# DESARROLLO DE UN ASISTENTE DE CONDUCCIÓN LONGITUDINAL MEDIANTE UN ALGORITMO DE APRENDIZAJE PROFUNDO.


**AUTOR**

JOSEPH JHONAS VOGULYS MEDINA

Trabajo de grado presentado como requisito para optar al título de:

**MAGISTER EN MECATRÓNICA**

Director:

**RICARDO ANDRÉS CASTILLO ESTEPA PhD**

**UNIVERSIDAD MILITAR NUEVA GRANADA**

**FACULTAD DE INGENIERIA**

**MAESTRIA EN INGENIERIA MECATRÓNICA**

**BOGOTÁ, 10 FEBRERO 2020**

# Desarrollo de un asistente de conducción longitudinal mediante un Algoritmo de Aprendizaje Profundo

**Joseph Jhonas Vogulys Medina**

APROBADO:

| | |
|---|---|
| **Ing. Ricardo Andrés Castillo Estepa, PhD** | **Ing.** |
| Tutor | Cotutor |
| **Ing. Robinson Jiménez M. PhD** | **Ing. Jorge E Romero R.** |
| Firma | Firma |

Bogotá D.C. 10 de Febrero del 2020

# ○ AGRADECIMIENTOS

De esta bonita experiencia solo tengo agradecimientos. Gracias a Dios por brindarme la fortaleza y el esmero de seguir adelante, la compañía de mi familia y de aquellos compañeros que sacrificaron tiempo compartiendo conocimientos, apoyo emocional y espiritual, honrándome al conocer seres humanos con habilidades, inteligencia y calidez humana, a mi parecer, de un valor incalculable.

Un agradecimiento especial a los doctores Ricardo Andrés Castillo y Robinson Jiménez los cuales fueron tutores excepcionales, éticos, profesionales, sencillos y pacientes en la corrección documental. “La persona que hace las cosas difíciles en simples, es el educador.” Ralph Waldo Emerson.

A los ingenieros César Pachón y Camilo Castro, compañeros y formadores pedagógicos en el mundo académico, para mi crecimiento profesional e investigativo mostrando siempre su amor a la academia con perseverancia y resiliencia. “Los que saben, hacen, los que entienden, enseñan”. -Aristóteles.

Agradezco enormemente a mi padre Jonas quien desde niño me ha envuelto en un mundo apasionante en la inventiva, la electrónica, la innovación y mejora continua junto a su equipo SINCO (sociedad De Inventores Colombianos) transformando un carro de combustión interna a eléctrico, este fue el detonante, que me inspiró y llevó a seguir por la corriente de carros autónomos.

A mi madre Gloria por su ternura y apoyo, y mi hermana Minyela, siempre un ejemplo, dándome palabras de aliento y fortaleza espiritual. “Lo más importante en el mundo es la familia y el amor.” John Wooden.

Finalmente, gracias a la universidad Militar Nueva Granada una institución que quiero, respeto y me llena de orgullo. Esta casa, me permitió ejercer mis estudios, con esmero y pasión; y sin duda a mis mejores amigos y docentes, quienes siempre evidenciaron su apoyo y confianza. “La enseñanza afecta la eternidad; nunca se sabe dónde termina su influencia.” -Henry Brooks Adams

# Tabla de contenido

# ● Lista de Figuras

# ● Tabla de Ecuaciones

# ● Lista de tablas

## ○ Lista de Anexos



# ● Lista de Símbolos y abreviaturas

AI = Inteligencia artificial, (Artificial Inteligencia).

ROI=Región de interés

ONSV = Observatorio Nacional de Seguridad Vial (ONSV)

DL = Aprendizaje profundo (Deep Learning).

CNN= Redes Neuronales Convolucionales (Convolutional Neural Network).

ML = Aprendizaje de máquina (Machine Learning).

CA= Carro autónomo

OI=Objetos de interés

IOU= Intercepción sobre el objeto

# ● Resumen

El presente proyecto implementa y desarrolla una simulación de un automóvil con asistente de conducción para una trayectoria longitudinal donde el automóvil frena o acelera al detectar objetos de interés (OI), esto con el uso de técnicas de redes neuronales convolucionales y visión de máquina. El automóvil es capaz de identificar seis (6) OI (bicicletas, motos, señal de pare, automóviles, semáforos y peatones), logrando realizar estimaciones de la ubicación de los objetos y la categoría a la que pertenecen, utilizando como técnica de aprendizaje profundo (DL) la arquitectura YOLO y como red neuronal convolucional (CNN), con varias capas residuales para una mejor precisión, la RESNET50. En la simulación se implementa una cámara RGB-D acoplada al automóvil cuya función permite estimar la distancia a la que se encuentran los OI desde el automóvil. El trabajo se desarrolla en Matlab, el cual cuenta con la interfaz gráfica para etiquetar y entrenar las clases mencionadas. Se importan alrededor de 1400 imágenes, donde se compara la mejor técnica de reconocimiento entre DL y aprendizaje automático (ML), arrojando una precisión del 98.13% con una red de pocas capas, como la ALEXNET, mientras que la técnica de máquina de vector soporte obtuvo un 84.2% respectivamente, siendo la CNN superior en un 16,5% de precisión general.

Al identificar la técnica, se procede a realizar un entrenamiento, donde se etiquetan 2421 imágenes extraídas de las calles de la capital, de ellas se obtienen 4000 etiquetas divididas en las 6 clases mencionadas, las cuales son utilizadas para diseñar la arquitectura basada en regiones para la detección de objetos. Se entrena una red con 8000 imágenes de semáforos que trabaja en paralelo, la cual diferencia si los semáforos detectados se encuentran en estado verde o rojo a una precisión del 99,64%.

Una vez obtenida la arquitectura capaz de reconocer y detectar por regiones los OI siguientes: carros, motos, bicicletas, personas, señal de pare, semáforos verde y rojo, se acopla a un ambiente virtual. Para proceder a la simulación, se construye un automóvil al cual se le acopla una cámara de profundidad capaz de visualizar y estimar la distancia del

ambiente virtual, utilizando imágenes reales semejante a la realidad de las calles en Bogotá. Las ruedas, tendrán una velocidad angular constante, y mantienen una trayectoria lineal. El Automóvil utilizando la arquitectura de detección, al reconocer los OI a una distancia menor a 10 metros y utilizando la ecuación cinemática de aceleración uniforme, obtendrá las variables velocidad tiempo y posición- tiempo, reduciendo la velocidad angular de las ruedas, hasta detenerse totalmente a una distancia cercana a los dos metros, evitando de esta forma colisionar. La simulación corre en el ambiente virtual V-Rep y se acopla a tiempo real con Matlab, obteniendo como resultado con la base de datos de entrenamiento, un 93.3% de precisión – Recall de los 6 OI. Para la validación, se crea una base de datos de 600 imágenes, y se obtiene una precisión - Recall general del 60%.

**Palabras clave:**
Entorno virtual, asistente de conducción, redes neuronales convolucionales, aprendizaje profundo, carro autónomo, objetos de interés.

# ● Abstract

The present project implements and develops a simulation of a car with a driving assistant for a longitudinal path where the car brakes or accelerates when detecting OI (objects of interest), this with the use of convolutional neural network techniques and machine vision. The car is able to identify six (6) RO (bicycles, motorcycles, stop signs, cars, traffic lights and pedestrians), making estimates of the location of the objects and the category to which they belong, using as a deep learning technique (DL) the YOLO architecture and as a convolutional neural network, the RESNET50. The simulation implements an RGB-D camera attached to the car whose function allows knowing the depth at which the OIs are located. The work is developed in Matlab, which has the graphic interface to label and train the mentioned classes. Around 1400 images are imported, where the best recognition technique between DL and machine learning (ML) is compared, yielding 98.13% accuracy

in the DL network, the ALEXNET, while the support vector machine technique obtained a 84.2% respectively, the CNN being superior at 16.5% overall accuracy.
When obtaining the technique, a training is carried out, where 2421 images extracted from the streets of the capital are labeled, of them 4000 labels are obtained divided into the 6 mentioned classes, which are used to design the architecture based on regions to Object detection. A network with 8000 traffic light images that work in parallel is trained, which differentiates whether the detected traffic lights are in a green or red state at an accuracy of 99.64%.
Once the architecture has been obtained, capable of recognizing and detecting the following ROs by regions: cars, motorcycles, bicycles, people, stop signs, green and red traffic lights, it is coupled to a virtual environment. To proceed with the simulation, a car is built to which a depth camera is attached capable of visualizing and estimating the distance of the virtual environment, using real images similar to the reality of the streets in Bogotá. The wheels will have a constant angular speed, and maintain a linear trajectory. The Automobile using the detection architecture, recognizing the ROs at a distance of less than 10 meters and using the uniform acceleration kinematic equation, will obtain the variables speed time and position-time, reducing the angular speed of the wheels, until it stops completely at a distance close to two meters, thus avoiding colliding. The simulation runs in the virtual V-Rep environment and is coupled in real time with Matlab, resulting in a training database of 93.3% accuracy - Recall of the 6 OI. For validation, a database of 600 images is created, and an accuracy is obtained - General Recall of 60%.

**Keywords:** Virtual environment, driving assistant, convolutional neural networks, deep learning, autonomous car, interest objects.

# CAPITULO 1 INTRODUCCIÓN

El crecimiento de la ciudad y la expansión rural, han generado la necesidad de transportarse en automóvil, el cual ha generado que pasen de rodar en el mundo, treinta millones de automóviles en el siglo XX, a mil doscientos millones (1,2 millares) en la última década [1]. El uso de este transporte ha incentivado el desarrollo de asistentes para una conducción más cómoda y segura, donde en [2], durante el año 1977, el laboratorio japonés Tsukuba es pioneros en desarrollar un prototipo de automóvil capaz de seguir los marcadores blancos en el asfalto. Se presentan otros desarrollos en [3], donde el automóvil es capaz de dirigirse y mantenerse en un mismo carril conservando distancia y evitando colisionar con automóviles que encontraba al frente mientras se dirigía en avenidas [4], logrando estimar distancias y velocidad entre automóviles. Los desarrollos en **carros autónomos (CA)**, presentan varias de sus ventajas en [5], permitiendo que la conducción de un automóvil no tripulado pueda lograrse en entornos dinámicos. El desarrollo de los automóviles con conducción totalmente autónoma ha sido lenta por la necesidad de obtener una “visión panorámica”, los sensores que percibían el entorno eran capaces de identificar las líneas del asfalto y verificar posibles objetos con los que puede colisionar a través de sensores ultrasonido y laser [6], mas no por visión artificial la cual, cobra importancia al Fukushima presentar en el 80 y Lecun en el año 2000 las primeras técnicas de reconocimiento de imágenes con convoluciones. Posteriormente, se presentan propuestas de conducción inteligente en un ambiente virtual con redes neuronales para movimiento transversal utilizando fuzzy para movimiento longitudinal [7] y otros para reconocimiento lineas de carretera utilizando el algoritmo, máquina de vector soporte (SVM) [8]. En un ambiente real, se deben considerar no solo las líneas que demarcan las carreteras, como maniobran en modo turismo un asistente a la conducción, sino también, el reconocer escenarios con personas, señales de tránsito y otros OI. El auge en este desarrollo incentivó a la comunidad científica y militar, a construir un automóvil capaz de reconocer agentes urbanos utilizando visión de maquina en entornos controlados, obteniendo unos prototipos gracias al patrocinio de DARPA[1] Challenge [9], el cual convoca a las universidades más

[1] Agencia norteamericana de Proyectos de Investigación avanzada en la defensa (DARPA).

prestigiosas del mundo a participar, logrando los primeros avances experimentales en 2007 para CA, reconociendo patrones.

El aprendizaje de maquina (ML), ha sido pieza esencial desarrollando algoritmos para la visión de máquina en la conducción autónoma [10] [11]. Sus primeros trabajos se enfocan en la clasificación de dígitos numéricos manuscritos, a otras técnicas más complejas basada en técnicas de redes neuronales. El desarrollo en hardware y tarjetas de video, promovieron un nuevo tipo de red neuronal más compleja, la cual crea filtros automáticamente identificando y clasificando imágenes a través de técnicas de aprendizaje profundo, como son las redes neuronales convolucionales (**CNN**).Se presenta una CNN capaz de clasificar diez categorías (aves, perros, camiones) en el concurso CIFAR-10, alcanzando a reconocer hasta 1000 clases distintas con el IMAGNET [12].

Considerando las CNN como una arquitectura confiable en el estado del arte y de una precisión aceptable en identificar patrones, se implementó y desarrolló en este trabajo una simulación de un asistente para la conducción en un automóvil, el cual realiza una trayectoria longitudinal siendo capaz de reconocer ciertos OI, utilizando como sensor una cámara RGB-D ubicada sobre el automóvil, para visualizar a tiempo real el entorno y al mismo tiempo utilizando una arquitectura de aprendizaje profundo**,** el automóvil frene o acelere, reduciendo las posibilidades de chocar con algún obstáculo en las calles.

## ○ 1.1 Planteamiento del problema

En Japón, el gobierno ha destinado enormes fondos para hacer realidad la conducción total autónoma y un transporte más seguro y eficiente a través de las olimpiadas 2020.Los autos con conducción semi autónoma y autónoma [13], como los conocidos TESTLA, o el Mercedes Benz clase E [14], son de difícil acceso en países en desarrollo debido a los altos costos asociados en personal especializado, infraestructura y sensórica [5], a pesar de los esfuerzos de empresas como WAYMO[2] [15] ,en buscar reducir los precios de sensores para la industria en carros autónomos. Por ejemplo, el sensor **LIDAR** (Laser Imaging Detection and Ranging*)* es utilizado en la navegación y detección de obstáculo el cual crea un mapa virtual de 360 grados gracias a la resolución del láser junto al **GPS** (Global Positioning

[2] WAYMO antes conocida como Google self-driving car project, perteneciente al conglomerado Alphabet

System) [3] [16], los cuales facilitan el posicionamiento a tiempo real, sin embargo, pueden estar entre los 3000 y 10000 USD adquirir uno de ellos. Los costos asociados en los **CA** para Colombia y su implementación es poco probable de acuerdo con expertos en automotores. Promotores como el director de la revista 'Motor' considera que "varias décadas tendrán que pasar para que los automóviles autónomos rueden en ciudades colombianas, debido a los problemas de infraestructura y cultura ciudadana". En contraparte, la revista Business Insider en [17], considera necesaria la transición a carros autónomos, ya que "reducirá los accidentes, el consumo de energía, la contaminación y el tráfico", proyectando para el 2025 más de 20 millones de automóviles autónomos circulando sin la intervención de seres humanos. Para el desarrollo de los CA, Waymo, la división de carros autónomos de Alphabet que pertenece Google, asegura que sus vehículos han recorrido casi cinco millones de kilómetros desde que empezaron las pruebas [18], lo que ha permitido reducir el tiempo en carretera al estar conectada a la nube subiendo 25 GB/hr, comunicándose gracias al desarrollo en conectividad 4G y 5G, reduciendo el tiempo de movilizarse. En [19], se menciona que en Colombia frente a los Estados unidos en capacidad adquisitiva complica la obtención de un **CA**. Algunos expertos afirman que en Colombia se paga el doble por el mismo auto, considerando que el salario mínimo es 6 veces inferior al país norteamericano, donde el 20% del ingreso promedio de un ciudadano, se gasta en combustible y transporte".

