dependencias necesarias para el entrenamiento. Por ese motivo se usó Linux. A continuación el software que se usó para el entrenamiento.

Software

- Ubuntu 16.04.
- Python 3.
- TensorFlow 1.5.
- CUDA Toolkit 9.0.
- cuDNN 7.1.4.
- ProtoBuf 2.4.1.
- Bazel 0.27.0.
- TensorFlow Lite Converter (TOCO).

3.2.2 Etiquetado

Para el etiquetado de las imágenes se utilizó el programa *LabelImg* [25], este es el proceso más tedioso y repetitivo del entrenamiento. Para cada *frame* se etiquetó el máximo de objetos en la imágen; ver figura 3.2 y 3.3. Es también necesario tener en cuenta que la salida del etiquetado tiene que estar en formato (*PASCAL - VOC*), el cuál es un formato XML propiamente para cualquier modelo SSD ó *Faster RCNN*. En cambio para los modelos YOLO se usa su propio formato o formato de YOLO. En este caso, la salida es un archivo .txt o de texto.

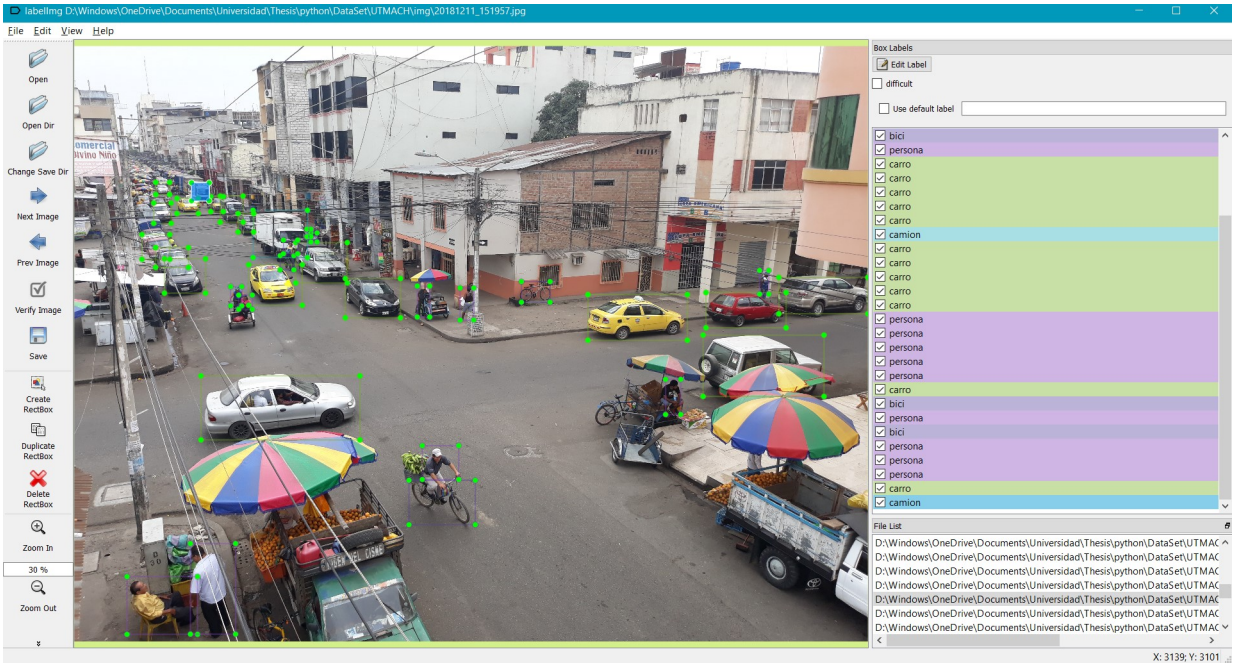


Figura 3.2: Frame etiquetado de Machala.

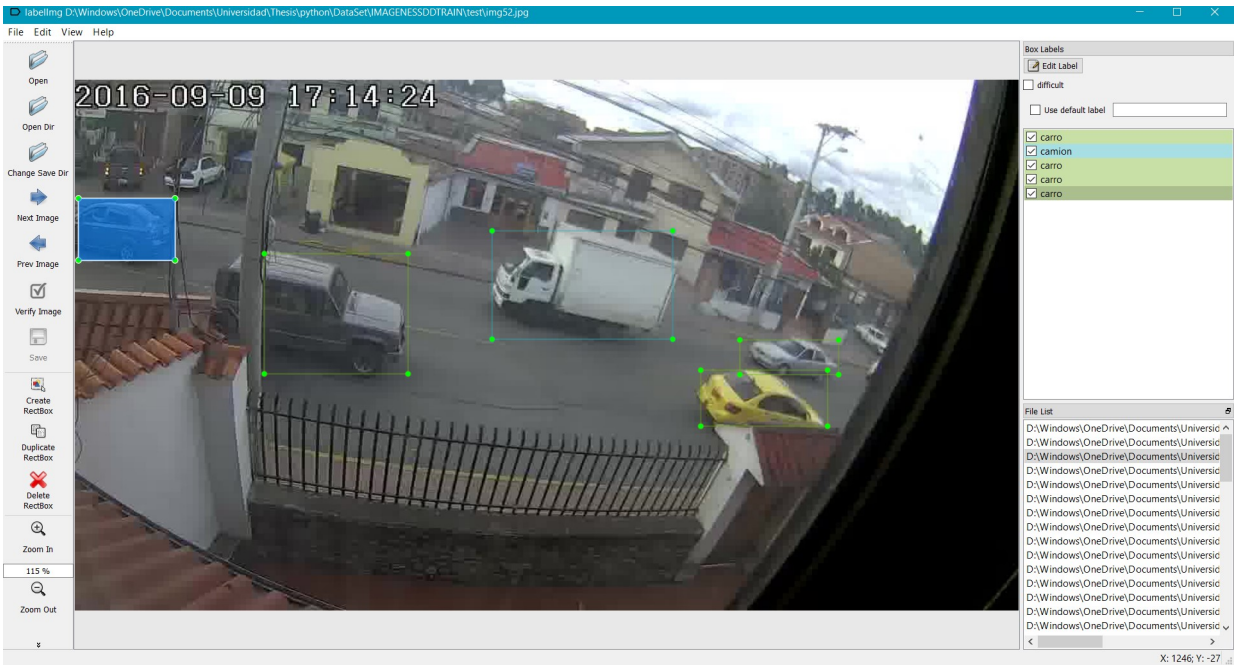


Figura 3.3: Frame etiquetado de Cuenca.

3.2.3 Procedimiento

Se realizó la instalación de todos los componentes de *software* detallados en 3.2.1, para esto se siguió 2 guías. El trabajo [26], que es una guía específica para Windows 10 y el trabajo [27] una guía para las dos plataformas Windows y Ubuntu; ambas guías contienen instructivos de como entrenar una red neuronal profunda para clasificación y detección de objetos. Se continuó con la segmentación de las imágenes dando como resultado: 2156 imágenes para entrenamiento, 268 imágenes para *test* y 59 imágenes para validación en el día y 22 imágenes para validación en la noche. Para poder entrenar una red neuronal profunda de detección de objetos es necesario, que estas imágenes estén en un archivo CSV para que se generen los *TF Records*, tanto para las imágenes de entrenamiento como para las de *test*.

Una vez que se obtuvo los *TF Records*, como penúltimo paso para entrenar se tiene que configurar el número de clases de salida de nuestra red neuronal y el *batch size* que soporte la tarjeta de video (GPU). Se modificó el archivo *.config* del modelo a re-entrenar mencionado en 3.1. Finalmente se tiene que generar el archivo *labelmap.pbtxt*, el cual contiene el nombre del objeto y el id o identificador por ejemplo: `item {id: 1 name: "carro"}`. Por último se ejecutó el comando para entrenar detallado en [26]. En nuestro caso se configuró para que la red neuronal profunda se entrene con un número de 200.000 pasos, lo que tomó al rededor de 60 horas. Para correr la red neuronal en el *smartphone* se tiene que transformar el modelo a TF Lite, para que sea utilizado de una manera eficiente por el dispositivo, para esto se siguió una guía desarrollada por Google. Lo primero que se tiene que hacer es obtener el

TensorFlow frozen graph por consiguiente se procede a usar Bazel con TOCO detallando las salidas como:

- (TFLite_Detection_PostProcess).
- (TFLite_Detection_PostProcess:1).
- (TFLite_Detection_PostProcess:2).
- (TFLite_Detection_PostProcess:3).

Las mismas que son cuatro arreglos: bounding boxes, clases detectadas, precisión de las clases y el número de detecciones.

3.3 Datasets o conjunto de datos

3.3.1 Videos

Los videos que se usan para los experimentos fueron capturados solamente de la ciudad de Cuenca, videos con ubicación en la calle Don Bosco y videos que tienen la ubicación en la calle 24 de Mayo, Universidad del Azuay. Para los videos mencionados se tiene el conteo para cada actor de la movilidad urbana.