La obtención de un CA en Colombia no solo es un lujo, la alta tasa de accidentalidad vial, se ha convertido en discusión mundial [20] [21], donde incluso investigadores han entrado a revisar la autonomía en la conducción como un riesgo en las vías y la ética que conlleva el uso de esta tecnología, en [22]. Fatales accidentes como el hecho ocurrido en 2019 en [23], donde un coche Tesla Model 3 sobrepasa la velocidad permitida utilizando la función Autopilot [134], genera que organismos de control como el Consejo Nacional de Seguridad en el Transporte (NTSB) de los EE. UU argumentara la necesaria participación humana en los coches autónomos.

Los beneficios de los autos autónomos son discutidos en [24], ya que la tecnología aporta en aspectos como reducir la siniestralidad generada por errores humanos [25], uso eficiente en combustible, aliviar atascos y mejorar el tránsito entre los vehículos [26], convirtiéndose en una prioridad por la Comisión Ética en la conducción automática 2017.

Colombia podría considerar la necesidad de una implementación de un asistente de conducción para automóviles que aporten a la reducción de siniestros viales, sin embargo, los automóviles que cuentan con estos sistemas, son costosos y aplicables para autopistas bien señalizadas, amplias y de grandes trayectos.

Según la ministra de Transporte en Colombia Ángela María Orozco, comenta que cada 94 minutos pierde la vida un colombiano en las vías, es decir 15 personas fallecen al día por accidentes de tránsito [27].

En Colombia, de acuerdo con la información preliminar del Observatorio Nacional de Seguridad Vial (ONSV) en [28], entre enero y diciembre de 2017 se registra un total de 6.479 fallecidos y 38.073 lesionados en accidentes viales, donde la mayoría de las víctimas (81,07%) son personas entre los 20 y 25 años según el Instituto Nacional de Medicina Legal y Ciencias Forenses (INMLCF) presentado en la **Figura 1**



**Figura 1 Fallecidos y lesionados en hechos de tránsito. Fuente: Observatorio Nacional de Seguridad Vial.**

Es importante identificar la diferencia semántica entre accidente y siniestro. Para la OMS (Organización Mundial de la Salud) un accidente es un suceso inevitable, en cambio, un siniestro vial es aquel que se pudo prevenir. De acuerdo a [25] en noticias EL TIEMPO, las principales causas de siniestros automovilísticos en Colombia del 2010 a 2016 fueron:

- **Saltar normas de tránsito**: el irrespeto a las señales ocasionó el 12 % de accidentes en 6 años: 148.501 en total.
- **No mantener la distancia**: depende de la velocidad máxima permitida. Por ejemplo, en zonas de máximo 30 km por hora debe haber 10 metros entre vehículos. Los accidentes por esta causa fueron 124.159.
- **Transitar entre vehículos**: ciclistas y motociclistas ocasionaron 71.841 accidentes por transitar entre filas de otros vehículos.
- **No ser visibles en la vía**: por no hacerse visibles con chalecos reflectivos, ciclistas y motociclistas ocasionaron 65.376 accidentes.
- **No respetar la prelación**: el hecho de no ceder el paso ni detenerse en cambios a vías rápidas o glorietas ocasionó 35.709 accidentes.
- **Adelantar cerrando**: adelantar a un vehículo obstruyendo su camino fue causal de 29.564 incidentes. Solo en el 2013 fueron 7.037 los accidentes por esta causa.
- **No mirar antes de cruzar la vía**: cruzar sin observar a ambos lados es la única causa del listado en la que los responsables son peatones. Ocasionó 25.051 accidentes.
- **Exceder límite de velocidad**: ir más rápido de lo permitido ocasionó unos 23.595 accidentes. Esta es la principal infracción de tránsito en el país.
- **Invadir zona peatonal**: los vehículos que se suben a andenes, zonas peatonales o separadores deliberadamente causaron 14.432 siniestros.
- **Aprovisionar indebidamente**: entre 2010 y 2011, llenar el tanque de combustible con pasajeros a bordo o con el motor encendido causó 11.055 accidentes

Los sucesos obtenidos anteriormente son cifras de siniestros a nivel Colombia durante 6 años. A nivel distrital según el RUNT (Registro único Nacional de Tránsito), Bogotá presentó en 2015 los siguientes sucesos relevantes [29].

- Las zonas con mayores muertes por siniestros viales se presentaron en Kennedy, suba, ciudad Bolívar y Engativá con 60 muertes en promedio por localidad.
- De 543 fallecidos en siniestro se dividen en 49% por peatones, 12% ciclistas, 24% motociclistas, 12% pasajeros, 3% conductores.
- En Bogotá circularon en 2015 un total de 2.017.779 vehículos, distribuidos en: 1.074.408 automóviles, 449.283 motocicletas, 259.284 camionetas, 208.307 camperos y 26.497 otros.

- De 300.000 comparendos, el 47% fue por estar mal estacionados, 27% no respetar las normas de tránsito, 11% no tener la revisión técnico-mecánica,7% transitar en zonas restringidas y 7% bloquear una calzada.
- La velocidad promedio fue de 25km/h en la ciudad.

Por otra parte, no solo los siniestros viales son un tema de interés, ciudades importantes de Colombia sufren de extenuantes trancones como se ve en [130]. Según la empresa IRIX 2018 Global Traffic ScoreCard, quienes evalúan el tráfico mundial, Bogotá ocupa el **tercer puesto** como la ciudad que más tiempo ocupa en trancones seguido de Moscú y Estambul, perdiendo aproximadamente **272** horas productivas al año.

Considerando los motivos obtenidos del observatorio nacional de Seguridad Vial y el RUNT, los siniestros viales que pudieron ser prevenidos (en su mayoría por errores humanos), trae a consideración para este trabajo, la utilización de herramientas tecnológicas que presten a los conductores una asistencia al conducir, aportando a reducir la tasa de siniestros preocupante, quienes son los más afectados y deben considerarse OI. Los siniestros viales, son causalidades producto de imprudencias, entre ellos están principalmente: saltarse las normas de tránsito, no mantener distancia, el exceso de velocidad, el tráfico denso y no ser visible en la vía. Conociendo estos efectos negativos, se identifica la necesidad de la creación de un software que funcione como asistente a la conducción, capaz de observar eventos en las calles, clasificando y detectando la distancia que separa al automóvil de un objeto de interés o de señal de tránsito, sirviendo de apoyo visual para que el conductor se abstenga de cometer las faltas de tránsito. Este trabajo, plantea generar un código en Matlab y simularlo en V-Rep, donde el automóvil tenga la capacidad de reconocer distancias y clasificar agentes reales de las calles de Bogotá sin poner en riesgo el colisionar con algún agente o afectar a peatones o vehículos.

¿Cómo se puede implementar un algoritmo de aprendizaje profundo, donde un automóvil sea capaz de identificar la categoría de objetos de interés considerando la distancia en la que se encuentran durante una trayectoria longitudinal, acelerando o deteniéndose, evitando colisionar con otro agente?

## ○ 1.2 Justificación

Se conoce que los conductores que obedecen las leyes de tránsito y las regulaciones reducen la posibilidad de tener accidentes automovilísticos innecesarios [30]. Se han observado imprudencias como uno de cada cuatro conductores de automóvil no usa indicadores (direccionales) al cambiar de carril, otros, sobrepasar los límites de velocidad los cuales generan 1,25 millones de heridos y muertos en todo el mundo [31].

La academia y las empresas invierten sumas de dinero en el desarrollo del CA [32]. Algunos experimentos realizados en visión de máquina para automóviles, se realizó con cámaras en cuatro lados para obtener una visión de 360 grados utilizando algoritmos de AI [33] [34]. El algoritmo debía ser capaz de visualizar offline al perder la conectividad, por ejemplo, cuando transita en un túnel. La visión se logra utilizando el mapeo de entornos a gran escala como los sistemas SLAM y posición geográfica con radares y GPS, para estimar trayectorias del automóvil en escalas tridimensionales optimas [35] [16], sin embargo, los sensores requeridos por su precisión y calidad tienen elevados precios.

Adquirir un **CA** en Colombia es considerado un lujo debido a el ingreso promedio, políticas arancelarias de importación altas ,infraestructura vial no adecuada, percepción negativa de la cultura ciudadana y el irrespeto por las normas de tránsito, plantean como opción para este trabajo, la creación de un prototipo de software aplicado en una simulación, que sirva como asistente de conducción , brindando un apoyo al conductor, informándole al piloto a qué distancia y velocidad debe mantenerse para evitar colisionar con algún objeto de interés que se localice en una panorámica frontal. En la simulación se debe poder con imágenes reales, señales de tránsito, entre ellos semáforos en señal roja y verde, señal de pare, y o mantener distancia frente a otros OI, entre ellos, motos, bicicletas, peatones y otros automóviles, promoviendo una conducción segura en Bogotá. Para la creación de este Asistente a la conducción, se requiere conocer técnicas de AI, que aportan en la visión artificial.

La utilización del ML ha facilitado el desarrollo de lo que hoy es la visión de máquina para clasificar y estimar distancias con el uso de cámaras RGB y profundidad [10] [36]. Diferentes algoritmos en reconocimiento, han aumento progresivamente su precisión de imágenes cada vez más complejas (colores y dimensiones de pixelación).Concursos como el MINST, CIFAR-10 e IMAGENET 1000 [12], y el avance conjuntamente de tarjetas gráficas, han

permitido el calculo que se requiere para el reconocimiento de patrones para cientos de aplicaciones, y posteriormente el avance de clasificar y detectar por regiones, facilita en este caso concreto ,la aplicabilidad de un algoritmo durante la conducción.

En conclusión, considerando la capacidad de compra en Colombia para los autos autónomos, el tiempo perdido en trancones, los errores humanos, la infraestructura vial en malas condiciones e imprudencias de conductores, se plantea la creación de un prototipo de asistente a la conducción simulado, usando un algoritmo de DL conocido como las **CNN** basado en regiones, que permita mejorar la seguridad en las calles, al dar aviso de distancia, y clasificar agentes de interés (automóviles, señales de tránsito y personas) en la conducción convencional en trayectorias longitudinales.

## ○ 1.3 Objetivos

En esta sección se presenta el objetivo general, cinco objetivos específicos y las delimitaciones, seguido de la presentación del documento y la metodología por etapas.

### 1.3.1 Objetivo General

Desarrollar e implementar un asistente de conducción para trayectorias longitudinales en un automóvil, utilizando un algoritmo de aprendizaje profundo dentro de un entorno virtual tridimensional.

### 1.3.2 Objetivos específicos

- Crear dos bases de datos, una data set para elección de algoritmo de clasificación. Una segunda data set para el entrenamiento y pruebas de detección, mediante la recolección y etiquetado de imágenes extraídas en las calles de Bogotá.

- Desarrollar en un entorno virtual tridimensional (3D) para la simulación del tránsito de un automóvil con una cámara de profundidad, que simule la realidad parcial de las calles colombianas y permita evaluar el sistema de reconocimiento de patrones escogido.

- Evaluar al menos dos tipos de algoritmos para reconocimiento y clasificación de imágenes, mediante métricas como matrices de confusión.

- Implementar dentro del entorno de simulación el algoritmo de reconocimiento de patrones, que permita al agente móvil tomar decisiones durante su trayectoria longitudinal acelerando o frenando, en función de la distancia del objeto de interés.

- Evaluar el funcionamiento del algoritmo implementado para evitar colisiones y mantener una trayectoria de desplazamiento segura, utilizando diferentes medidas de rendimiento como precisión vs Recall y tiempo de procesamiento.

### 1.3.3 Delimitación

- El algoritmo será probado en un entorno virtual, utilizando imágenes reales, para de esta forma evitar posibles daños a terceros.
- La simulación tendrá en cuenta únicamente el movimiento longitudinal como lo tienen los autos autónomos nivel uno (1) y reconocimiento de algunos patrones de transito como lo tiene el nivel tipo dos (2) de conducción autónoma según la SAE.
- El móvil no reconocerá las líneas del carril en la carretera
- Para el entrenamiento de las redes de aprendizaje profundo se tendrán en cuenta hasta un máximo de seis (6) categorías distintas, entre las que se incluyen personas, carros, semáforos, motos, bicicletas y señal de pare a una distancia menor a 10 metros, esto con el fin de acotar las etiquetas de la red, enfocándose principalmente en los objetos más importantes hallados en las calles vs confiabilidad.
- El algoritmo pretende únicamente conseguir una trayectoria lineal capaz de seguir la ruta evitando colisionar con posibles obstáculos.
- Los tiempos de ejecución del algoritmo se pueden llegar a ver afectados por el hardware utilizado (GPU NVIDIA 920) y el algoritmo al redimensionar el tamaño de las imágenes.
- En la simulación no se tendrá en cuenta variables de fricción del suelo o el viento, considerando el reconocimiento de patrones como prioridad a pesar de evaluarse en un entorno ideal y controlado.
- Este trabajo No pretende solucionar todos los problemas convencionales en la navegación autónoma en las que se evalúan parámetros como a) Localización: el robot debe saber en todo momento dónde se encuentra. b) Mapeo: el robot debe saber construir una representación de su entorno o mapa. c) Planificador de rutas: el robot debe saber calcular el mejor camino para llegar a su destino.

## ○ PRESENTACIÓN DEL DOCUMENTO

El presente trabajo fue desarrollado en siete facetas. El capítulo introductorio, describe brevemente por qué el interés en la conducción autónoma, seguido de datos relacionados a siniestros viales en Colombia, y visión de máquina para el reconocimiento de objetos de interés. Se generaliza el contexto de porque es útil un sistema de conducción con AI para automóviles en Bogotá, justificando el porqué de la creación de un software que funcione como asistente a la conducción, y permita un aporte en reducir las causas de siniestros viales, planteando unos objetivos y delimitaciones.

El segundo capítulo describe el estado del arte de la visión artificial y los CA. Inicia con generalidades en el reconocimiento de patrones por visión computacional, qué se entiende como niveles de conducción autónoma, y los sensores que utilizan. Seguido, una revisión de los antecedentes en aprendizaje profundo explicando qué son las redes neuronales y una arquitectura de red neuronal convolucional basada en regiones. Por otro parte los antecedentes en carros autónomos, exponiendo su historia, un marco de referencia y técnicas utilizadas para pilotos autónomos en situaciones controladas.

En la sección tres, inicia el desarrollo práctico del trabajo titulado “Desarrollo de un asistente de conducción longitudinal, utilizando un algoritmo de aprendizaje profundo”: Se describe la elaboración de dos bases de datos con imágenes, la primera es utilizada como criterio de elección para escoger el algoritmo de visión computacional, implementada en la sección 5.2, 5.3 y 5. 4, para entrenar diferentes algoritmos de ML y uno de DL. Se compara la precisión del mejor algoritmo reconociendo siete (7) OI seleccionados y en base a esta evidencia, se selecciona el algoritmo de DL. Para la segunda base de datos, se etiquetan 2421 imágenes, la cual utiliza la arquitectura YOLOv2 de red neuronal convolucional, para desarrollar el asistente de conducción al clasificar seis (6) etiquetas, la cual se desarrolla en la sección 5.5 de este trabajo.