3.3.2 Imágenes

Las imágenes recolectadas tuvieron varios propósitos, el primero fue entrenar y hacer el *test* en la red neuronal, el segundo propósito fue obtener los valores reales *(ground truth)*, ver figura 3.4, para realizar los experimentos detallados en

la sección 3.4.1. Estas imágenes provienen de dos ciudades Cuenca y Machala, ya que en estos dos lugares se tuvo acceso a cámaras de video.

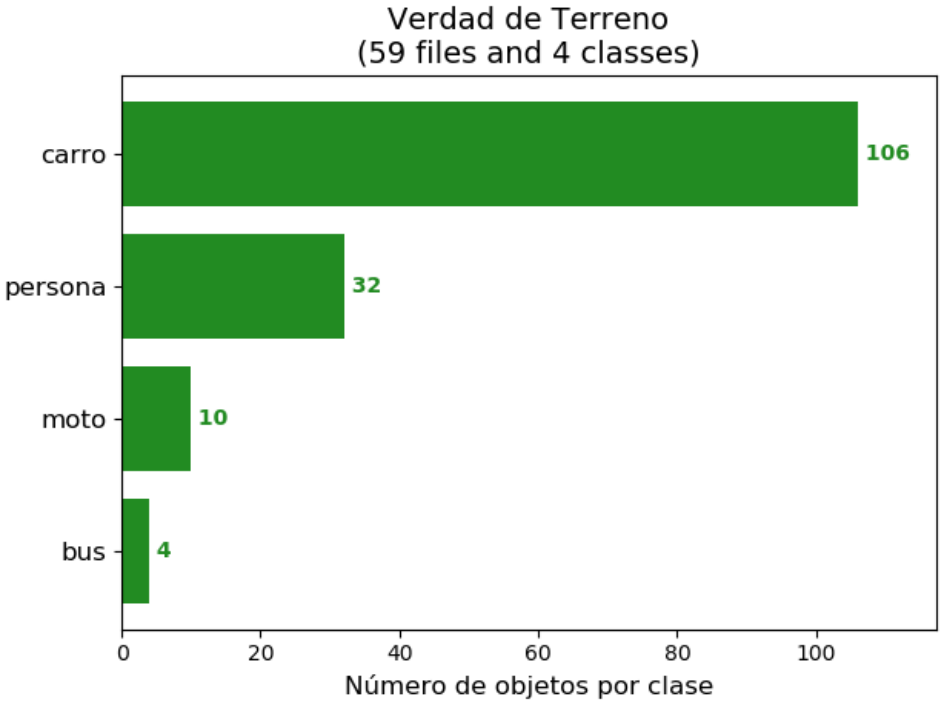


Figura 3.4: Número de objetos y clases, *ground truth* para el día.

3.4 Definición de experimentos

Se probaron los modelos con el *dataset* de videos. El principal obstáculo para leer estos videos desde el almacenamiento del *smartphone* fue no encontrar librerías nativas de Java para Android que lean un video *frame* por *frame*. Se realizaron varios experimentos para solventar esta deficiencia. Por ejemplo, se usó la librería más popular en visión por computadora *Open Computer Vision* (Open CV) para Android; sin embargo, la librería devolvió un error que los únicos formatos que puede leer desde el almacenamiento son los formatos correspondientes a imágenes. También se probó FFmpeg para Android sin

obtener los resultados requeridos. Por lo tanto, la solución fue utilizar una aplicación que lea una imagen e infiera sobre ella, así para el número de *frames* recolectados. A continuación, se explica los dos tipos de experimentos que se realizaron, el primero, con inferencia sobre imágenes con Flutter [28] y la segunda sección sobre videos en tiempo continuo con TF Lite [29].

3.4.1 Experimento sobre imágenes

El experimento que permitió adquirir las medidas de precisión y rendimiento consistió en obtener un número de *frames* del *dataset* de videos lo que llamamos *ground truth* en la sección 3.2.3, para esto se utilizó *VLC media player*; este programa da dos alternativas para obtener *frames* de un video: La primera es de una manera gráfica y la segunda es mediante línea de comandos que es la que se utilizó. Para obtener la medida de tiempo de inferencia sobre imágenes se usó un contador que empieza a aumentar su cuenta desde que el *input* va a la red neuronal hasta que sale de la red como se detalla en [19] en la sección **3.8.1**, imprimiendo el tiempo en un *log* de Android. Se usó la aplicación que hace la inferencia sobre imágenes para el modelo por defecto, una vez que se terminó el experimento para dicho modelo se procedió con el segundo modelo el re-entrenado. Se hace énfasis que el mismo experimento se realiza para condiciones con buena luminosidad y con condiciones de baja luminosidad.

3.4.2 Experimento sobre video en tiempo continuo

Se realizaron dos experimentos sobre video, el primero, para determinar el tiempo de inferencia entre TensorFlow Mobile (TF Mobile), una versión para

dispositivos *smartphone* descontinuada en el año 2019 y TF Lite la nueva alternativa desarrollada por Google. Es importante señalar que los modelos usados para este experimento fueron tres arquitecturas SSD, dos son *coco ssd mobilenet v1* con la única diferencia que el uno tiene una extensión `.pb` y el segundo tiene la extensión `.tflite`. Los dos son modelos denominamos por defecto. El tercer modelo es el denominado re-entrenado. Para las pruebas de rastreo se cortó una parte de un video del *dataset*. En donde aparece solo un objeto, un carro y verificar que el color del *boundig box* del objeto efectivamente no cambia. Por lo tanto, es el mismo objeto y que además mientras se mueve el *BoundingBox* también se traslada. Se realizó también una segunda prueba que consistió en tener mas objetos en el video en donde hay diferentes objetos de una misma clase y objetos de diferentes clases con casos de solapamiento entre objetos.

Capítulo IV

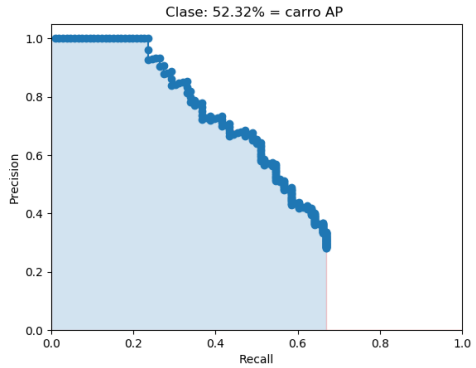
RESULTADOS Y DISCUSIÓN

4.1 Imágenes

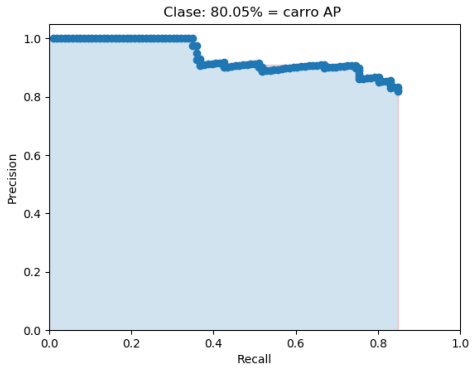
4.1.1 Precisión y rendimiento en el día

Se realiza una comparación de PRE, REC y AP para cada clase detallada en la figura 3.4. No se reportan resultados para la clase bicicleta por falta de datos en el *ground truth*. Para las figuras de los resultados se tiene *precision* en el eje Y, *recall* en el eje X y cada uno de los puntos en azul es el número de detecciones, el área bajo la curva se define por el promedio de precisión.

Como se observa en la figura 4.1, la gráfica (a) con unos resultados muy inferiores a los de la gráfica (b), en la cual resultó que el AP para el modelo re-entrenado es de 80.05%.



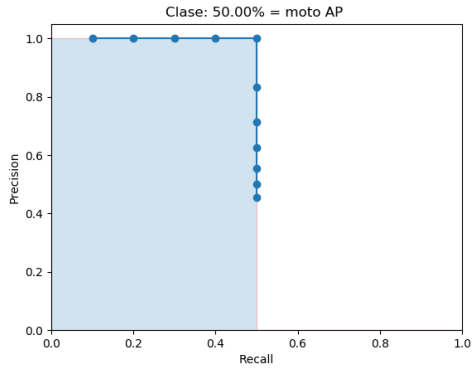
(a) *Precision* y *Recall* para la clase carro modelo por defecto.



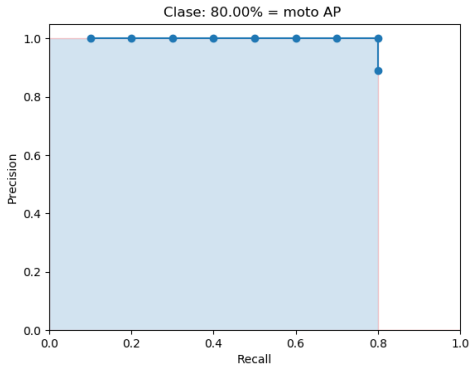
(b) *Precision* y *Recall* para la clase carro modelo re-entrenado.