En el capítulo 4 se describe brevemente la creación del entorno virtual tridimensional de un automóvil, utilizando una cámara de profundidad (RGB-D) en su techo. El automóvil, podrá visualizar un ambiente controlado y realista, utilizando imágenes extraídas de las calles de Bogotá como textura en paredes para darle realismo. La fórmula de cinemática considera las velocidades angulares de las ruedas a partir de los objetos que detecta la cámara en un rango de 8 a 2 metros. El acople entre el software V-Rep y Matlab se realiza a través de una interfaz gráfica (.gui).

En la sección 5 se evalúan técnicas de reconocimiento por extracción de caracteres. Posteriormente se realiza el entrenamiento de 6 algoritmos de técnicas clásicas de ML con la base de datos del apartado 3.1, tomando el de mejor resultado en precisión identificando siete agentes de interés. Seguido se utiliza la base de datos 3.1 para una técnica de aprendizaje profundo, en este caso por transfer Learning se pone a prueba la red neuronal convolucional ALEXNET. Una vez se evidencia la superioridad en precisión del Deep Learning frente a técnicas clásicas del ML, en el apartado 5.5, se realiza un nuevo entrenamiento utilizando la base de datos de la sección 3.2, para la detección y clasificación de 6 agentes con la arquitectura del algoritmo YOLOv2 (carro, moto, persona, pare, bicicleta, semáforo) y una red neuronal extra para la diferenciación entre el semáforo rojo y verde, las cual se implementan en el simulador V-Rep.

El capítulo 6 analiza los resultados y comportamiento del automóvil simulado al detectar los OI frenando o acelerando en la trayectoria longitudinal. Por otra parte, se analizan los resultados del algoritmo reconociendo y detectando los OI, tomando la base de datos de entrenamiento y otra de validación para evaluar la precisión-Recall. Finalmente, en el capítulo 7 se incluyen conclusiones y trabajos futuros. Al finalizar se encontrará anexos en la caracterización de los materiales y equipos.

# METODOLOGIA

El trabajo realizado obedece a la implementación de un asistente a la conducción en un simulador semejante al nivel dos de seis categorías según la SAE. Utilizando una cámara y algoritmos de aprendizaje profundo sobre el automóvil simulado, este será capaz de clasificar, detectar y estimar la distancia de diferentes objetos de interés en las calles en Bogotá. Para lograr entrenar el algoritmo fue fundamental la elaboración y etiquetado de imágenes como bases de datos, necesarias para el entrenamiento de un tipo de red neuronal bautizada red neuronal convolucional, la cual aprenda a detectar los OI, tome decisiones de frenar o acelerar en su trayectoria longitudinal y de acuerdo con lo que detecte utilizando la cámara estereoscópica evitando colisionar.

Por tal motivo, se dividió la investigación en 6 pasos metodológicos, los cuales se dividen en:

1. *Revisión de antecedentes en visión artificial*

A partir de la problemática en Bogotá de siniestralidad vial y atascos, surge la posibilidad de implementarle un sistema como asistente a la conducción, tratando de asemejar lo que industrias multimillonarias de automotores en el mundo buscan conseguir, la conducción autónoma. En el desarrollo de este proceso se investigan antecedentes en técnicas para la conducción autónoma, algoritmos de reconocimiento de imágenes y visión computacional a partir de técnicas clásicas de machine Learning y posteriormente de Deep Learning, utilizando una cámara de profundidad RGB-D, la cual obtendrá la información a tiempo real.

2. *Creación de bases de datos*

Se crean dos bases de datos, una para comparar la capacidad de clasificación entre dos algoritmos, y la otra base de datos para el desarrollo del algoritmo de asistente para la conducción, utilizando una arquitectura que clasifica y detecta por ROI (regiones de interés), en este caso la YOLOv2.

Para la **primera base de datos**, se toman alrededor de 200 imágenes para cada una de las 7 etiquetas seleccionadas. Para acotar la investigación, estas 7 clases se dividieron en

(carros, señal de pare, bicicletas, motos, personas, semáforo verde y semáforo rojo) para un total aproximado de 1400 imágenes.

Para la creación de la **segunda base de datos**, se utilizan una interfaz gráfica de Matlab (ImageLabeler), para etiquetar seis (6) clases divididas en carros, señal de pare, bicicletas, motos, personas y semáforos[3]. Para la extracción de imágenes, se realizan varias grabaciones con la misma cámara para evitar problemas de resolución, los cuales serán utilizan como frames, para generar un base de datos de entrenamiento y de validación con etiquetas de OI encerradas en rectángulos, las cuales serán detectadas y clasificadas utilizando una arquitectura basada en regiones en el capítulo 5.

*3-Desarrollo del ambiente virtual de simulación*

Se utiliza el ambiente virtual V-Rep, el cual es un software libre y de fácil acople a Matlab. Se crea un ambiente de simulación que presenta imágenes reales de las calles de Bogotá en paredes las cuales se pondrán como textura, donde el automóvil importado como archivo .obj utilizando una cámara RGB-D observe y obtenga la profundidad del ambiente virtual tridimensional, Por otro lado, se crea una interfaz gráfica desde Matlab. GUI, para reproducir videos almacenados, y un botón que se sincroniza y acopla con v-Rep, para visualizar a tiempo real lo que “ve” la cámara del automóvil.

*4-Evaluación, selección e implementación del algoritmo de clasificación y detección.*

En esta sección se utilizan las dos bases de datos creadas en el capítulo 3, una primera fase para seleccionar el algoritmo adecuado, y la segunda base de datos se utiliza para entrenar la red por detección que se expone a continuación. En un primer momento, se realizan tres etapas para verificar qué técnica clasifica con mejor precisión, las siete clases de la base de datos #1 elaborada. En la primera etapa, se extraen características con funciones de procesamiento de imágenes, posteriormente en la segunda etapa se utilizan algoritmos de técnicas clásicas de aprendizaje automático para evidenciar usando la app Classification learner entre varios algoritmos y con el mejor en precisión se almacenan las características. En la tercera etapa se entrena una red neuronal convolucional y se obtienen los datos de precisión. Final mente se compara y selecciona cuál de las técnicas es más eficiente en reconocer las clases de la primera base de datos.

[3] Los semáforos etiquetados están en luz roja, amarilla y verde

En la segunda sección se utiliza la arquitectura YOLOv2 capaz de detectar por regiones. En este caso se utilizó la segunda base de datos la cual contiene ya no 7 sino 6 clases, siendo una de estas clases, el semáforo. La arquitectura YOLO junto a la red resnet50, ya no solo clasifica, sino detecta simultáneamente varios agentes en el espacio. Posteriormente, para diferenciar si el agente semáforo está en estado verde o estado rojo, se requerirá entrenar una red neuronal convolucional independiente, capaz de detectar su estado.

*5-Analisis y resultados*

En este capítulo se presentará el comportamiento del automóvil simulado en V-Rep, a partir de la detección de la red YOLOv2 entrenada, y se evaluará con métricas de calidad con gráficos de precisión vs Recall utilizando el data set #2 de entrenamiento y validación.

En este punto se van a evaluar decisiones de tal modo que el automóvil frene o acelere cuando sea necesario de acuerdo a la distancia y agentes de interés. Se realiza un acople entre Matlab como interfaz para correr la arquitectura YOLOv2 y V-Rep la creación del automóvil el cual tendrá una cámara RGB-D en la parte superior capaz de conocer la distancia de los objetos de interés, reduciendo la velocidad angular de las ruedas, a partir de que se acerque a uno de los objetos de interés si así se requiere.

*6- Presentación de conclusiones y resultados*

Se describen resultados, y conclusiones de el por qué utilizar redes convolucionales con arquitecturas de detección por regiones y si es factible la utilización de un asistente de conducción para automóviles en Bogotá.

# CAPITULO 2. MARCO REFERENCIAL

Este capítulo, explica con más detalle qué es visión de máquina, los pasos para el reconocimiento de objetos y algoritmos de aprendizaje automático supervisado. Se describe la base del funcionamiento de cómo funciona una red neuronal, seguido la explicación de la arquitectura de una red neuronal convolucional (CNN), referenciando antecedentes del aprendizaje profundo en clasificar y detectar agentes por regiones, referenciando la arquitectura YOLO. Una segunda etapa contextualiza los 6 niveles de conducción citados por la SAE, tomando como referencia para este trabajo el nivel dos bautizado como asistente de conducción y la sensórica general en los CA. Por otra parte, se explica en la sección de antecedentes, la evolución del aprendizaje profundo, redes neuronales por regiones, la historia y aparición de los CA y diferentes algoritmos utilizados para crear documentos de investigación en conducción autónoma.

## ○ 2.1 INTELIGENCIA Y VISIÓN ARTIFICIAL

En la sección de inteligencia y visión artificial se explica brevemente, qué se entiende como visión de máquina, los pasos para el reconocimiento de objetos, algoritmos clásicos de machine learning en aprendizaje supervisado, qué es una red neuronal y algoritmos actuales de aprendizaje profundo utilizados en el reconocimiento, clasificación y detección de patrones, entre ellas las redes neuronales convolucionales (CNN) y la arquitectura para detección YOLO.

### ▪ 2.1.1 Visión de maquina

Dentro de la IA (Inteligencia Artificial) se encuentran técnicas especializadas en visión artificial. Los métodos presentados brindan soluciones reales y eficientes para procesar, analizar e interpretar videos e imágenes. En la **Figura 2** se ilustra en un flujograma las etapas generales en el procesamiento de imágenes, desde la adquisición, digitalización y reconocimiento [97]**.** En la visión por computador, es necesario digitalizar la imagen, la cual se representa en pixeles y datos numéricos para ser interpretada en lenguaje de máquina. Las imágenes extraídas, son una representación bidimensional gráfica de lo que percibe el ojo humano visto en el plano digital, en un conjunto de valores capturados por una cámara

u objeto óptico, que visualmente se representa en escala de grises o en colores por RGB (Red, Green, Blue), refiriendo la síntesis aditiva, como la obtención de una gama de colores a través de la suma de distintas longitudes de onda. Los colores y su combinación generan formas, los cuales se abstraen con características de aristas, bordes, brillo, color y ángulos [37]. Estas características se estudian en el procesamiento de imágenes con el uso de filtros que con apoyo de la visión artificial se reorganizan con características redundantes para clasificar o detectar objetos.

Entre las aplicaciones más destacadas en el reconocimiento de imágenes se encuentra [38]:

- Reconocimiento de patrones: Se define como "El reconocimiento de características únicas que identifican a un sujeto de otros de la misma especie [39]. El reconocimiento de patrones es una técnica para el procesamiento de la visión computacional.
- Seguimiento de objetos: Es una de las aplicaciones más utilizadas en la actualidad, permitiendo la ubicación de un objeto a tiempo real. Durante la detección se realizan técnicas de reconocimiento de objetos, lo que aumenta la complejidad del proceso. Ha tenido varias aplicaciones en vigilancia e inteligencia de negocios, aplicaciones médicas y vehiculares.
- Reconstrucción: La reconstrucción de objetos es posible gracias a la AI, capaz de lograr que una imagen incompleta pueda ser reconocida y clasificada. Se aplica en el análisis de restos arqueológicos y pictografías.
- Extracción de características: Se basa en la identificación y selección de características (features) principales que están contenidas en una imagen. Estos features, son útiles para clasificar y detectar un objeto.
- Segmentación: La segmentación se define como "la división de una imagen en sus partes u objetos constituyentes" [40]. La segmentación permite obtener los objetos o áreas de interés, seleccionar las de mayor probabilidad de aparición y rechazar las áreas con ruido para así obtener el objeto. El tipo de segmentación a seguir depende de la aplicación, ya sea en el campo de detección o clasificación del objeto de interés.


**Figura 2 Etapas de un Sistema de Visión de Máquina. [97]**

### ▪ 2.1.2 Pasos para el reconocimiento de objetos

En general, el reconocimiento de objetos, parte del procesamiento de imágenes, parametrizando, reconociendo y clasificando, al identificar el OI, con una probabilidad o score de 0 a 100, señalando “qué tanto se parece una imagen a otra”. De esta manera, si hay una correcta detección e identificación, se cataloga como verdaderos positivos, de lo contrario, como falso positivo; Al mismo tiempo, debe proporcionar un rápido reconocimiento para entornos dinámicos, siendo capaz de identificar los bordes y la posición del objeto;

De forma general, el reconocimiento automático de objetos se obtiene mediante los siguientes pasos [38]:

• Pre – procesamiento: En esta sección aplican técnicas que ajustan la imagen utilizando la mejora de contraste, la reducción de ruido, la mejora de características, la normalización del brillo, la transformación del espacio de color, entre otras.

• Etapa de parametrización: En esta etapa se definen los patrones, los cuales permitirán discriminar a un objeto con información general de sus características. Los descriptores generales son aquellos que contienen información básica, como el color [41], el movimiento [42] la forma [43] la localización o la textura [44] entre otros. Para la extracción de caracteres específicos se encuentran los atributos topológicos [45] perímetro, área [46], distancias, tamaño y orientación [47].

• Construcción del clasificador: Esta etapa es esencial en la clasificación y detección de objetos, construyendo un modelo con imágenes entrantes de acuerdo con los parámetros elegidos. Los objetos que contienen los mismos descriptores se agrupan para discriminar, separar y obtener una respuesta deseada aproximada. De este modo automáticamente se reciben los parámetros.

• Reconocimiento del objeto: Una vez definido el clasificador, este tiene la habilidad de identificar objetos de interés en una entrada sea un frame de video o imagen, dando como

resultado una respuesta de salida en un score según los descriptores. De esta manera, se completa el ciclo de detección del objeto, donde el clasificador es capaz de detectar el objeto a partir de cualquier entrada.

### ▪ 2.1.3 Aprendizaje supervisado

"En el aprendizaje supervisado existen técnicas de ML donde el arquitecto al diseñar el clasificador, agrupa en la carpeta "a" imágenes de solamente una clase, por ejemplo perros, y en la carpeta "b" solamente gatos, de tal forma, el algoritmo al percibir caracteres que pertenezcan o se asemejen más a los descriptores de contenido en las imágenes de la carpeta "a" que "b", arrojará entonces, el score de probabilidad en ser un verdadero positivo, sin embargo, el rendimiento de los algoritmos de aprendizaje supervisado, requieren de la evaluación humana para evidenciar si hubo un correcto funcionamiento.

Entre el aprendizaje supervisado existen tres tipos [49]

Aprendizaje por corrección de errores, Aprendizaje por refuerzo y Aprendizaje estocástico [38].

- Aprendizaje por corrección: la estructura de esta técnica, encuentra las diferencias de cada salida real frente a la deseada, siendo semejante al error cuadrático medio consiguiendo minimizar el error entre las salidas
- Aprendizaje por refuerzo: Aunque este algoritmo es más lento, es evaluado en juegos de video gracias a una "puntuación" positiva que se da cuando el algoritmo explora nuevas áreas y las hace correctamente, en caso contrario obtendría puntuación negativa.
- Aprendizaje estocástico parte desde la aleatoriedad de datos hasta obtener una salida deseada.

Entre los algoritmos de aprendizaje supervisado más conocidos se encuentran a continuación en la **Tabla 1**:

**Tabla 1 algoritmos supervisados, Pasquel [38].**

| Algoritmo | Descripción / Función | Aplicaciones | Ventajas | Desventajas |
|---|---|---|---|---|
| Árbol de decisión | Busca clasificar los elementos mediante el uso de una serie de variables y posibilidades | -Medicina, identificación de patrones de lesiones fatales (Timarán, Calderón, & Hidalgo, 2017).<br>-Industrial, detección de niveles de agua (Rivas, 2008). | Llega a una conclusión lógica presentando las posibles consecuencias y eventos fortuitos, construyendo un modelo. | En cada elección considera únicamente una hipótesis por lo que lo hace un proceso lento. |
| K vecinos más cercanos (k-NN) | Se centra en criterios de vecindad, dado un conjunto de ejemplos clasificados, se clasifica uno nuevo a la clase que contenga la mayor cantidad de vecinos más cercanos. | -Industrial, clasificador del producto para predicción de precios (Troncoso, 2007).<br>-Medicina, identificador de señales respiratorias para clasificar pacientes (González & Giraldo, 2015). | Es un algoritmo sencillo debido a que no es necesario construir un modelo explicito, ni hacer suposiciones sobre los datos. | Lentitud en la clasificación cuando existen muchos datos, dependencia de la función distancia, gran influencia del ruido en los datos. |
| Naïve Bayes | Es capaz de clasificar un nuevo elemento que no tenga grupo perteneciente, haciendo uso de elementos ya conocidos. | -Industria automotriz, como coches autónomos.<br>-Informática, localización de correos y clasificación de articulos. | Toma decisiones óptimas razonando con probabilidades, las nuevas instancias se clasifican combinando hipótesis. | Elevado coste computacional en el proceso de actualización de probabilidades, requiere conocer gran número de elementos. |
| Regresión ordinaria por mínimos cuadrados | Utiliza el método matemático llamado regresión lineal con la estrategia de mínimos cuadrados ordinarios, para estimar los valores que no se conocen. | -Medicina, como el reconocimiento de patrones de comportamiento (Pacheco & Díaz, 2005).<br>-Climatología, análisis de los patrones del comportamiento del clima. | Permite realizar predicciones a partir de datos parciales, lo que le hace un algoritmo de funcionamiento rápido. | No es apto para el manejo de variables aleatorias continuas. |
| Regresión logística | Manera estadística de representar un resultado binomial con variables explicativas, estima la probabilidad de ocurrencia de un suceso. | -Medicina, identificación de variables endocrinólogas (Domínguez & Padilla, 2001).<br>-Académico, reconocimiento de patrones de rendimiento estudiantil (Reyes, Escobar, Duarte, & Ramírez, 2007). | Se utiliza en escenarios de grandes volúmenes de datos por su eficiencia, además el resultado del modelo se puede interpretar como una probabilidad. | Resultados satisfactorios en caso de que los atributos y clases sean linealmente separables. |
| Support Vector Machines (SVM) | Es de clasificación binaria, con dos grupos de distintos tipos, el SVM buscará la manera de separarlos mediante líneas rectas estando lo más lejos posible entre ellos. | Reconocimiento de sitios de empalme humano, detección de genero basado en imágenes y clasificación de imágenes a gran escala. | Predice el grupo al que pertenece una nueva entrada, basándose en una sola variable. Permite minimizar el error en la separación, utilizando pocos parámetros a estimar. | Es un clasificador binario y no está diseñado para identificar los atributos importantes que construyen la regla discriminante. |

Los algoritmos descritos en la Figura previa son técnicas de ML, las cuales han tenido buenos resultados, sin embargo, la evolución de las redes neuronales, han permitido una mejoría en el aprendizaje automático el cual se explica a continuación.

## ▪ 2.1.4 Redes neuronales

El modelo de trabajo de una red neuronal interconectada se ejecuta para transmitir información. Las redes deben estar organizadas en capas que forman un gráfico cíclico, es decir, la salida busca asemejarse a la entrada.

La **Figura 3** muestra un ejemplo llamado redes totalmente conectadas. El plano o capa input (rosado), representa diferentes neuronas, donde cada una representa un dato o pixel de entrada, el cual se interconecta con una hidden layer o capa oculta (color azul), modificando los pesos para obtener un valor semejante a la salida esperada. Durante este proceso se extraen propiedades del valor de entrada automáticamente [50].

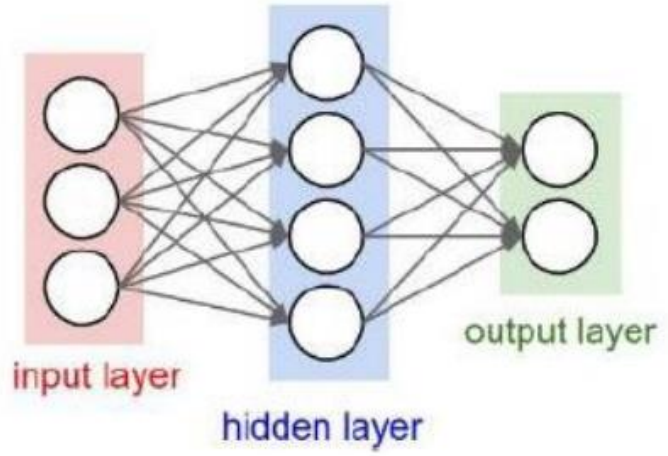



**Figura 3 Red neuronal con 3 neuronas de entrada, 4 ocultas y 2 de salida**

En la **Figura 4,** explica brevemente el funcionamiento matemático de la red neuronal. En este ejemplo, se ejemplifica una red neuronal con dos entradas bautizadas X1, X2, y un b o bias con valor igual a 1. Estas entradas no son modificables, lo modificable son los pesos W, los cuales son valores aleatorios que multiplican cada entrada, es decir, el valor recibido en la neurona X1, se multiplica con el peso W1, X2 con W2, X3 con W3, b con W0 etc.

Estos valores (multiplicación de entradas y pesos) se suman, obteniendo un valor o sumatorio total igual a Z. Una vez se obtiene esta salida $Z$, entra en una función de activación. Las funciones de activación calculan un valor según el dato requerido (valores de 0 a 1, valores de -1 a 1 o valores positivos, variando según el tipo de función de activación), como las funciones Relu, Sigmoidea, tangencial entre otras. En la figura se ejemplifica una función de activación clásica, conocida como la sigmoidea, donde el exponente de e, es $Z$, es decir el valor hallado en la sumatoria de las entradas por los pesos descritos anteriormente. El resultado de esta función de activación se bautiza como sigma, el cual entra en la función RMS (error cuadrático medio), el cual busca reducir el error de la salida deseada entre la salida obtenida, la cual se desarrolla elevándose al cuadrado y dividiéndose entre 2, a esto se bautizará como E de error. Este proceso de cadena hacia adelante se conoce como forward.

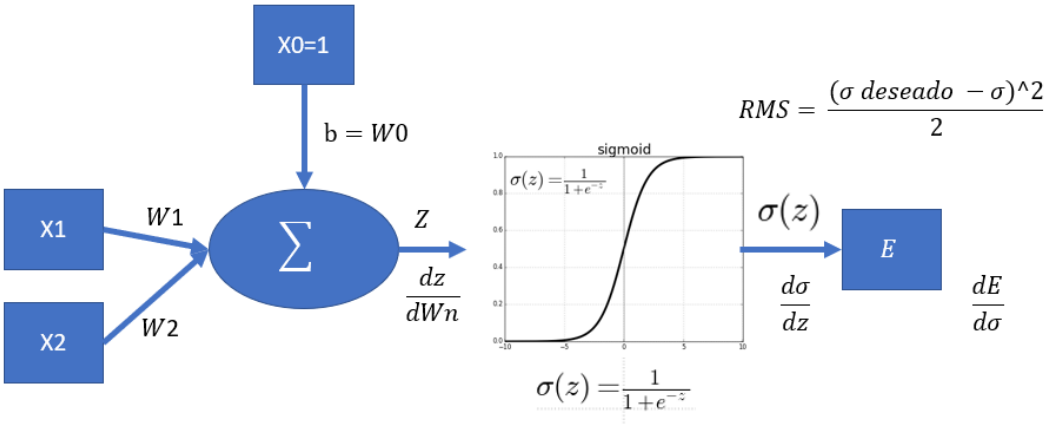



**Figura 4 Esquemático de red neuronal. Fuente propia**

El error $E$ es la salida, la cual no sirve de nada en la primera iteración, este dato debe iterar varias veces hasta alcanzar el mínimo utilizando, el gradiente descendente estocástico. Estas iteraciones se logran a través de derivadas parciales con la regla de la cadena, conocido como la propagación hacia atrás del inglés back propagation. Para obtener el gradiente descendente, se debe comparar el $E$ frente a los pesos Wn (que multiplicaban las entradas), la cual se obtiene a través de la regla de la cadena dada en la **Ecuación (1)**

$$\frac{dE}{dWn} = \frac{dE}{d\sigma}\frac{d\sigma}{dz}\frac{dz}{dWn} \qquad (1)$$

Una vez se obtiene el gradiente descendente $\frac{dE}{dWn}$, se debe multiplicar por un valor alfa $\alpha$, bautizado como Learning Rate o factor de aprendizaje, el cual multiplica el gradiente para que encuentre el mínimo ($\alpha$ debe ir de 0 a 1, entre más se acerque a 0 más preciso será, sin embargo, tendrá un mayor costos computacional y tardará más tiempo encontrando el mínimo, en contraparte, si el número es mayor, puede que encuentre el valor mínimo más rápidamente, o nunca llegue a conseguirlo al obtener saltos demasiado grandes). Al resultado cómo se evidencia en la **Ecuación (2)**, se resta a $Wn$ (peso de entrada), para de esta forma obtener un nuevo peso $Wn^{+1}$. Esta operación iterará tantas veces como considere el diseñador, teóricamente supone que, entre más iteraciones o épocas, mejorará la precisión de aprendizaje.

$$Wn^{+1} = Wn - \alpha \frac{dE}{dWn} \qquad (2)$$

En resumen, dentro de cada neurona podemos decir que sucede una regresión lineal donde los pesos le dan la inclinación a la pendiente y el bias la traslación de esta, y se parametriza según la función de activación sea escalón, rampa, o sigmoide. Posteriormente, se busca el mínimo global por medio del Gradiente Descendente Estocástico con Momento que no es más que un cálculo de derivadas parciales, el cual se utiliza con el conocido Back Propagation para reducir el consumo de máquina, el cual al multiplicarse con una tasa de aprendizaje y restándose con el valor deseado aprenderá las características del objeto de entrada. En la **Tabla 2**, se describen algunas de sus aplicaciones.

**Tabla 2 Redes Neuronales fuente: Pasquel [38].**

| Algoritmo | Descripción / Función | Aplicaciones | Ventajas | Desventajas |
|---|---|---|---|---|
| Redes neuronales | Simulan el comportamiento de un cerebro humano, conectando varios procesadores entre sí. | -Telecomunicaciones, reconocimiento de patrones, filtrado de señales, reconocimiento de caracteres (Galán & Martínez, 2015).<br>-Robótica, identificación de movimientos (Moya, Herrero, & Guerrero, 1998).<br>-Psicología, identificación de características para diagnóstico (Montaño, 2002). | Son capaces de aprender de la experiencia, crea su propia organización y tiene un bajo porcentaje de fallos o errores. | Complejidad de aprendizaje para grandes tareas, por su gran cantidad de datos, requieren de una gran capacidad de hardware. |

Una red neuronal convolucional o convnet, difiere al proceso de una red neuronal particular, por el gran número de capas, las cuales van de decenas, a centenas, e internamente, realizan procesos que reducen la imagen, para optimizar costo computacional [37].

Las ConvNets o redes neuronales convolucionales (CNN), constan de varias etapas donde la entrada y la salida representa una serie de arreglos llamados "mapas de características", es decir operan por pixeles, empleando múltiples filtros, funciones de activación lineales, capas de normalización, pooling, redes neuronales totalmente conectadas como vector columna que será evaluado con el criterio de una red neuronal [51] y clasificado en la capa Softmax.

### ▪ 2.1.5 Redes Neuronales Convolucionales (CNN)

La **Figura 5** obtenida de Mathworks@, evidencia gráficamente la arquitectura de una CNN, y sus partes. El proceso general, inicia con una imagen de entrada que se lee en pixeles, seguido por una convolución o filtro, el cuan es la multiplicación entre arreglos, para obtener bordes, diagonales o imágenes más complejas automáticamente. El pooling, reduce la cantidad de pixeles obteniendo las características relevantes. Posteriormente, la función de activación Relu obtiene los valores positivos, obtenidos en la convolución o filtro previo. El

proceso se realiza reiterativamente en varias capas hasta llegar en forma de vector columna. Las capas flatten y fully conected, se interconectan como neuronas buscando el error y actualizando los pesos con el gradiente descendente [52], para al finalizar a la salida se obtengan las clases deseadas en la función Softmax. El proceso de aprendizaje de una CNN extrae filtros (bordes, diagonales, verticales) automáticamente, aprendiendo las características de los componentes RGB en una imagen.

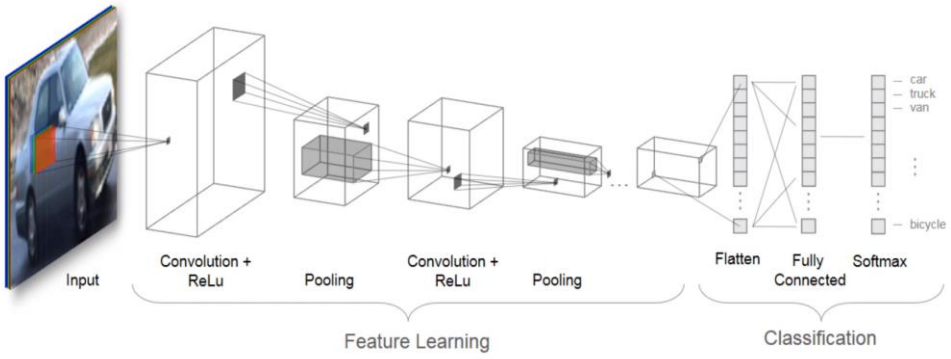



**Figura 5 Arquitectura de una CNN. Fuente: Mathworks [53]**

Los hiperparámetros se entienden como aquellas características que modifican el comportamiento del algoritmo de aprendizaje profundo y están especificados por el desarrollador.