Figura 4.1: Comparación de PRE y REC para la clase carro en el día.

En la figura 4.2 se observa que la clase por defecto o gráfica (a) tiene un número de detecciones que en la medida de *precision* están por debajo del 0.5 y en cambio en la gráfica (b) el *recall* es 0.8.



(a) *Precision* y *Recall* para la clase moto modelo por defecto.

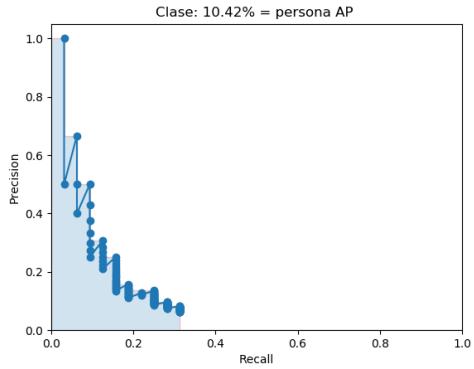


(b) *Precision* y *Recall* para la clase moto modelo re-entrenado.

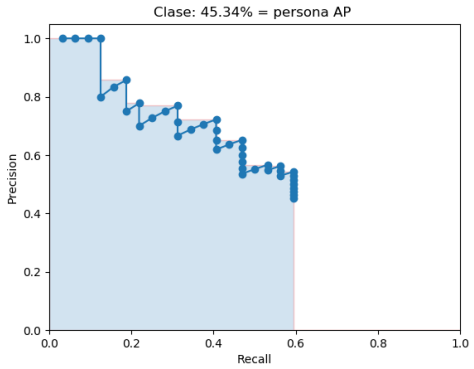
Figura 4.2: Comparación de PRE y REC para la clase moto en el día.

En la siguiente figura 4.3, se compara para la clase persona. Es muy notorio que el modelo por defecto, o gráfica (a), tiene una *precision* por debajo de 0.4. En los experimentos ya se tenía una indicios del resultado, pues la red neuronal detectaba como persona a árboles, tubos y más. Por esta razón, se entrenó la

red neuronal con personas que están de costado para mejorar la medida PRE y REC, obteniendo valores de *recall* de hasta el 60 % o 0.6.



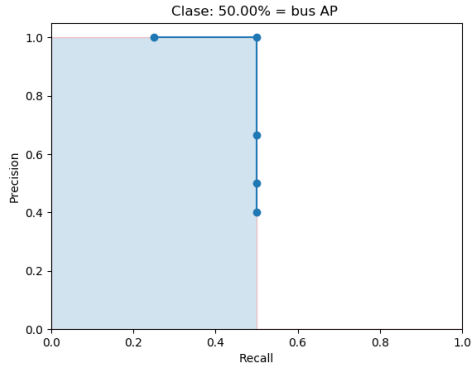
(a) *Precision* y *Recall* para la clase persona modelo por defecto.



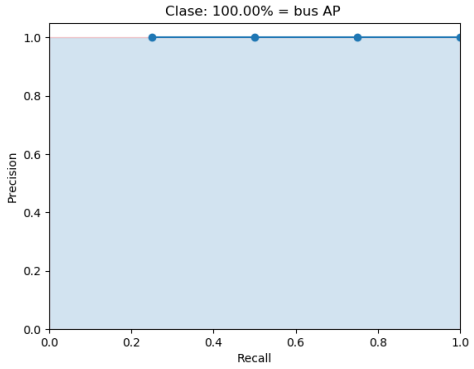
(b) *Precision* y *Recall* para la clase persona modelo re-entrenado.

Figura 4.3: Comparación de PRE y REC para la clase persona en el día.

Para la figura 4.4, en la gráfica (b), se puede observar un AP de 100%. Esto se debe al entrenamiento incluyendo buses de la ciudad de Cuenca y dando como resultado un PRE y REC de 1. Sin embargo, el número de casos analizados es reducido.



(a) *Precision* y *Recall* para la clase bus modelo por defecto.

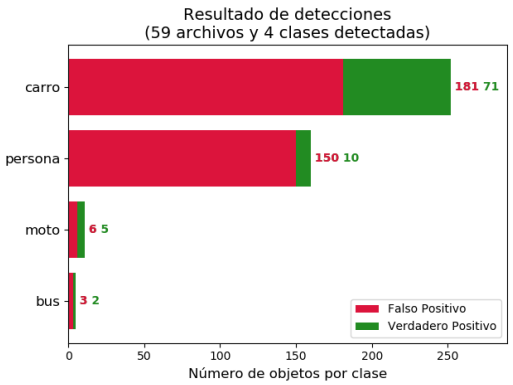


(b) *Precision* y *Recall* para la clase bus modelo re-entrenado.

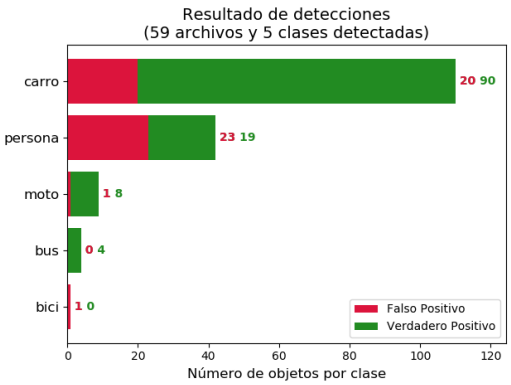
Figura 4.4: Comparación de PRE y REC para la clase bus en el día.

Un resultado importante es, que el número de FP para las 4 clases bajó

considerablemente con el modelo re-entrenado como se observa en la figura 4.5. Atención, las figuras no tienen el mismo rango en el eje x.



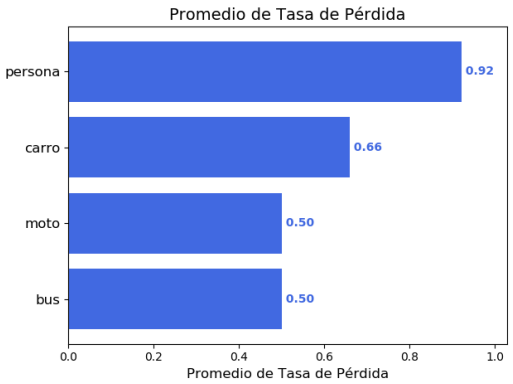
(a) Número de detecciones para modelo por defecto.



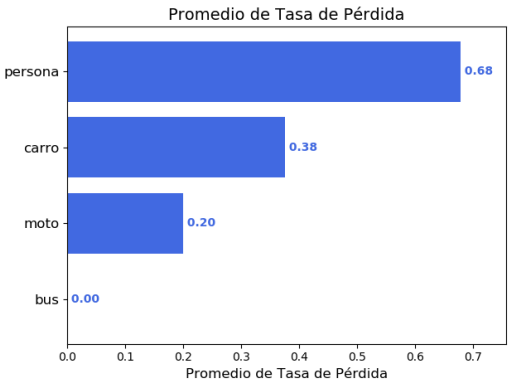
(b) Número de detecciones para modelo re-entrenado.

Figura 4.5: Comparación entre FP y VP detectados en el día.

Es muy importante tener en cuenta la medida de promedio logarítmico de tasa de pérdida, con esta se puede determinar en cuanto está fallando por cada clase la red neuronal. Como se puede observar en la figura 4.6 gráfica (b) los valores se redujeron notoriamente con el modelo re-entrenado.



(a) Promedio de tasa de pérdida para modelo por defecto.

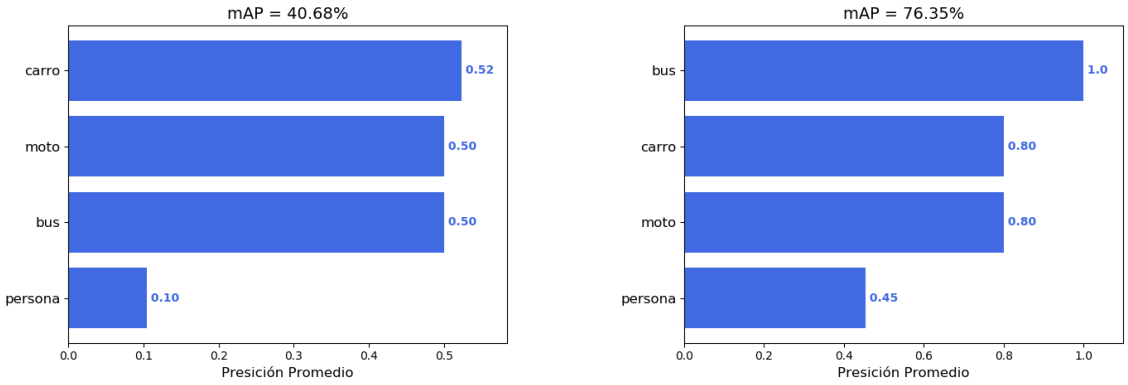


(b) Promedio de tasa de pérdida para modelo re-entrenado.