La **Figura 8**, ejemplifica la convolución en un cuadro matricial, donde cada número representa un pixel en una imagen que va de 0 a 255, pasa por un filtro de 3x3 y se extrae así un solo número, el cual es el resultado de la sumatoria del producto punto visto y explicado en la **Ecuación 3.**

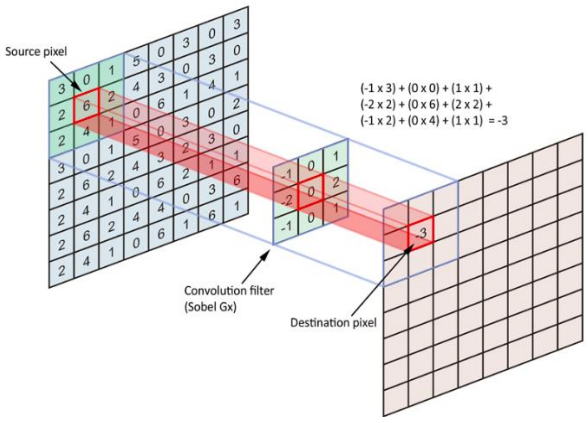



**Figura 6 Operación de convolución. Fuente: Mathworks [54]**

$$S(i,j) = \sum_m \sum_n I(m.n)K(i-m, j-n) \quad (3)$$

**Dimensiones del filtro (F):**

El filtro kernel o capa de convolución es utilizado para la extracción de información en una imagen de entrada obteniendo como salida los caracteres o feautures principales. Por ejemplo, una capa de convolución de bordes verticales o de bordes horizontales se pueden ver respectivamente de la **forma (4) y (5)** a continuación.

$$\text{Capa detección bordes verticales} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \quad (4)$$

$$\text{Capa de bordes horizontales} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad (5)$$

Multiplicando esta matriz por las dimensiones semejantes a la imagen de entrada, se logra obtener información por secciones, la cual se va arrastrando por pixeles, siendo este resultado, aquel que continua el proceso de pooling, y función de activación hasta terminar las capas de la CNN.

**Convolución:** Las capas de las convoluciones, combinan parámetros que permiten el aprendizaje de patrones [52] para categorizarlas correctamente [51] como se aprecia en la **Ecuación 6** del documento.

$$s(t) = \int x(a)w(t-a)\,da \quad (6)$$

**Padding:**

El Padding o relleno con pixeles negros al rededor, permite el aprendizaje de bordes para redimensionar la imagen y evitar errores de aprendizaje. Se recomienda la utilización de imágenes con bordes negros lo suficientemente grandes, para que la salida sea igual en dimensiones que la entrada y así evitar problemas de aprendizaje en la convolución como se aprecia en la **Ecuación 7**, siendo $W$ el ancho o vector fila de la imagen, $P$ el Padding o cantidad de pixeles negros que se agregaran para hacer el relleno de una imagen y $F$ el ancho o vector fila del filtro, sumando un uno se obtiene:

$$\underline{(W + 2P - F) + 1} \qquad (7)$$

El resultado se halla de la misma forma para obtener a la altura de la salida. Tomando como ejemplo una imagen de 6x6, e implementando un filtro (F) de 3x3, se ejemplifica como $((6 + 2(0) - 3) + 1)$, obteniéndose como salida, una imagen de dimensiones 4x4 como resultado.

**Stride:** Representa el salto en píxeles del filtro tanto en alto (H) como ancho (W). El filtro si no cuenta con stride, se desliza en la imagen pixel por pixel de izquierda a derecha, según mas grande sea el salto, mayor será la reducción del resultado de la convolución expresado en la **Ecuación 8**. Como ejemplo suponga una entrada $W * H$ =7 (Imagen 7x7 pixeles), $P$ =0 (sin relleno), F=3(filtro 3x3), y un S=2 (Stride o salto de dos) obteniendo a la salida una imagen después de la convolución de 3x3.

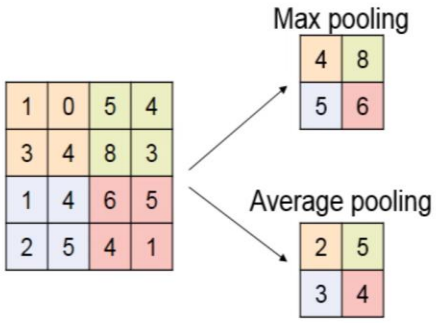
$$Weigh = \frac{(W + 2P - F)}{S} + 1 \; x \; High = \frac{(H + 2P - F)}{S} + 1 \qquad (8)$$

$$Weigh = \frac{(7 + 2 * 0 - 3)}{2} + 1 \; x \; High = \frac{(7 + 2 * 0 - 3)}{2} + 1 = 3X3$$

El stride es utilizado para reducir la imagen.

**Pooling**. El Max pooling y average pooling, son usados en el estado del arte para reducir las imágenes y extraer el feauture relevante. Tomando como ejemplo la aplicación de un

Max pooling, se extraen los pixeles de valor mayor en secciones de 2x2, a una imagen de 16 pixeles (4x4), obteniendo como salida una nueva de 4 pixeles (2x2) vista en la **Figura 7**. Se evidencia que la imagen se reduce 4 veces [55] a la imagen anterior, extrayendo así, la característica principal del filtro [56].




$$f(x) = \text{Max}\ (xi, j, d)$$

**Figura 7 Max Pooling y Average Pooling. Fuente: Mathworks**

En la **Tabla 3,** se describen tres (3) tipos de pooling, por máximo valor, por promedio o subsampling.

**Tabla 3 Función capa Pooling [57]**

| Tipo de pooling | Implementación | Descripción |
|---|---|---|
| Max-pooling | $\max_{rxr}(x_i)$ | Máximo de una región cuadrada de tamaño rxr. |
| Promedio (Average) o downsampling | $\text{mean}_{rxr}(x_i)$ | Promedio de una región cuadrada de tamaño rxr. |
| Subsampling | $\tanh(\beta \sum_{rxr} \frac{x_i}{r^2} + b)$ | Promedio de la entrada de una región cuadrada de tamaño rxr y multiplicado por un escalar entrenable (β) y sumado un bias entrenable (b), calculando el resultado a través de una función no lineal. |

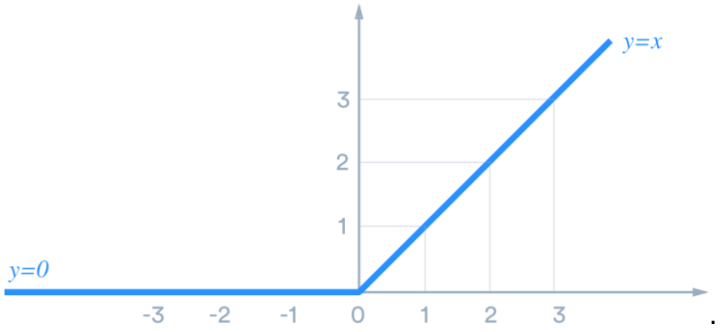
**Profundidad:** Son los canales de la disposición de entrada, una imagen en grises tiene un canal, un RGB tiene tres canales.

Acá se explica que para tratar imágenes de 3 canales o 3D, se utiliza un filtro por cada canal con la misma dimensión de alto y ancho, y al final se obtiene uno bidimensional. Si se quieren obtener más filtros simplemente se representará como FxC, siendo F el vector de

salida por ejemplo 4x4 y C la cantidad de filtros, en este caso C= 3 por los componentes RGB.

Como ejemplo de entrada, una imagen de 39x39x3 considerando la formula ((n+2p-f) /s) +1 se implementan 10 filtros de 3x3 (f=3), stride(s) =1 y sin relleno (p=0), obteniendo ((39+0-3) /1) +1, obteniendo una imagen de 37x37*10, es decir más pequeña, pero con más profundad, en este caso, diez canales. Se puede repetir el proceso con más convoluciones hasta obtener una imagen de 7x7x40, esto es igual a 1960 datos, los cuales se resumen en un vector columna en la fully connected, para aplicar una regresión logística o función Softmax [1].

**Rectified Linear Unit (Relu):** Unidad de rectificación lineal o RELU, es la función de activación que obtiene los números positivos, los números negativos obtenidos en el proceso de convolución y pooling se vuelven cero [58]. Al aplicarse convoluciones o multiplicar por filtros, los datos recibidos pueden ser negativos o positivos, siendo la función Relu, aquella que simplifica el proceso remplazando los pixeles negativos visto en la **Figura 11**.




$$f(x) = \text{Max}\,(0, x\ i, j, d)$$

**Figura 8 Rectified Linear Unit (Relu). Fuente: TinyMind**

**Normalización:** En la normalización se utilizan dos técnicas para evitar confundir la red Batch Normalization [59] y el Group Normalization [58] para corregir el desplazamiento covariable interno.

**Fully Connected** Se considera como una red neuronal convencional que interconecta las salidas con capas ocultas y genera unas salidas deseadas. Se suele utilizar capas

completamente conectadas en la que cada pixel se considera como una neurona separada al igual que en un perceptrón multicapa.

**Dropout:** En el aprendizaje puede ocurrir un over fitting o sobre ajuste el cual genera problemas cuando se entrena una red, ya que aprende perfectamente unos parámetros específicos, pero al validar con información nueva, es incapaz de reconocerla ya que prácticamente ha memorizado las entradas de entrenamiento, por eso desconecta aleatoriamente neuronas disminuyendo el sobre ajuste para que reconozca ahora si entradas similares sin problema [50].

**Softmax:** Esta última capa, evalúa la cantidad de salidas, siendo la sumatoria de las probabilidades de salida de clasificación igual a 1. por ejemplo, una moto puede considerarse semejante a un móvil de dos ruedas, sin embargo, propiedades de la moto hace que difiera de la bicicleta o el monopatín [60], como lo representa la **Ecuación 9** a continuación.

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}} \qquad (9)$$

En la **Figura 12**, se muestran los hiperparámetros mencionados anteriormente, donde H es la altura y el ancho W de la imagen, F el tamaño o dimensión del filtro, S es el salto o stride, P el relleno o Padding y C o D representa el Depth, profundidad o número de canales [12].

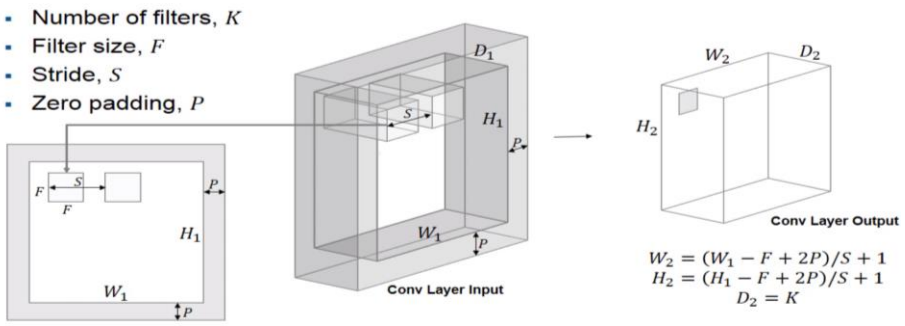



**Figura 9 Hiperparámetros de la operación de las capas de las CNN. [53]**

**Función de perdida:** La función de pérdida o Loss Function caracterizan la similitud de una etiqueta con algunas características previa. Es decir, la función de perdida es el resultado de la interacción de una salida vs una entrada en una red neuronal con las variables peso y bias, obteniendo un margen de error, restando estos dos. Este margen de error se lo realiza tantas veces sea el entrenamiento de la red, de este error la red aprende y actualiza los pesos y las ganancias de cada neurona. Entre más cerca esté de cero este error, significa que menos trabajo le costara a la red entrenada identificar el objeto de entrada.

**Red neuronal RESNET**

El entreno de las CNN muy profundas se vuelve más lento y complejo en conseguir el valor optimo utilizando el Gradiente.

En teoría una red al ser más profunda y recibir una mayor cantidad de datos, mejorará la precisión en la salida sin embargo en la práctica, al recibir una cantidad de datos mayor, la red empieza a perder precisión y a sobrentrenarse. La Resnet tiene la facultad de evitar este problema gracias a sus capas residuales.

Gracias a las convoluciones 1x1, se puede lograr reducir el cálculo ya que esta convolución es capaz de tomar solamente los filtros que se consideren con mayor relevancia en el proceso. Por ejemplo, una capa de 28x28x192, (nH 28, nW28, nC 192), es decir con 192 filtros, se le aplica una Conv de 1x1x192, junto a la Relu, obteniendo una no linealidad sin los números negativos, y a la salida obtener por ejemplo 28x28x32, es decir la capa en ancho y alto de las mismas dimensiones, pero con menos filtros.

La Resnet omite conexiones, lo cual le permite tomar la activación de una capa y alimentar a otra más profunda de la red. La Resnet se construye a partir de un bloque residual. La función de activación lineal va de acuerdo a la **Ecuación** (10), donde W (Es la matriz de peso) y b es (un vector de sesgo).

$$Z^{[l+1]} = W^{[l+1]} * a^{[l]} + b^{[l+1]} s \qquad (10)$$

Luego se aplica la función Relu la cual es a^[l+1] = g(z^[l+1]). Posteriormente se repite el proceso lineal a un nivel mayor siendo Z^[l+2]= W^[l+2]*a^[l+1]+b^[l+2] y de nuevo la función Relu expresada como a^[l+2] = g(z^[l+2]). Lo que busca esta CNN, es tomar atajos (short cuts) y omitir conexiones, igualando la entrada a la segunda capa quedando siempre la

función lineal antes de la Relu, es decir, la función obtenida al finalizar se expresa en la **Ecuación** (11).

$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]}). \quad (11)$$

La red incepción, evalúa si es mejor utilizar pooling, un tamaño específico del filtro, o todos a la vez. En este proceso se pueden obtener varios filtros con las mismas dimensiones de la capa. por ejemplo, en una capa de 28x28x192, se le aplica una convolución de 1x1x64, otra de 3x3x128, otra de 5x5x 32 y un pooling, obteniendo a la salida una capa de 28x28x256, la cual contiene varios filtros. Para lograr que los parámetros sean equidistantes (es decir de 28x28), se aplica Padding o relleno. Hacer una operación de este tipo, se requiere de casi 120M de operaciones, a diferencia del uso de convoluciones de 1x1, lo cual reduce 10 veces el cálculo. Concluyendo la Resnet es una CNN muy profunda, liviana y potente, capaz de aprender varias características gracias a su arquitectura residual, utilizando convoluciones de 1x1 para reducir la cantidad excesiva de filtros.

### ▪ 2.1.7 Detección de Objetos

Para hablar de reconocimiento de objetos basados en regiones, se requiere conocer el funcionamiento de localización de objetos, antes de entender qué es la detección de objetos.

Para la creación de un algoritmo de asistente a la conducción, no basta solamente con clasificar una imagen, ya que, en un entorno dinámico lleno de automóviles, personas, señales de tránsito, entre otros, no sería suficiente la clasificación de una categoría general por imagen, por esto se requiere conocer el proceso de localización, es decir que en una imagen se encierre y clasifique en un cuadro delimitador o bounding box, el objeto detectado. Como se presenta en la **Figura 10**, el cuadro delimitador cuenta con una altura bh y un ancho bw, el cual se ubicará según una coordenada del punto bx y by, el cual irá en el centroide del recuadro (punto amarillo). Los números (bx, by, bh y bw, son nuevas salidas de la red neuronal). Por otro lado, existe otro número PC, el cual define con un 0 si no hay un objeto y 1 si lo hay, el cual se obtiene a través del error cuadrático medio. También existirá un numero c1, c2, cn, uno por cada clase seleccionada, sin embargo, ninguno de estos cálculos tiene sentido si el PC es cero (0).