Figura 4.6: Comparación de promedio de tasa de pérdida en el día.

Por último en los resultados para precisión y rendimiento, se hace la comparación del mAP obtenido para el *ground truth*. Se discute sobre el por

qué una red neuronal entrenada con un *dataset* muy reconocido como: *Common Objects in Context* (COCO), tiene un mAP por debajo del 50% en el entorno objetivo que es la ciudad de Cuenca. Se puede observar en la figura 4.7 la gráfica (b) muestra un resultado considerablemente mejor que la gráfica (a).



(a) mAP para modelo por defecto.

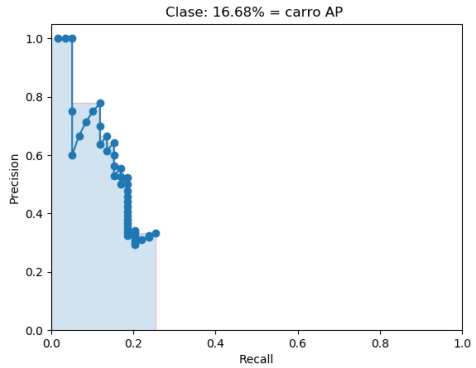
(b) mAP para modelo re-entrenado.

Figura 4.7: Comparación de mAP en el día.

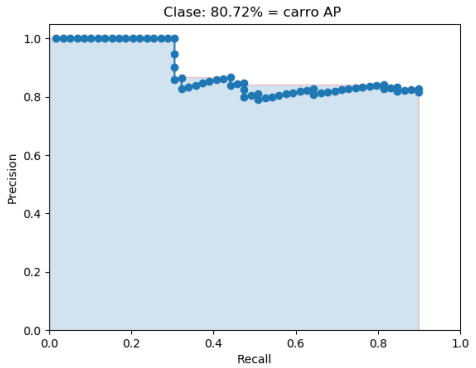
4.1.2 Precisión y rendimiento en la noche

Se realizó pruebas para comparar la precisión y el rendimiento en condiciones de poca luz, en la noche. Solamente se realizaron para 22 imágenes y con un *ground truth* con un número de tres clases: 59 carros, 4 personas y 2 buses. Se hace hincapié que para el entrenamiento no se usaron imágenes en condiciones de poca luz.

En la siguiente figura 4.8 se puede observar que las medidas de PRE, REC para la clase carro en el modelo re-entrenado son superiores a los resultados del modelo por defecto.



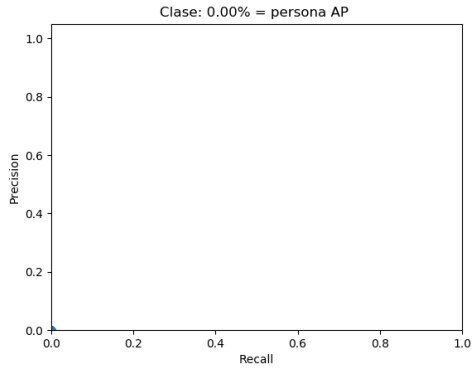
(a) *Precision* y *Recall* para la clase carro modelo por defecto.



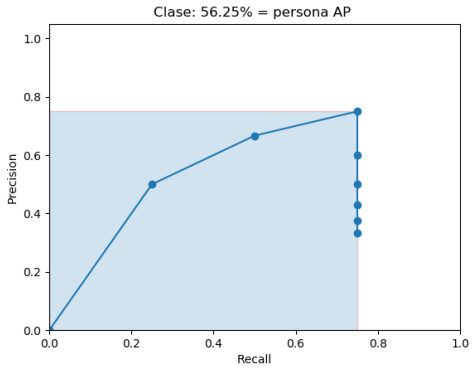
(b) *Precision* y *Recall* para la clase carro modelo re-entrenado.

Figura 4.8: Comparación de PRE y REC para la clase carro en la noche.

En la figura 4.9 en la que se compara la clase persona es muy notorio que el modelo por defecto o gráfica (a), no detectó ningún objeto perteneciente a la clase persona en comparación a la gráfica (b), sin embargo, las medidas para el modelo re-entrenado se deberían mejorar entrenando la red neuronal profunda con imágenes en poca luz.



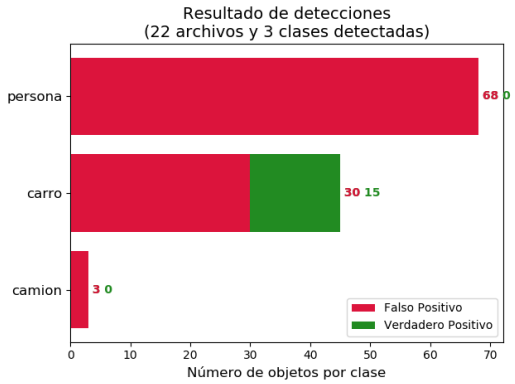
(a) *Precision* y *Recall* para la clase persona modelo por defecto.



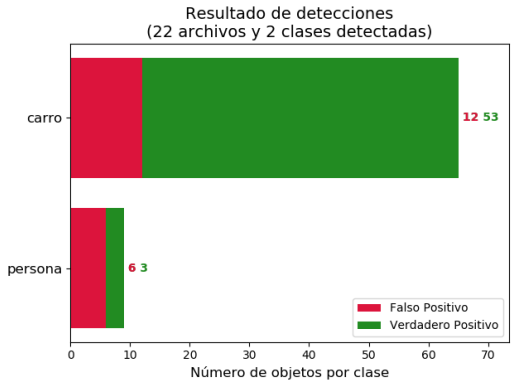
(b) *Precision* y *Recall* para la clase persona modelo re-entrenado.

Figura 4.9: Comparación de PRE y REC para la clase persona en la noche.

La figura 4.10, reporta que ninguno de los dos modelos detectó objetos de la clase bus. Por lo tanto, de igual manera se debería hacer un etiquetado de más buses pero agregando escenarios en condiciones de noche.



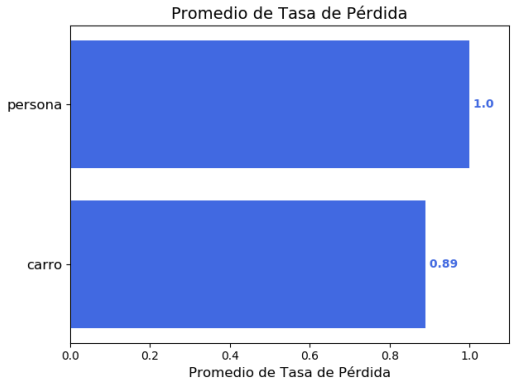
(a) Número de detecciones para modelo por defecto.



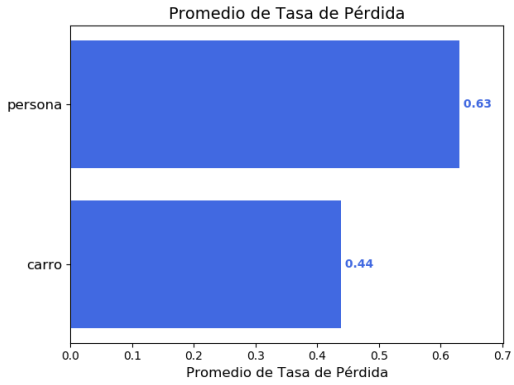
(b) Número de detecciones para modelo re-entrenado.

Figura 4.10: Comparación entre FP y VP detectados en la noche.

En la figura 4.11 gráfica (a) los valores del modelo por defecto son notoriamente mayores, indicando que tiene un mayor promedio de tasa de pérdida para las dos clases.



(a) Promedio de tasa de pérdida para modelo por defecto.



(b) Promedio de tasa de pérdida para modelo re-entrenado.

Figura 4.11: Comparación de promedio de tasa de pérdida en la noche.

4.1.3 Tiempo de inferencia

Es importante reportar que el tiempo de inferencia para el modelo por defecto como se menciona en 3.1 con un número de 80 clases tenga una menor tiempo de inferencia que el modelo re-entrenado con un número menor de clases.

Los tiempos que se reportan en la figura 4.12 tienen una diferencia de 34ms considerando el punto mínimo y el máximo. Comparando entre los restultados de las 2 versiones de modelos SSD señalados en la figura 2.3 de la sección 2.3.2, los resultados que se obtuvieron son más prometedores aún teniendo un mayor número de clases que [19].

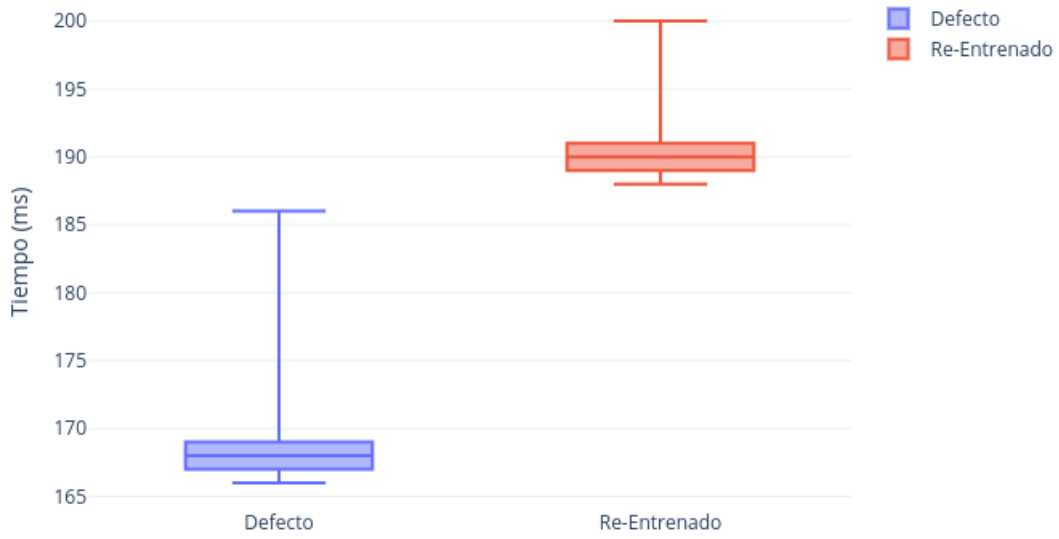


Figura 4.12: *BoxPlot* para comparar tiempos de inferencia sobre imágenes.

4.2 Videos

4.2.1 Tiempo de inferencia

El sistema de conteo debe ser capaz de manejar la transmisión de video en vivo. Sin embargo, las bibliotecas bien conocidas como OpenCV y FFMpeg no siempre están disponibles para ser utilizadas en el middleware de Android. Como resultado, no es posible realizar un perfil más detallado de los tiempos de inferencia de nuestra aplicación. No obstante, la figura 4.13 presenta una

comparación de los tiempos de inferencia entre TensorFlow Mobile y los otros dos modelos ya comparados utilizando TensorFlow Lite. Las imágenes se seleccionaron al azar mientras se procesaba un flujo de video desde la cámara. Las mejoras en el uso de TF Lite son claras, 120 ms frente a casi 900 ms por imagen.

Figura 4.13: Gráfico de líneas para comparar tiempos de inferencia sobre video.

Capítulo V

CONCLUSIONES

5.1 Conclusiones

Se presenta un acercamiento a la implementación de un sistema contador automático aplicado a la movilidad urbana. Es una novedad el uso de aprendizaje profundo basado en redes neuronales convolucionales sobre un *smartphone* "reciclado" para el problema clasificación y detección de objetos. Fué posible re-entrenar un modelo SSD para mejorar las medidas de *precision* y *recall* que inicialmente se obtuvieron en el modelo por defecto. El aumento en mAP se debe a, que el conjunto de datos usados el entrenamiento se obtuvo utilizando datos del entorno objetivo y una disminución en el número de clases. Los resultados de los tiempos de inferencia se obtuvieron tanto de video como de imagen, y son bastante rápidos. Esto permite tener una aplicación móvil con un seguimiento fluido en comparación con la aplicación móvil que usa TF mobile. Sin embargo, en promedio, el modelo predeterminado es 20 ms más rápido pero con menos precisión que el modelo re-entrenado. Haciendo

un contraste de los resultados de tiempo de inferencia del trabajo [19], con los resultados obtenidos se concluye que con un mayor número de clases los tiempos de inferencia son menores. Los trabajos futuros incluyen la evaluación de algoritmos de seguimiento, así como la inclusión de diferentes posiciones y ángulos de la cámara, haciendo uso de *data augmentation*.

REFERENCIAS

[1] Miguel Arévalo. “El 38% de aceras no tiene el tamaño adecuado”. In: *El Tiempo* (2018). URL: https://www.eltiempo.com.ec/noticias/cuenca/2/aceras-cuenca-azuay.

[2] Felipe A. Torres and Gabriel Barros. “Sound noise monitoring platform: Smart-phones as sensors”. In: *European Wireless 2017 - 23rd European Wireless Conference* (2017).

[3] Walter Verdugo-romero, Luis Gonz, Erwin J Sacoto-cabrera, and Alexandra Verdugo-cabrera. “Mathematical and statistical analysis for the simulation of the vehicular flow in a specific sector of the city of Cuenca-Ecuador”. In: (2018), pp. 1–4.

[4] Gabriel Barros Gavilanes and Don Bosco. “Persons counter through Wi-Fi ’ s passive sniffing for IoT”. In: (2018).

[5] Stanislav Sobolevsky, Ekaterina Levitskaya, Henry Chan, and Marc Postle. “Impact Of Bike Sharing In New York City Future Cities Catapult , London , UK * Correspondence to be addressed to : sobolevsky@nyu.edu”. In: (2016), pp. 1–26.

[6] Zezhi Chen, Tim Ellis, and Sergio A Velastin Smieee. “Vehicle Detection, Tracking and Classification in Urban Traffic”. In: (2012).

[7] Felipe Torres, Gabriel Barros, and Maria Jose Barros. “Computer vision classifier and platform for automatic counting: More than cars”. In: *2017 IEEE 2nd Ecuador Technical Chapters Meeting, ETCM 2017* 2017-Janua (2018), pp. 1–6. DOI: 10.1109/ETCM.2017.8247454.

[8] Matti Siekkinen. “Latency and Throughput Characterization of Convolutional Neural Networks for Mobile Computer Vision”. In: *CoRR* ABS/1803.0 (2018). arXiv: 1803.09492.

[9] Andrey Ignatov, Radu Timofte, Przemyslaw Szczepaniak, William Chou, Ke Wang, Max Wu, Tim Hartley, and Luc Van Gool. “AI Benchmark: Running Deep Neural Networks on Android Smartphones”. In: (2018). arXiv: 1810.01109. URL: http://arxiv.org/abs/1810.01109.

[10] Chen Sun and Kevin Murphy. “Speed/accuracy trade-offs for modern convolutional object detectors”. In: (2016). arXiv: arXiv:1611.10012v3.

[11] Steve MacKenzie, Ian and Meyer, Chris and Noble. “How Retailers can keep up with Consumers”. In: *McKinsey & Company* October (2013), pp. 1–10.

[12] Farhad Manjoo. “The Sublime and Scary Future of Cameras With A.I. Brains”. In: *The New York Times* (2018). URL: https://nyti.ms/2t3i5oN.

[13] Adam Yala, Constance Lehman, Tal Schuster, Tally Portnoi, and Regina Barzilay. “A Deep Learning Mammography-based Model for Improved Breast Cancer Risk Prediction”. In: *Radiology* 02139 (2019), p. 182716. ISSN: 0033-8419. DOI: 10.1148/radiol.2019182716.

[14] Shantanu Ingle and Madhuri Phute. “Tesla Autopilot : Semi Autonomous Driving, an Uptick for Future Autonomy”. In: *International Research Journal of Engineering and Technology* (2016), pp. 2395–56. ISSN: 2395-0072.

[15] Siraj Raval. *Best Laptop for Machine Learning*. theschool.ai. 2018. URL: https://www.youtube.com/watch?v=dtFZrFKMiPI.

[16] You-chi Cheng. “A MAXIMAL FIGURE-OF-MERIT LEARNING APPROACH TO MAXIMIZING MEAN AVERAGE PRECISION WITH DEEP NEURAL NETWORK BASED CLASSIFIERS Kehuang Li † Zhen Huang Chin-Hui Lee School of Electrical and Computer Engineering”. In: (2014), pp. 4536–4540.

[17] Piotr Dollár, Christian Wojek, Bernt Schiele, and Pietro Perona. “Pedestrian detection: An evaluation of the state of the art”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.4 (2012), pp. 743–761. ISSN: 01628828. DOI: 10.1109/TPAMI.2011.155.

[18] N.H. Farhat. “Photonic neural networks and learning machines”. In: *IEEE Expert* 7.5 (2002), pp. 63–72. ISSN: 0885-9000. DOI: 10.1109/64.163674.

[19] Oscar Alsing. “Mobile Object Detection using TensorFlow Lite and Transfer Learning”. In: (2018).

[20] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng Yang Fu, and Alexander C. Berg. “SSD: Single shot multibox detector”. In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9905 LNCS (2016), pp. 21–37. ISSN: 16113349. DOI: 10.1007/978-3-319-46448-0_2. arXiv: 1512.02325.

[21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.6 (2017), pp. 1137–1149. ISSN: 01628828. DOI: 10.1109/TPAMI.2016.2577031. arXiv: arXiv:1506.01497v3.