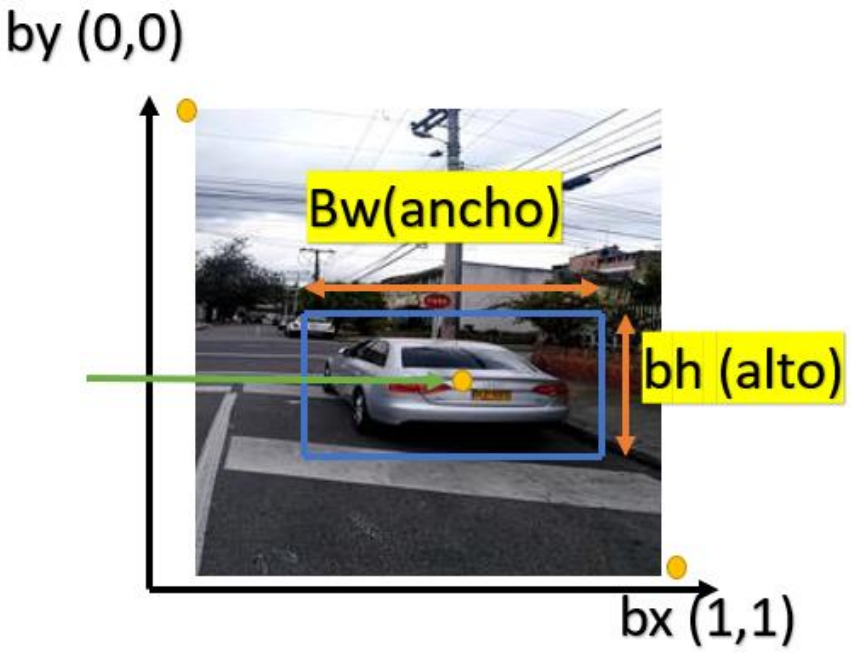



**Figura 10 Localización de imagen. Autoría propia**

- ## 2.1.7 Red neuronal YOLO (You Only Look Once)

La arquitectura YOLOv2 (You Only Look Once), se inspiró en la visión humana en cómo los ojos pueden detectar objetos de un vistazo en su traducción del inglés (usted ve una sola vez), con una regresión pura capaz de predecir directamente los cuadros delimitadores, clasificar y entregar un score o puntaje de precisión, utilizando una única CNN [38].

En este trabajo de aplicación, se implementó la arquitectura YOLO debido a que presenta un mejor comportamiento en velocidad y precisión detectando objetos, frente a otras arquitecturas del estado del arte como la Faster RCNN.

Esta arquitectura como se observa en la **Figura 11**, separa las imágenes por regiones, y predice en cada recuadro si se encuentra o no algún objeto de interés basado en una probabilidad mayor al 50%**.**

El algoritmo extrae por si solo los filtros necesarios de cada objeto de interés disminuyendo considerablemente el error de detección para nuevas entradas, es decir, imágenes diferentes a las utilizadas en el entrenamiento.

Este algoritmo ha sido probado en una GPU Titan X [91], detectando a una velocidad de 45 cuadros por segundo (FPS) sin mini Batch, que para el presente trabajo tarjeta, se realiza con una tarjeta gráfica NVidia 920 mx, capaz de procesar aproximadamente 10 (FPS), siendo un algoritmo aceptable para aplicaciones en tiempo real.

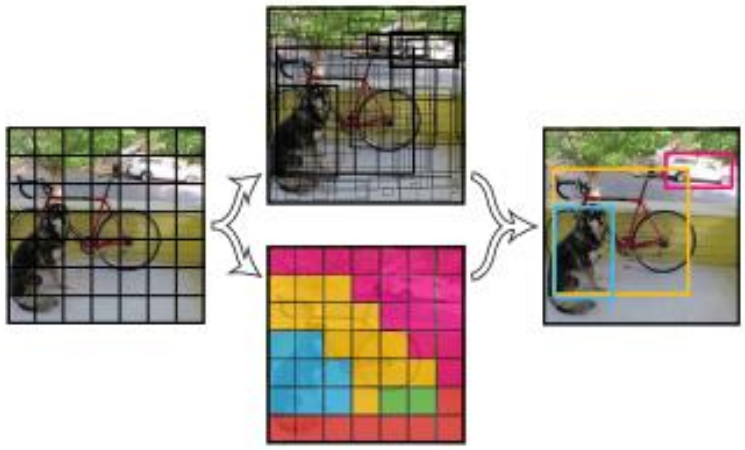


**Figura 11 Arquitectura YOLO, a una malla de 7x7 [61]**

Esta arquitectura divide la imagen en una de malla de 7x7, tiene 24 convoluciones [61], con dos capas de conexión totalmente conectadas, vista gráficamente en la **Figura 12**. La arquitectura YOLO sirve para aplicaciones a tiempo real, utilizando una única CNN encargada de caracterizar los objetos en imágenes por regiones. En lugar de los módulos iniciales propuestos por GoogleNet, YOLOv2 utiliza capas de reducción con filtros de 1×1 seguidos de capas convolucionales de 3×3 para obtener formas, contornos, texturas, colores, entre otros, [61]. Posteriormente dos capas totalmente conectadas capaces de encontrar la probabilidad del cuadro delimitador en ubicación por coordenadas y el tamaño de un recuadro que encierra la detección (x, y, w, h), siendo la normalización por lote, capaz de regularizar el modelo y eliminar el sobreajuste en un mapa de características de 13×13. Si bien esto es suficiente para objetos grandes, debe beneficiarse de funciones de grano más fino para localizar objetos más pequeños [98].

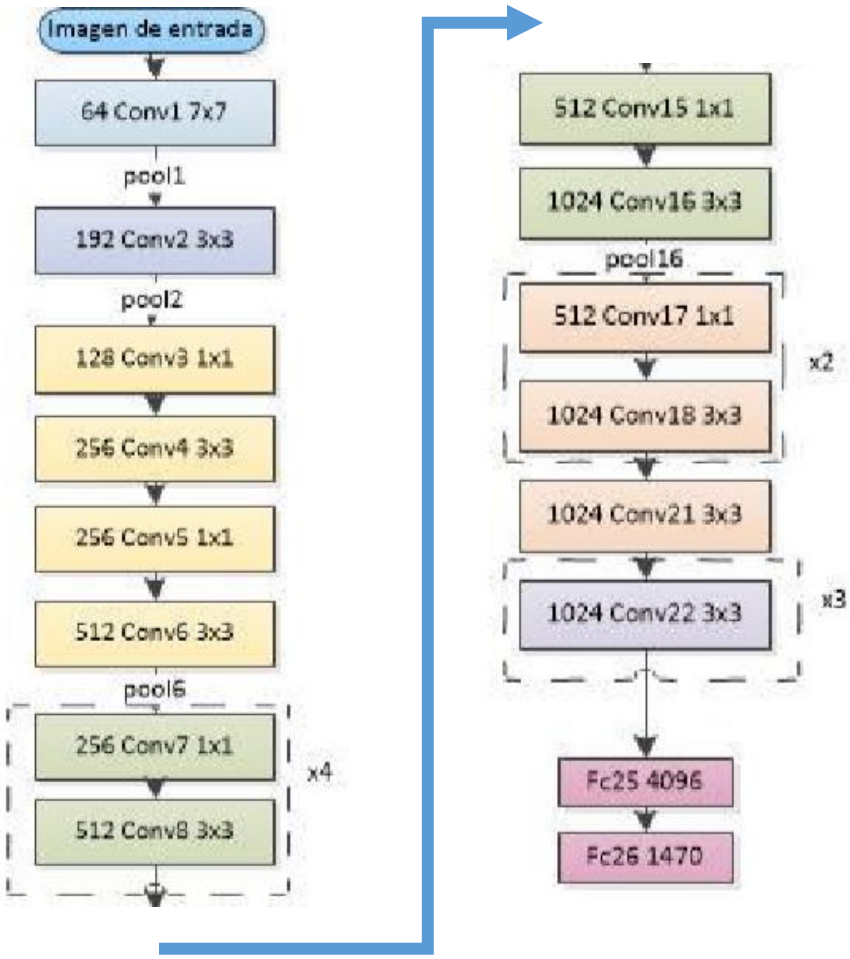



**Figura 12 Arquitectura (YOLO) [38]**

La evolución para el uso de esta arquitectura requirió el desarrollo de los LandMark detection conocidas también como la detección por etiquetas de puntos. Se intentó el estudio por ventanas deslizantes, donde un recuadro pasaba cuadro por cuadro cómo funcionan las convoluciones deslizándose por ventanas en filas, sin embargo, eran muy lentas, reduciéndose al análisis del uso de una sola convolución.

En el proceso de búsqueda por regiones de interés o bounding boxes, se analiza en un vector columna, donde ubicar los parámetros ver **Figura 13**, entre ellos la primera fila se ubica un 1 si hay un objeto de interés o un 0 si no lo hay, en este caso encontró un perro, en la segunda fila bx representa un punto en las coordenadas de x del objeto de interés y

by de las coordenadas de y respectivamente, ubicándose en el centro de masa del cuadrado que encierra el objeto de interés. bw la distancia en x que ocupa la imagen conocido en geometría como la base y bh, la distancia en Y o altura del cuadrado, finalmente se agregan dos espacios los cuales representan la cantidad de objetos de interés (perros y bicicletas) agregándose un 1 o 0 si está o no respectivamente. En este caso aplica la entropía binaria, para más de dos clases aplica la categoría de entropía cruzada**.**

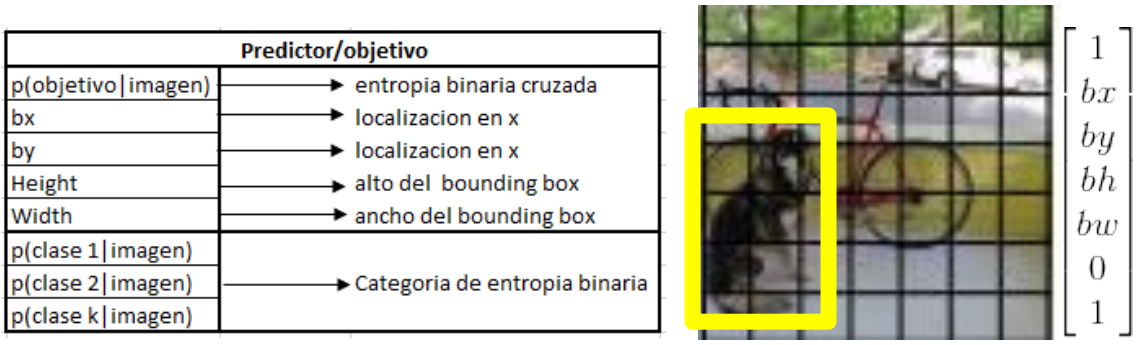



**Figura 13 Función de perdida Loss function Autoría: Propia**

Finalmente es importante identificar los Anchors, los cuales son utilizados para que la arquitectura las dimensiones del ROI, evitando confusiones a pesar de que un OI, esté cubriendo a otro como se evidencia la **Figura 14**.

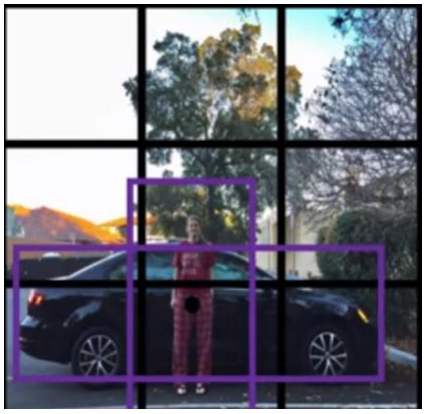


**Figura 14 Parametrización de Anchors persona y carro. Fuente Deep Learning COURSERA.ORG**

Dividiendo la imagen en una malla, donde existen varios OI, se genera una intersección sobre la unión, calculando el área interceptada sobre el total donde se obtenga un valor superior al 50% como parámetro de búsqueda del OI, habrá un recuadro dentro de la imagen procesada. Este proceso, se conoce como IOU (Intersección Sobre Unión), junto a

la supresión máxima, que considera el mayor porcentaje de aceptación y quita las que estén cerca con menor probabilidad como se ve en la Figura 17 y 18. Formalmente se define la confianza como Pr (Probabilidad de Objeto) * IOU, donde, si no existe ningún objeto en esa celda, el puntaje de confianza debería ser cero.

La superposición se calcula según las coordenadas de intersección de los recuadros delimitadores del detector, tanto de la parte que corresponde al alto (H) **Ecuación 12**, como el ancho (W) **Ecuación 13**, siendo el área total de la intercepción la **Ecuación 14**.

$$H = \min(box1[y2], box2[y2]) - \max(box[y1], box2[y1]) \quad (12)$$

$$W = \min(box1[x2], box2[x2]) - \max(box[x1], box2[x1]) \quad (13)$$

$$\text{Área de Intersección} = H * W \quad (14)$$

Por otro lado, el área de unión se obtiene de la suma de las dos áreas de los cuadros delimitadores, restando la intersección, ver **Ecuación 15**:

$$\text{Área de unión} = \text{áreaBox1} + \text{áreaBox2} - \text{área de intersección} \quad (15)$$

El IOU calculado del detector debe ser mayor o igual al valor utilizado de umbral IOU (0,5) para que el cuadro delimitador sea graficado y visualizado en la imagen de salida. El cálculo del IOU (intercepción sobre el objeto) del detector se realiza mediante la **Ecuación 16**

$$IoU = \frac{Área\ de\ Intersección}{Area\ de\ Unión} \quad (16)$$

La salida YOLOv2 consta en predicciones como se muestra en la **Figura 15** de clase C donde, a partir de la grilla W x H, se tiene B cajas delimitadoras en cada una de las grillas y cada caja tiene 6 predicciones, es por ello que el vector de salida es W x H x (B * 6 + C) [38].

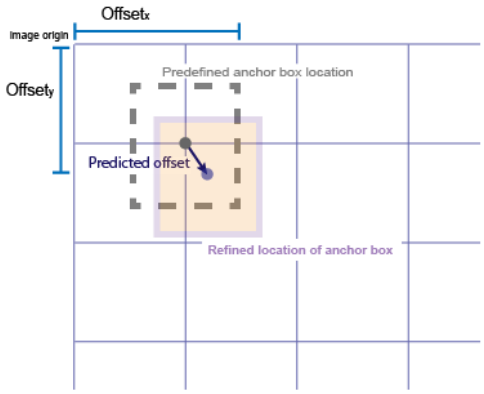



**Figura 15 Intercepción sobre la unión (IOU) Fuente Mathworks [135]**

El modelo YOLOv2, se utilizó para el data set de PASCAL VOC. Las primeras capas con el uso de convoluciones y Pooling, extraen los filtros para al finalizar predecir con técnicas de probabilidad IOU la ubicación del objeto.

En el entrenamiento se utilizan 20 capas convolucionales y una capa que promedia grupos; Para implementarla se utilizan 4 capas de convolución y dos capas de conexión completa, analizando el error cuadrático medio para con el valor mínimo alcanzar a predecir los objetos de interés [62].

El presente trabajo propone la utilización de la arquitectura de red neuronal convolucional YOLOv2, el cual permite mediante detectar seis OI, entre ellos señales de pare, semáforos, personas, carros, motos y bicicletas lo cual es novedoso considerando su uso para implementarlo como asistente a la conducción en un automóvil. En la **Figura 16** se adjunta un esquemático del funcionamiento de la YOLO.