[22] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. “You Only Look Once: Unified, Real-Time Object Detection”. In: (2015). ISSN: 01689002. DOI: 10.1109/CVPR.2016.91. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[23] Valtteri Takala and Matti Pietikäinen. “Multi-object tracking using color, texture and motion”. In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (2007). ISSN: 10636919. DOI: 10.1109/CVPR.2007.383506.

[24] Equipo de TensorFlow. *Object Detection*. tensorflow. 2019. URL: https://www.tensorflow.org/lite/models/object_detection/overview.

[25] 52 Contribuidores. *Labelimg*. MIT. 2019. URL: https://github.com/tzutalin/labelImg.

[26] EdjeElectronics. *How To Train an Object Detection Classifier for Multiple Objects Using TensorFlow (GPU) on Windows 10*. EdjeElectronics. 2018. URL: https://github.com/EdjeElectronics/TensorFlow-Object-Detection-API-Tutorial-Train-Multiple-Objects-Windows-10.

[27] pythonprogramming. *Training Custom Object Detector - Tensorflow Object Detection API Tutorial*. pythonprogramming. 2017. URL: https://pythonprogramming.net/introduction-use-tensorflow-object-detection-api-tutorial/.

[28] Qian Sha. *Flutter Tflite*. Google. 2018. URL: https://pub.dev/packages/tflite.

[29] TensorFlow Lite. *Android TensorFlow Lite*. Google. 2019. URL: https://www.tensorflow.org/lite/models/object_detection/overview.

UNIVERSIDAD DEL AZUAY

Doctora María Elena Ramírez Aguilar, Secretaria de la Facultad de Ciencias de la Administración de la Universidad del Azuay

C E R T I F I C A:

Que, el Consejo de Facultad en sesión del 23 de enero de 2019, conoció y aprobó la solicitud para realización del trabajo de titulación, presentada por:

Estudiante:	Wilson Andrés Campoverde Quito (cód. 70003)
Tema:	**"Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con smartphone Android y Tensorflow Lite".** Previo a la obtención del título de Ingeniero de Sistemas y Telemática
Director:	Ing. Juan Gabriel Barros Gavilanez
Tribunal:	Ing. Marcos Orellana Cordero e Ing. Pablo Pintado Zumba.

Plazo de presentación del trabajo de titulación: Se fijó como plazo para la entrega del trabajo de titulación, conforme a la Disposición Tercera del Reglamento de Régimen Académico, un período académico contado desde la fecha de aprobación del diseño del trabajo, esto es hasta el 23 de julio de 2019.

E INFORMA:

Que, en aplicación de la Disposición General Cuarta del Reglamento de Régimen Académico vigente, en caso de que el estudiante no culmine y apruebe el trabajo de titulación luego de dos períodos académicos contados a partir de su fecha de culminación de estudios, deberá realizar la actualización de conocimientos previa a su titulación.

Cuenca, 24 de enero de 2019

Dra. María Elena Ramírez Aguilar
Secretaria de la Facultad de Ciencias de la Administración

UNIVERSIDAD DEL AZUAY
Facultad de Ciencias de la Administración
SECRETARIA

CONVOCATORIA

Por disposición de la Junta Académica de la escuela de Ingeniería de Sistemas y Telemática se convoca a los Miembros del Tribunal Examinador, a la sustentación del Protocolo del Trabajo de Titulación: **Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con Smartphone Android y Tensorflow Lite,** presentado por el estudiante Wilson Andrés Campoverde Quito con código 70003, previa a la obtención del título de Ingeniero de Sistemas y Telemática, para el día **Viernes, 18 de enero de 2019 a las 09:40**

Tomar en cuenta que posterior a la sustentación del Diseño del Trabajo de Titulación, por ningún concepto se puede realizar modificaciones ni cambios en los documentos; únicamente, en caso de diseño aprobado con modificación, el Director adjuntará al esquema un oficio indicando que se procede con los cambios sugeridos.

Cuenca, 16 de enero de 2019

Dra. María Elena Ramírez Aguilar
Secretaria de la Facultad

Ing. Gabriel Barros Gavilanes .....

Ing. Marcos Orellana Cordero .....

Ing. Pablo Pintado Zumba .....

Oficio Nro. 004-2019-DIST-UDA

Cuenca, 4 de enero de 2019

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De nuestras consideraciones,

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 4 de enero del 2019, revisó la documentación del trabajo de titulación denominado **"ANÁLISIS DEL RENDIMIENTO DE DETECTAR, CLASIFICAR VEHÍCULOS Y PEDESTRES EN TIEMPO CONTINUO CON SMARTPHONE ANDROID Y TENSORFLOW LITE"**, por la/el estudiante **WILSON ANDRÉS CAMPOVERDE QUITO,** con código estudiantil 70003, estudiante de la Escuela de Ingeniería de Sistemas y Telemática, y revisado por **GABRIEL BARROS**, previo a la obtención del título de Ingeniero de Sistemas y Telemática.

La Junta Académica considera que la documentación cumple con las normas legales y reglamentarias de la Universidad y de la Facultad de Ciencias de la Administración y designa como miembros del tribunal a MARCOS ORELLANA y PABLO PINTADO, así por su digno intermedio, el conocimiento y aprobación por parte del Consejo de Facultad.

Atentamente,

Marcos Orellana Cordero
Coordinador de la Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

UNIVERSIDAD DEL AZUAY

ACTA

SUSTENTACIÓN DE PROTOCOLO/DENUNCIA DEL TRABAJO DE TITULACIÓN

Fecha de sustentación: **Viernes, 18 de enero de 2019 a las 09:40**

1.1. Nombre del estudiante: Wilson Andrés Campoverde Quito
1.2. Código: 70003
1.3. Director sugerido: Ing. Gabriel Barros Gavilanes
1.4. Codirector (opcional): _____
1.4.1. Tribunal: Ing. Marcos Orellana Cordero e Ing. Pablo Pintado Zumba
1.4.2. Título propuesto: **Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con Smartphone Android y Tensorflow Lite**

1.4.3. Aceptado sin modificaciones : _____

1.4.4. Aceptado con las siguientes modificaciones:

- OBSERVACIÓN EXPLICITA A EVITAR LA PARTE CONTINUA
- CAMBIAR PRECISIÓN X SELECCIÓN DE LA PARTE DE CLIMATOLOGÍA

1.4.5. No aceptado
1.4.6. Justificación:

Tribunal

Ing. Gabriel Barros Gavilanes

Ing. Marcos Orellana

Ing. Pablo Pintado Zumba

Sr. Wilson Andrés Campoverde Quito

Dra. María Elena Ramírez Aguilar

RÚBRICA PARA LA EVALUACIÓN DEL PROTOCOLO DE TRABAJO DE TITULACIÓN
(Tribunal)

1.37. Nombre del estudiante: Wilson Andrés Campoverde Quito
1.37.1. Código : 70003
1.37.2. Director sugerido: Ing. Gabriel Barros Gavilanes
1.37.3. Codirector (opcional):
1.37.4. Título propuesto: **Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con Smartphone Android y Tensorflow Lite**
Revisores tribunal: Ing. Marcos Orellana Cordero e Ing. Pablo Pintado Zumba
1.38. Recomendaciones generales de la revisión:

	Cumple	No cumple
Problemática y/o pregunta de investigación		
127. ¿Presenta una descripción precisa y clara?	✓	
128. ¿Tiene relevancia profesional y social?	✓	
Objetivo general		
129. ¿Concuerda con el problema formulado?	✓	
130. ¿Se encuentra redactado en tiempo verbal infinitivo?	✓	
Objetivos específicos		
131. ¿Permiten cumplir con el objetivo general?	✓	
132. ¿Son comprobables cualitativa o cuantitativamente?	✓	
Metodología		
133. ¿Se encuentran disponibles los datos y materiales mencionados?	✓	
134. ¿Las actividades se presentan siguiendo una secuencia lógica?	✓	
135. ¿Las actividades permitirán la consecución de los objetivos específicos planteados?	✓	
136. ¿Las técnicas planteadas están de acuerdo con el tipo de investigación?	✓	
Resultados esperados		
137. ¿Son relevantes para resolver o contribuir con el problema formulado?	✓	
138. ¿Concuerdan con los objetivos específicos?	✓	
139. ¿Se detalla la forma de presentación de los resultados?	✓	
140. ¿Los resultados esperados son consecuencia, en todos los casos, de las actividades mencionadas?	✓	

Ing. Gabriel Barros Gavilanes Ing. Marcos Orellana Ing. Pablo Pintado Zumba

UNIVERSIDAD DEL AZUAY

Facultad de Ciencias de la Administración
Escuela de Ingeniería de Sistemas y Telemática

Cuenca, 21 de enero del 2019

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De mi consideración,

Yo **Juan Gabriel Barros Gavilanes** informo que he revisado los cambios implementados al protocolo del trabajo de titulación previo a la obtención del título de Ingeniero de Sistemas y Telemática, denominado **"ANÁLISIS DEL RENDIMIENTO DE DETECTAR, CLASIFICAR VEHÍCULOS Y PEDESTRES EN TIEMPO CONTINUO CON SMARTPHONE ANDROID Y TENSORFLOW LITE"**, realizado por el estudiante **Wilson Andrés Campoverde Quito,** con código estudiantil 70003. Trabajo que según mi criterio cumple con las modificaciones sugeridas por el Tribunal, por lo que puede continuar su desarrollo planificado.

Sin otro particular, me suscribo

Atentamente

Gabriel Barros

Universidad del Azuay

UNIVERSIDAD DEL AZUAY

DOCTORA MARIA ELENA RAMIREZ AGUILAR, SECRETARIA DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACION DE LA UNIVERSIDAD DEL AZUAY.

CERTIFICA:

Que, el señor **Wilson Andrés Campoverde Quito,** registrado con código 70003, alumno de la carrera de Ingeniería de Sistemas y Telemática, tiene aprobado el 93.43% de créditos de su malla curricular.

Cuenca, 18 de Enero de 2019

Dra. María Elena Ramírez Aguilar
SECRETARIA DE LA FACULTAD
DE CIENCIAS DE LA ADMINISTRACION

UNIVERSIDAD DEL AZUAY
Facultad de Ciencias de la Administración
SECRETARÍA

No. Derecho 0144392
rgp.-

0883672

Cuenca, 04 de enero de 2019

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De mi consideración,

Estimado Señor Decano, yo **Wilson Andrés Campoverde Quito** con C.I. **0106200207**, código estudiantil 70003; estudiante de la Carrera de Ingeniería de Sistemas y Telemática, solicito encarecidamente a usted, la aprobación del protocolo de trabajo de titulación con el tema **"ANÁLISIS DEL RENDIMIENTO DE DETECTAR, CLASIFICAR VEHÍCULOS Y PEDESTRES EN TIEMPO CONTINUO CON SMARTPHONE ANDROID Y TENSORFLOW LITE"** previo a la obtención del título de Ingeniero de Sistemas y Telemática para lo cual adjunto la documentación respectiva.

Por la favorable acogida que brinde a la presente, anticipo mi agradecimiento.

Atentamente

Wilson Campoverde
Estudiante de la Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

0881636

Oficio Nro. 004-2019-DIST-UDA

Cuenca, 4 de enero de 2019

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De nuestras consideraciones,

La Junta Académica de la Escuela de Ingeniería de Sistemas y Telemática, reunida el día 4 de enero del 2019, revisó la documentación del trabajo de titulación denominado **"ANÁLISIS DEL RENDIMIENTO DE DETECTAR, CLASIFICAR VEHÍCULOS Y PEDESTRES EN TIEMPO CONTINUO CON SMARTPHONE ANDROID Y TENSORFLOW LITE"**, por la/el estudiante **WILSON ANDRÉS CAMPOVERDE QUITO**, con código estudiantil 70003, estudiante de la Escuela de Ingeniería de Sistemas y Telemática, y revisado por **GABRIEL BARROS**, previo a la obtención del título de Ingeniero de Sistemas y Telemática.

La Junta Académica considera que la documentación cumple con las normas legales y reglamentarias de la Universidad y de la Facultad de Ciencias de la Administración y designa como miembros del tribunal a MARCOS ORELLANA y PABLO PINTADO, así por su digno intermedio, el conocimiento y aprobación por parte del Consejo de Facultad.

Atentamente,

Marcos Orellana Cordero
Coordinador de la Escuela de Ingeniería de Sistemas y Telemática
Universidad del Azuay

UNIVERSIDAD DEL AZUAY

Facultad de Ciencias de la Administración
Escuela de Ingeniería de Sistemas y Telemática

Cuenca, 04 de enero de 2019

Ingeniero,
Oswaldo Merchán Manzano
DECANO DE LA FACULTAD DE CIENCIAS DE LA ADMINISTRACIÓN
UNIVERSIDAD DEL AZUAY

De mi consideración,

Yo, **Juan Gabriel Barros Gavilanes** informo que he revisado el protocolo de trabajo de titulación, elaborado previo a la obtención del título de Ingeniero de Sistemas y Telemática, **"ANÁLISIS DEL RENDIMIENTO DE DETECTAR, CLASIFICAR VEHÍCULOS Y PEDESTRES EN TIEMPO CONTINUO CON SMARTPHONE ANDROID Y TENSORFLOW LITE"**, realizado por el estudiante **Wilson Andrés Campoverde Quito, con código estudiantil** 70003**,** protocolo que a mi criterio, cumple con los lineamientos y requerimientos establecidos por la carrera.

Por lo expuesto, me permito sugerir que sea considerado para la revisión y sustentación del mismo,

Sin otro particular, me suscribo.

Atentamente

Gabriel Barros

Universidad del Azuay

Universidad del Azuay Ingeniería de Sistemas y Telemática

Metodología de la Investigación

Guía para la presentación de la propuesta de proyecto de titulación

DENUNCIA/PROTOCOLO DE TRABAJO DE TITULACIÓN

1. Datos generales

1.1 Nombre del estudiante: Wilson Andrés Campoverde Quito

1.1.1 Código: 70003

1.1.2 Contacto:
teléfonos: +593988739756
mail: mrrobot@es.uazuay.edu.ec andrescamp_ac@hotmail.com

1.2 Director sugerido: Gabriel Barros

1.2.1 Contacto: gbarrosg@uazuay.edu.ec

1.3 Codirector sugerido: Juan Pablo Pendones

1.3.1 Contacto: jpendones@uzuay.edu.ec

1.4 Asesor metodológico: Daniela Ballari

1.5 Tribunal designado: (de acuerdo con la normativa interna de cada Facultad).

1.6 Aprobación: pendiente

1.7 Línea de investigación de la carrera:

a) Sistemas informáticos.

1.7.1 Código UNESCO:

- 1203 Informática de computadores
- .04 Inteligencia Artificial.
- .11 Software de Computador.

1.7.2 Tipo de trabajo:

a) Investigación

1.8 Área de estudio:

a) Inteligencia Artificial; Aprendizaje de Máquina; Visión por Computadora; Programación en Móviles.

1.9 Título propuesto:

Análisis del rendimiento de detectar, clasificar vehículos y pedestres en tiempo continuo con smartphone Android y TensorFlow Lite.

1.11 Estado del proyecto:

Nuevo.

2. Contenido

2.1 Motivación de la investigación:

Los datos de movilidad urbana son importantes para resolver problemas como: congestión de tráfico, espacios inadecuados para bicicletas y pedestres, la falta de opciones de transporte público urbano. Existen datos que indican que el espacio de las veredas está sobrepoblado y además son inseguras debido a que su tamaño es inadecuado (Arévalo, 2018). Contar con datos de movilidad es importante para la investigación como la medición de ruido (F. A. Torres & Barros, 2017), simular flujo vehicular (Verdugo-romero, Gonz, Sacoto-cabrera, & Verdugo-cabrera, 2018), conteo de personas (F. Torres, Barros, & Barros, 2018), análisis de impacto del uso de bicicleta (Sobolevsky, Levitskaya, Chan, & Postle, 2016). La falta de datos sobre movilidad en la ciudad de Cuenca es evidente, existen datos publicados, pero el periodo desde que los datos son capturados hasta que son publicados es demasiado extenso alrededor de seis meses. Sin embargo, la recolección de los datos manualmente es costoso y extensos en el tiempo; por otro lado, los precios tanto de hardware como de software que se usan en la automatización de estos procesos también son elevados. La sociedad consumista cada año genera desperdicios tecnológicos. ¿Podrá un smartphone servir como sensor debido a su cámara y antenas de comunicación?

2.2 Problemática:

La obtención de datos sobre movilidad urbana es costosa por el hardware y software que se usa. Los algoritmos utilizados en este tipo de aplicaciones en dispositivos embarcados involucran Principal Component Analisis (PCA) y Suport Vector Machine (SVM) (Chen, Ellis, & Smieee, 2012; F. Torres et al., 2018) . En teoría la capacidad de cálculo de estos dispositivos es reducida cuando se compara con los servidores actuales. Por lo tanto, se desea conocer si los algoritmos clasificadores basados en redes neuronales pueden usarse en los dispositivos smartphone para realizar conteos automáticos.

2.3 Resumen:

Generalmente, se utilizan personas para un conteo manual, lo cual resulta costoso. En la ciudad de Cuenca no hay datos centralizados sobre la movilidad urbana que se actualicen constantemente, por lo cual sería conveniente clasificar y contar con un sistema automático a

bajo costo. Se analiza el video usando técnicas actuales de visión por computadora usando un smartphone. El objetivo para este trabajo es conocer el rendimiento de estas técnicas con respecto a métricas bien conocidas en la literatura como (precisión y rendimiento). Esto se realizará usando TensorFlow Lite para que sea capaz de implementarse en un smartphone Android.