En resumen, Para la detección se tienen en cuenta principalmente tres caracteres:

- Intersection over union (IOU) — Predecir la puntuación de objetividad de cada cuadro de anclaje.
- Anchor box offsets — Refinar la posición de la caja de anclaje
- Class probability — Predecir la etiqueta de clase asignada a cada cuadro de anclaje.

En conclusión, la estructura de la YOLO (ver Figura 19), es una arquitectura capaz detectar y clasificar varios OI en un ROI con una sola CNN lo que permite aplicabilidad para el presente trabajo. Las CNN como algoritmos de DL, demostraron poder actualizar los filtros a través de técnicas de convoluciones permitiendo actualizar los pesos automáticamente como lo haría una red neuronal multicapa. La arquitectura YOLO, procesa las imágenes de entrada a través de una grilla que genera cuadros delimitadores con bounding boxes, identificando según el mapa de probabilidad. La única CNN (en este caso GoogleNet), genera un score según el label o etiqueta de la imagen clasificando y detectando los OI seleccionados.

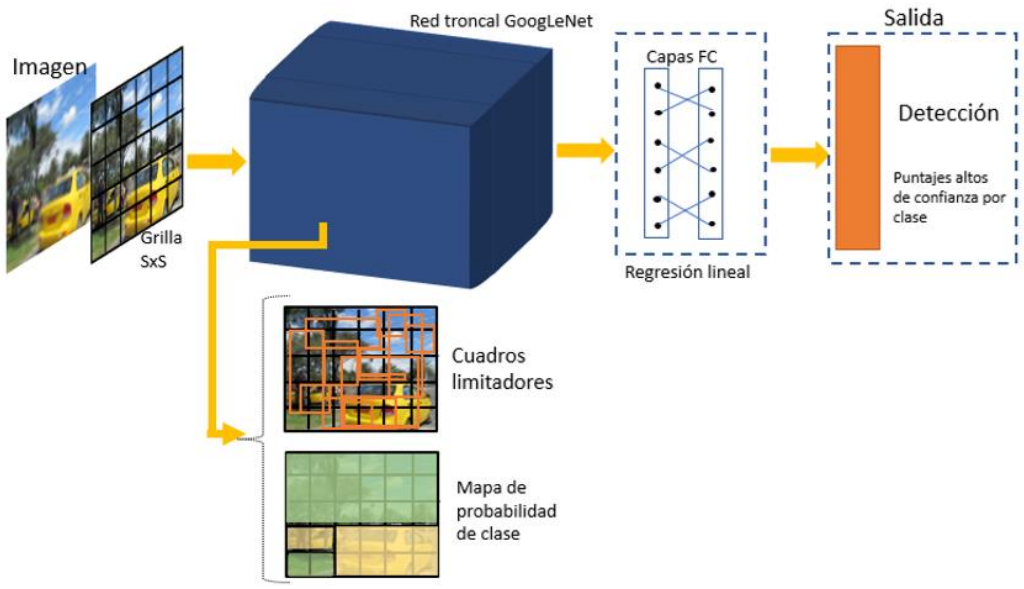



**Figura 16 Estructura de red YOLO [38]**

## ○ 2.2 CARROS AUTONOMOS

La comprensión actual de autonomía en los automóviles, parte de identificar los niveles de autonomía en los CA y algunos de sus sensores.

### ▪ 2.2.1 Niveles de autonomía en un automóvil.

Existen seis (6) niveles de autonomía según la SAE [13] (Sociedad De Ingenieros Automotrices), los tres primeros son tripulados, los tres siguientes no requieren de una participación importante de un conductor. La autonomía en coches ha ido progresando considerablemente, aportando avances a la reducción potencial de accidentes y descongestionar el tráfico, considerando comodidad, seguridad y un rendimiento óptimo en combustible. Los departamentos que regulan la autonomía en conducción es la SAE y la NHTSA (Administración Nacional de Seguridad de Tráfico en Carreteras del Departamento de Transporte de EE. UU), siendo estos dos departamentos pioneros en clasificar los niveles de autonomía en CA. Cada nivel según la SAE se caracteriza por funciones más complejas, siendo a mayor nivel de autonomía, menos dependiente en intervención humana. Para transformar un vehículo normal en uno autónomo es crucial la sensórica utilizada, la cual va desde radares y cámaras, a escáneres y sensores LIDAR, siendo esto, un aspecto importante en la adquisición de datos con una visión amplia del entorno en el que se inmerge el automóvil y toma de decisiones [63].

Autonomía significa independencia o poder de autogobierno sobre sí mismo, considerando las ventajas y las desventajas en sucesos ocurridos por un CA, ¿a quién culpar en caso de falla o accidente, al automóvil o al estado, al tener deterioradas las vías y las señalizaciones de transito?

Se debe considerar con precaución la semántica al hablar de autonomía en el coche, ya que si bien, un automóvil totalmente autónomo debe ser capaz de realizar su propio mantenimiento en software y hardware, además de conducir de un punto A, a un punto B igual que lo haría un humano. Instituciones como la Organización Internacional de fabricantes de vehículos de motor (OICA), el Instituto Federal de Investigación de Carreteras de Alemania (BAST) y la Administración Nacional de Seguridad del Tráfico en las Carreteras de EE. UU (NHTSA) [63], intentaron describir los niveles de autonomía, difícil tarea al depender de diferentes fabricantes, requisitos del sistema, usos adecuados del sistema y nivel de automatización. Los 6 niveles de automatización se describen según la

SAE seis niveles, donde el nivel 0 no tiene ningún tipo de automatización, en el primer (1) y segundo (2) nivel existen DDT (Tareas Dinámicas de Conducción, por sus siglas en ingles). Los tres niveles superiores de automatización de conducción (3-5), se refieren a casos en los que el ADS (Sistema de conducción automatizado), realiza los DDT de forma automática [64]. En la **Figura 17**, se ilustran diferentes niveles del estándar J3016 con una descripción por cada nivel

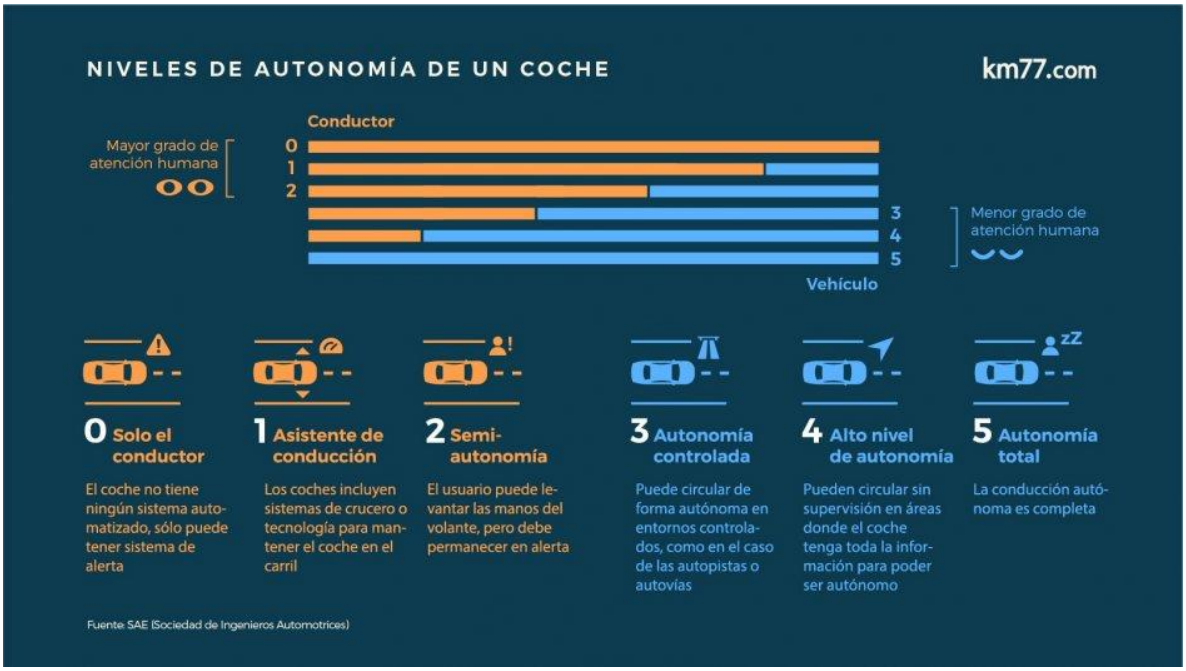



**Figura 17 Niveles de carros autónomos SAE (Sociedad de Ingenieros Automotrices) [13]**

### Niveles de automatización

**Nivel 0** (sin automatización), el conductor realiza toda la conducción DDT.

**Nivel 1** (asistencia al conductor) durante una condición específica, el sistema realiza el control de movimiento lateral o longitudinal pero nunca los dos simultáneamente, y el conductor realiza las tareas restantes del mismo.

las principales funcionalidades que el vehículo preferiblemente necesita tener son el control de crucero activo (ACC) y el Sistema de advertencia de abandono de carril (LDWS). El ACC es una tecnología que tiene la capacidad de mantener una velocidad predefinida y mantener una distancia segura frente a vehículos al frente [63]. El LDWS es la función que detecta el marcador de la línea en la carretera y llama con una alerta, advirtiendo al conductor de un cambio de carril involuntario. Para que estas tecnologías funcionen correctamente, algunos de los sensores necesarios serían sensores ultrasonido, un sensor radar de largo alcance

(LRR) o una cámara complementaria. Un sensor ultrasónico utiliza la propagación del sonido para detectar objetos y vehículos, el radar (detección de radio y rango) utiliza ondas electromagnéticas emitidas por una antena que reflejada en objetos que calcula con el tiempo de rebote, la distancia. Las cámaras para sonido envolvente se componen básicamente de cuatro a seis lentes de ojo de pez (esto significa cámaras de gran angular), colocadas alrededor del vehículo para generar una alineación geométrica y componer así la vista del conductor [65].

**Nivel 2** (conducción parcial) durante una condición el sistema realiza tanto el movimiento lateral como el longitudinal, donde el conductor supervise el sistema. El nivel dos de conducción parcial, tiene como funcionalidades el LKA (asistencia de mantenimiento de carril por sus siglas en ingles) y la PA (Asistencia de estacionamiento). El LKA en algunos casos puede verse como un complemento para el LDWS ya que el LKA toma medidas si el conductor no realiza ninguna maniobra para evitar el cambio involuntario de carril. En otro caso, el sistema LKA ayuda al automóvil a mantenerse en el curso de la carretera, evitando dejar el carril involuntariamente o chocar con objetos al costado del vehículo. Los sensores más comunes utilizados para implementar estas funcionalidades son los LRR (ultrasónicos de largo alcance), la cámara de sonido envolvente y ahora un radar de corto alcance (SRR) de 0,15 a 30 metros, aunque también existen de mediano alcance (MRR) de 1 a 100 metros, y de 10 a 250 metros para el LRR [63].

**Nivel 3** (automatización de conducción condicional): La conducción autónoma condicional, reúne las funciones de los dos niveles inferiores, sumado al Frenado automático de emergencia (AEB por sus siglas en ingles), el Monitoreo del conductor (DM)), Traffic Jam Assist o asistente de tráfico (TJA) [63]. El AEB es una característica que previene accidentes y siniestros automovilísticos, advirtiendo al conductor de un choque, frenando sistemáticamente. El DM monitorea al conductor mientras conduce el automóvil en la carretera. Si el conductor no está prestando atención y se detecta una emergencia, el sistema advierte mediante luces intermitentes o sonidos llevándolo a un estado con un mínimo riesgo. El TJA es una función ADAS que ayuda al conductor a conducir el vehículo en atascos si la función está habilitada, el automóvil puede frenar, arrancar, mantener una distancia segura y mantener el vehículo en el carril, todo esto sin la intervención del conductor.

**Nivel 4** Alta automatización de conducción durante una condición específica. El sistema realiza todo el DDT, sin esperar que el usuario responda para intervenir si se le solicita. En este nivel, la función realiza toda la operación de conducción en autopista.

**Nivel 5** (automatización de conducción completa) no existe una condición específica para que el sistema funcione, la función debe ser capaz de realizar la DDT sin intervención humana. En este nivel, el sistema es capaz de guiar el vehículo durante todo el viaje, independientemente de los puntos de inicio y finalización en las condiciones el tráfico y el clima [66].

### ▪ 2.2.2 Carros autónomos y sensores

Para los diferentes niveles de autonomía se requieren sensores capaces de atender parcialmente, a las necesidades del vehículo para visualizar el medio ambiente. Estas características funcionen de manera óptima, si el vehículo cuenta con los siguientes sensores: Una cámara de larga distancia, una cámara estéreo, sensor LIDAR e infrarrojos o térmicos. Una cámara estéreo se basa en la visión considerando la distancia, semejante a la naturaleza, ya que humanos y animales “mapean" con precisión los entornos 3D y miden parcialmente la distancia desde el punto de vista referencial a un objeto en el espacio.

**Caracterización de sensórica por coche autónomo**

En el mercado europeo y Norteamérica se comercializan automóviles con diferentes tipos de autonomía, entre ellos asistentes a la conducción, parcialmente autónomos, y autonomía controlada [13], los cuales han invertido innumerables esfuerzos en conseguir la autonomía total entre ellos BMW, Mercedes Benz, Nissan, Google, la General Motor, con una caracterización especial en sensores ilustrados en la **Tabla 4**.

Para el automóvil autónomo de Google serie Prius and Linux, se consideró un sensor Lidar capaz de detectar objetos a un rango de 360 grados, cámaras RGB que identifique colores, radares frontales y laterales, un enconder de posición en las ruedas y mapas que lo ubican satelitalmente. Es posible ver que el automóvil de Google es el más robusto al utilizar sensor LIDAR, aunque se comercialice en el mercado. Todas las compañías comparadas, usan

cámaras para detectar el medio ambiente, siendo uno de los sensores más importantes del vehículo [67].

**Tabla 4 Technology current used by some carmakers / Source: MIT Technology Review**





| | BMW | Mercedes-Benz | Nissan | Google | General Motors |
|---|---|---|---|---|---|
| VEHICLE | 5 Series (modified) | S 500 Intelligent Drive Research Vehicle | Leaf EV (modified) | Prius and Lexus (modified) | Cadillac SRX (modified) |
| KEY TECHNOLOGIES | • Video camera tracks lane markings and reads road signs<br>• Radar sensors detect objects ahead<br>• Side laser scanners<br>• Ultrasonic sensors<br>• Differential GPS<br>• Very accurate map | • Stereo camera sees objects ahead in 3-D<br>• Additional cameras read road signs and detect traffic lights<br>• Short- and long-range radar<br>• Infrared camera<br>• Ultrasonic sensors | • Front and side radar<br>• Camera<br>• Front, rear, and side laser scanners<br>• Four wide-angle cameras show the driver the car's surroundings | • LIDAR on the roof detects objects around the car in 3-D<br>• Camera helps detect objects<br>• Front and side radar<br>• Inertial measuring unit tracks position<br>• Wheel encoder tracks movement<br>• Very accurate map | • Several laser sensors<br>• Radar<br>• Differential GPS<br>• Cameras<br>• Very accurate map |

Mercedes Benz clase E [14], cuenta con un asistente a la conducción que lo hace un auto parcialmente autónomo contando con funcionalidades de frenado al reconocer situaciones de peligro, entre ellas:

- Advertencia de distancia y de colisión: Emite señales de iluminación o acústicas como advertencia al encontrarse muy próximo al objeto de interés delantero.
- Asistencia al frenado en función de la situación. Apoya la presión del frenado al percibir la reacción
- Frenado autónomo. Si el conductor no reacciona a pesar de las advertencias iniciará automáticamente un frenado autónomo.
- Función de frenado de emergencia ante retenciones: Si el sistema no detecta ninguna posibilidad de esquivar, la función de frenado de emergencia ante retenciones puede reducir automáticamente la velocidad.