2.4 Base conceptual e indagación exploratoria:

En la literatura existe una brecha de investigación sobre la recolección de datos urbanos con smartphones, usando TensorFlow Lite. Existe un sistema para detectar, seguir y clasificar vehículos la cual usa mediante una computadora (Chen et al., 2012), con un Raspberry Pi (F. Torres et al., 2018). Además se han medido el rendimiento de redes neuronales convolucionales en smartphones (Siekkinen, 2018) y se ha hecho una comparación en varios smartphones corriendo redes neuronales para determinar cuál tiene mayor rendimiento (Ignatov et al., 2018). Adicionalmente, se midió la velocidad y precisión de tres meta arquitecturas que detectan objetos (Sun & Murphy, 2016).

Existe también una tesis que tiene por objetivo determinar la viabilidad de detectar un objeto (Post It), usando modelos de aprendizaje profundo con TensorFlow (Abadi et al., 2016) con su nuevo desarrollo TensorFlow Lite. Esta hace uso de tres tipos de modelos como son Single Shot Detector (SSD), Faster Recurrent Convolutional Neural Networks (R-CNN) y Tiny You Only Look Once (Yolo) (Aising, 2018).

2.5 Objetivo General.

Medir y analizar en términos de precisión y rendimiento al menos dos modelos usando TensorFlow Lite en un smartphone para clasificar: vehículos, bicicletas, motocicletas y pedestres.

2.6 Objetivos Específicos.

1. Revisar la literatura: modelos de redes neuronales profundas que tengan un mejor desempeño en smartphones, conocer si existen técnicas de rastreo de objetos específicas para smartphones, indagar sobre métricas para evaluar el rendimiento.
2. Seleccionar métodos compatibles con los smartphones Android usando TensorFlow Lite.
3. Preparar y evaluar los modelos de aprendizaje profundo seleccionados, documentar y comparar los resultados en términos de precisión y rendimiento.

2.7 Metodología:

Para este estudio se usará como recurso de hardware un smartphone que cuente con sistema operativo Android con versión base Android 5.0 (API 21). La investigación será de tipo analítica debido a que se analizarán los resultados obtenidos después de haber implementado el sistema. Determinar si un modelo con menor número de clases es más rápido en tiempo de inferencia que un modelo con abundantes clases. Para realizar esto nos basaremos en los métodos que (Alsing, 2018) usó para su tesis. A continuación, se detallan las actividades a seguir.

1. Selección de la literatura de artículos correspondientes a la visión por computadora en movilidad urbana.
    a. Implementar TensorFlow Lite en un smartphone Android para profundizar en su funcionamiento.
    b. Probar que modelos pre-compilados y datasets probados son capaces de implementarse junto a TensorFlow Lite.
    c. Posibles métricas para evaluar el rendimiento en dispositivos embarcados.
    d. Construcción de modelos con clases necesarias para movilidad urbana.
2. Seleccionar modelo a utilizar basado en la disponibilidad de código, compatibilidad del dispositivo objetivo.
3. Seleccionar un *dataset* para la evaluación del rendimiento que incluya fuentes de video en distintas condiciones ambientales. Se documentarán los resultados en términos de precisión y tiempo de inferencia.
4. Generar discusión y propuestas a futuro.

2.8 Alcances y resultados esperados:

Se deberá conocer los resultados de rendimiento en términos de medidas habiendo probado en un smartphone con sistema operativo Android. Determinar si un modelo con menor número de clases es más rápido en tiempo de inferencia que un modelo con abundantes clases. Hacer un contraste del consumo de recursos por cada método. Saber si las técnicas actuales de visión por computador que se basan en aprendizaje profundo son viables para el conteo automático de actores de movilidad urbana.

UNIVERSIDAD DEL AZUAY

2.9 Supuestos y Riesgos:

Supuestos	Riesgos	Gestión de Riesgos
El framework utilizado no se compatible con la versión del smartphone de prueba.	Incompatibilidad de la API del smartphone de prueba.	Disponer de varios smartphones de prueba.
No hay acuerdos entre universidad y repositorios de conocimiento para acceso a *artículos académicos*.	No tener acceso a fuentes confiables de información.	Buscar otras fuentes o financiamiento para adquirir la información requerida.
Consumo de energía alto cuando se estén realizando los experimentos.	No se realicen las pruebas correctamente.	Usar fuentes de alimentación externas PowerBank.

2.10 Presupuesto:

Rubro	Costo	Destino
Trabajos escritos y/o diagramas impresos	$ 100.00	Impresiones debidas de la documentación del proyecto.
Gastos Varios	$ 650.00	Consumo de viáticos y recursos dentro del proyecto (internet, servicios básicos, transporte, alimentación, etc.).
Computadora portátil para trabajo.	$ 1100.00	Adquisición de equipo para el desarrollo del proyecto.
Trabajo horas hombre.	$ 1600.00	20 horas semanales por 4 meses a un costo de hora hombre de 5$.
Tutorías de tesis y papeleo.	$ 300.00	Pago de tutor, derechos, etc.

TOTAL, SIN IVA = 3,750.00 USD

2.11 Financiamiento:

Este proyecto será financiado por el estudiante en su totalidad. Debido a las relaciones entre este tema de tesis y el proyecto "CEPRA-XII-2018-15; Movilidad" se podrá hacer uso de recursos de CEDIA como un clúster para temas de entrenamiento.

2.12 Esquema Tentativo:

- Dedicatoria.
- Agradecimientos.
- Resumen.
- Índice
- Capítulo 1 Introducción.
  - Objetivos
  - Justificación
- Capítulo 2 Revisión de la literatura.
- Capítulo 3 Métodos y experimentos.
- Capítulo 4 Resultados.
- Capítulo 5 Discusión.
- Capítulo 6 Conclusiones.
- Bibliografía
- Anexos

2.13 Cronograma:

Objetivo	Actividad	Capítulo/ Documento	Tiempo
1	Selección de la literatura de 15 artículos correspondientes a la visión por computadora en movilidad urbana.	1,2,3	3 semanas
2	Implementar TensorFlow Lite en un smartphone Android con la finalidad de profundizar sobre el framework y adquirir destreza sobre el desarrollo de aplicaciones.	2,3	3 semanas
2	Selección de datos y algoritmos.	3,4,5	2 semanas
3	Pruebas	3	3 semanas
3	Documentación y comparación.	3,4	2 semanas
3	Generar discusión y propuestas a futuro	6	2 semanas

2.14 Referencias:

Arévalo, M. (23 de Octubre de 2018). eltiempo. *Obtenido de https://www.eltiempo.com.ec/noticias/cuenca/2/aceras-cuenca-azuay*

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... Zheng, X. (2016). TensorFlow: A system for large-scale machine learning. https://doi.org/10.1016/0076-6879(83)01039-3

Alsing, O. (2018). Mobile Object Detection using TensorFlow Lite and Transfer Learning.

Chen, Z., Ellis, T., & Smieee, S. A. V. (2012). Vehicle Detection, Tracking and Classification in Urban Traffic.

Ignatov, A., Timofte, R., Szczepaniak, P., Chou, W., Wang, K., Wu, M., ... Van Gool, L. (2018). AI Benchmark: Running Deep Neural Networks on Android Smartphones. Recuperado de http://arxiv.org/abs/1810.01109

Siekkinen, M. (2018). Latency and Throughput Characterization of Convolutional Neural Networks for Mobile Computer Vision. *CoRR, ABS/1803.0*.

Sobolevsky, S., Levitskaya, E., Chan, H., & Postle, M. (2016). Impact Of Bike Sharing In New York City Future Cities Catapult , London , UK * Correspondence to be addressed to : sobolevsky@nyu.edu, 1–26.

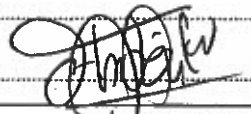
Sun, C., & Murphy, K. (2016). Speed/accuracy trade-offs for modern convolutional object detectors.

Torres, F. A., & Barros, G. (2017). Sound noise monitoring platform: Smart-phones as sensors. *European Wireless 2017 - 23rd European Wireless Conference*.

Torres, F., Barros, G., & Barros, M. J. (2018). Computer vision classifier and platform for automatic counting: More than cars. *2017 IEEE 2nd Ecuador Technical Chapters Meeting, ETCM 2017, 2017–Janua*, 1–6. https://doi.org/10.1109/ETCM.2017.8247454

Verdugo-romero, W., Gonz, L., Sacoto-cabrera, E. J., & Verdugo-cabrera, A. (2018). Mathematical and statistical analysis for the simulation of the vehicular flow in a specific sector of the city of Cuenca-Ecuador, 1–4.

2.15 Firma de Responsabilidad (estudiante):



Andrés Campoverde

2.16 Firma de Responsabilidad (director sugerido)

Gabriel Barros

2.17 Fecha de Entrega:

21 de enero de 2019