Por otra parte, los automóviles Tesla, le han apostado al uso de la conducción autónoma sin sensores Lidar, considerado lo como barrera de entrada al ser de alto costo. Tesla

cuenta en sus coches 8 cámaras para obtener una visión 360 grados, divididas en dos laterales con alcance de 80 metros, tres frontales para percibir una vista de 120 grados y longitudinalmente a 250 metros, 3 con vista hacia atrás que permiten los adelantamientos y 12 sensores que complementan su “visión”. Entre ellos un radar que permite ver a través de zonas oscuras y con neblina de hasta 160 m, ultrasonidos laterales que reconocen autos cercanos y un computador con tarjeta gráfica capaz de procesar un gran número de información a tiempo real ver **Figura 18**. Cada cámara tiene una función específica, entre ellas están para crucero de control adaptativo, detección de peatones, freno de emergencia, asistente espejo y de parqueo. A pesar de los esfuerzos realizados aún hay algunos errores que solucionar, entre ellos las sombras producidas por los túneles percibidas como objetos, o el no poder adelantar un coche por la derecha sin contar con los problemas de reglamentación e infraestructura vial que varía en cada país.

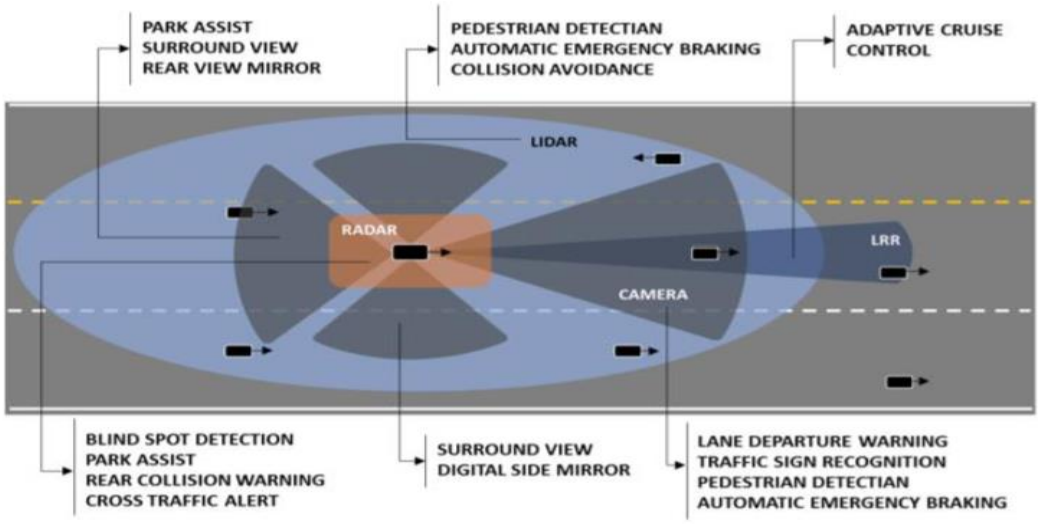



**Figura 18 Cubierta de sensores visión 360 grados [68]**

### Comparación de cámara mono y estéreo

Una cámara de video estéreo está compuesta por dos cámaras que, al colocarse a una distancia fija entre sí, es posible sintetizar y obtener la información RGB y estimar la

distancia. En comparación con un sistema mono cámara, las cámaras estéreo son más robustas y confiables para calcular al obtener datos de dos cámaras**. La Tabla 5** presenta una comparación de parámetros entre un sistema mono cámara y uno estéreo, algunos puntos destacados es ser más simple y más confiable que uno con cámara mono, pero exige más costo de máquina para procesar la imagen [69].

En conclusión, la SAE cuenta con (6) niveles de autonomía, para lo que, en este trabajo, se consideró el desarrollo del nivel dos debido a las características que lo relacionan, entre ellas, dar asistencia a la conducción, mantener un movimiento longitudinal, proponer un sistema con una cámara estéreo, capaz de reconocer la distancia de los OI, que se pueda implementar para un automóvil convencional en Colombia.

**Tabla 5 Comparativo entre mono cámaras vs cámara estéreo [122]**

| **Comparación mono cámara vs stereo cámara** | | |
|---|---|---|
| **Parámetros** | **Mono cámara** | **Stereo cámara** |
| **Numero de sensores** | 1 | 2 |
| **Dimensiones físicas** | Small(6''x4''x1'') | 2 montajes small separado a por 25-30cm |
| **Cuadros por segundo (FPS)** | De 30 a 60 FPS | 30 FPS |
| **Procesamiento de imagen** | Medio | Alto |
| **Confianza de detección** | Medio | Alto |
| **Sistema confiable para:** | Detectar peatones, trafico, señales y demás en RGB | Detecta imágenes RGB y además la profundidad de los OI |
| **Costo del sistema** | 1x | 1.5x |

## ○ 2.3 ANTECEDENTES

### ▪ 2.3.1 Aprendizaje Profundo

Las redes neuronales convolucionales empiezan a utilizarse en 1989 para reconocer dígitos de mano alzada [70] conocida en el concurso MINST (ver **Figura 19**) al ser capaz de detectar números del 0 al 9, obteniendo un error de clasificación del 3%. La evolución de las redes neuronales presenta nuevos avances en el año 2010 gracias a desarrollos de tarjetas gráficas capaces de hacer millones de cálculos en paralelo, ideales para el procesamiento de barrido de pixeles. Por ejemplo, algoritmo de DL, mostró 16.6% (ver **Figura 20**) de error al clasificar 10 clases de imágenes complejas en el conjunto de datos CIFAR-10 (este concurso demostró la capacidad de un algoritmo en reconocer imágenes complejas, entre ellas gatos, perros, pájaros y camiones por nombrar algunas categorías, [71], evidenciando una mejora considerable en la clasificación, lo que ha generado nuevos desarrollos en las redes neuronales convolucionales.

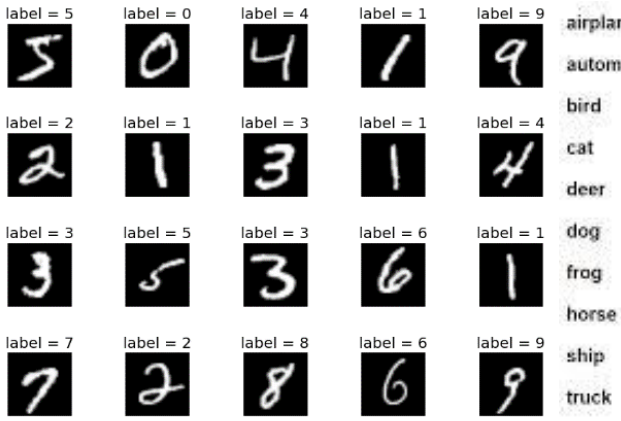



**Figura 19 Concurso de escritura MINST**


**Figura 20 Concurso CIFAR-10**

Gracias al concurso ImageNet, donde se pone aprueba el algoritmo con buen rendimiento en precisión y velocidad de identificación, obtuvo como ganador en 2012 la CNN bautizada Alex Net [12], capaz de identificar y clasificar 1000 categorías, obteniendo un 15,3% de error [72]. Aunque la AI (Inteligencia Artificial) tiene diversas ramas de trabajo, el **DL** ha presentado mejores resultados frente a técnicas clásicas de **ML** (Machine Learning) [54] disminuyendo progresivamente el error medio, ver **Figura 21**.

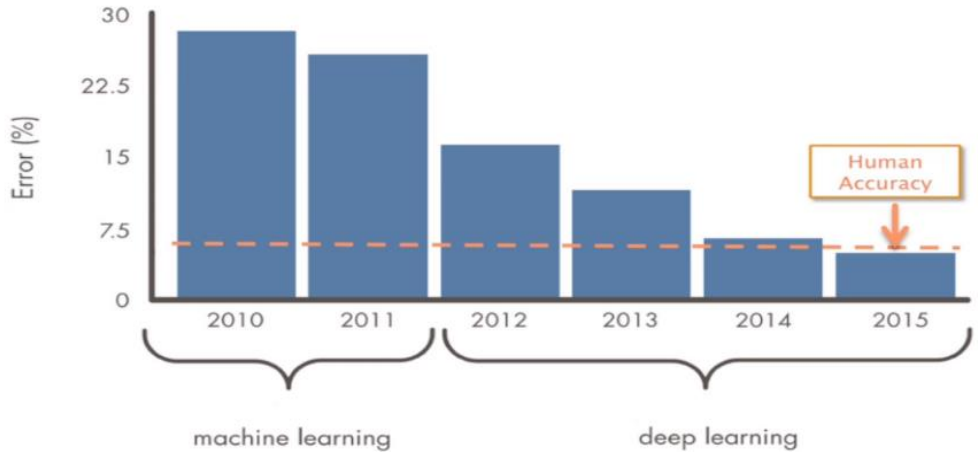



**Figura 21 Evaluación de rendimiento del error (RMS) Vs Precisión de base de datos en imágenes. Respuesta de ML vs DL en el tiempo. Fuente: Mathworks**

El aprendizaje profundo [73], y el desarrollo de las GPU (graphics processing unit) [16] ha permitido que las redes neuronales convolucionales tengan una precisión mayor a las técnicas clásicas de ML siendo su arquitectura, optima en la extracción de caracteres y filtros automáticamente al aumentar la base de datos de entrenamiento [74]. Las convnet Alex Net y Resnet50, fueron entrenadas con millones de imágenes, y diseñadas con decenas de capas para clasificar 1000 categorías de imágenes las cuales hoy, con la **transferencia de conocimiento,** son de código abierto al público para descargarse y modificarse. Estas CNN, cuentas con varios filtros de sombras y bordes al tener actualizados los pesos, facilitando al diseñador, utilizar nuevas categorías de imágenes que nunca ha visto la red y modificarla con varias centenas de imágenes para entrenamiento y no millones como los primeros creadores [54]. De esta forma se ha demostrado el DL como una opción que demuestra buenos resultados en el aprendizaje supervisado para diversas áreas del procesamiento de imágenes [75], reconocimiento de voz [76], análisis de sentimiento [77], juegos [78] y navegación de robot [79], campos de la salud [80], identificación de rostros [81], análisis de imágenes médicas [82] entre otros. Para este caso concreto, se busca lograr clasificar con precisión aceptable los obstáculos tales como otros vehículos, peatones y señales de tránsito. Las nuevas técnicas de DL demuestran precisión del 96% y un error medio del 1,6% con error máximo del 4% y error mínimo del 0,001% usando cámaras de profundidad [36]. Considerando como necesidad el utilizar sensores de bajo costo como las cámaras RGBD en vez de otras tecnologías [11], cobra sentido implementar Kinect para la utilización en carreteras como asistente de conducción.

Imitando el procesamiento de la corteza visual de los mamíferos [83], [71], usando herramientas adquiridas por transfer Learning como la CNN Alex Net (ver **Figura 22**) [12] y GoogleNet en 2014 [52] mostrando un rendimiento muy cercano al nivel humano, al obtener una tasa de error de 6,67% [57], han convertido a las CNN, en una alternativa excepcional gracias a su desempeño en reconocimiento de imágenes.

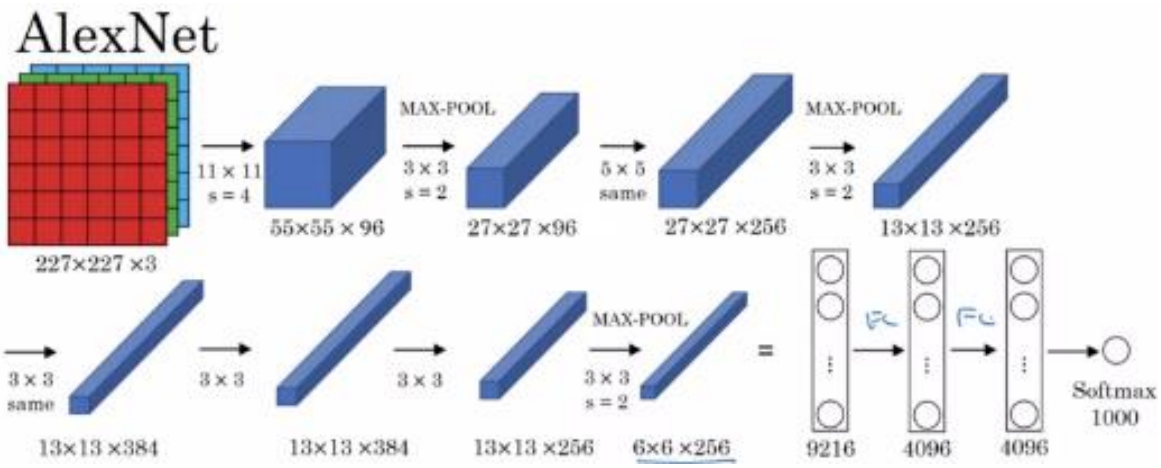



**Figura 22 CNN Alex Net [72]**

## ▪ 2.3.2 Redes neuronales convolucionales por región de interés(detección).

Los avances en DL y CNN permiten el entrenamiento de miles de imágenes completamente segmentadas [84], y de alta resolución [37].

La detección de objetos ayuda en la estimación de posturas, detección de vehículos, vigilancia, etc. La diferencia entre los algoritmos de detección de objetos y los algoritmos de clasificación es que, en los algoritmos de detección, no solo lo clasifica sino intenta dibujar un cuadro delimitador alrededor del objeto de interés para ubicarlo dentro de la imagen.

Existen diferentes análisis de peatones ya que no son objetos rígidos y el reconocerlos varía según la postura, distancia y apariencia, sumado a ello el background o fondo es variable, lo que lo hace más complejo.

En los antecedentes, la extracción de caracteres se realizaba a través del procesamiento de imágenes como el Viola-Jones (VJ) [90] e Histograma de gradiente orientado (HOG) [88] similares a Haar, pero se vieron fácilmente afectadas por la luz a lo que el modelo basado en piezas deformables (DPM) [91] mejora en calidez y puede resolver eficazmente el problema de la oclusión, pero por su complejidad lo hace lento.

En el año 2014, una CNN combinada con algoritmos de identificación de regiones [85], la cual se le llamó R-CNN, era capaz de evaluar una gran cantidad de regiones y localizar varios OI. Ross Girshick propuso un método en el que se utiliza la búsqueda selectiva para extraer 2000 regiones de la imagen produciendo un vector de características de 4096 dimensiones como salida y las llamó propuestas de región conocido como las R-CNN [85], en las que como se evidencia en la **Figura 23**, bordea con un cuadrado el ROI. La RCNN no resultaba practica ya que era lenta por la variedad de operaciones utilizando la máquina de vector soporte **SVM** (Support Vector Machine) para clasificar la región y 2000 CNN en las 2000 regiones sobrepuestas.

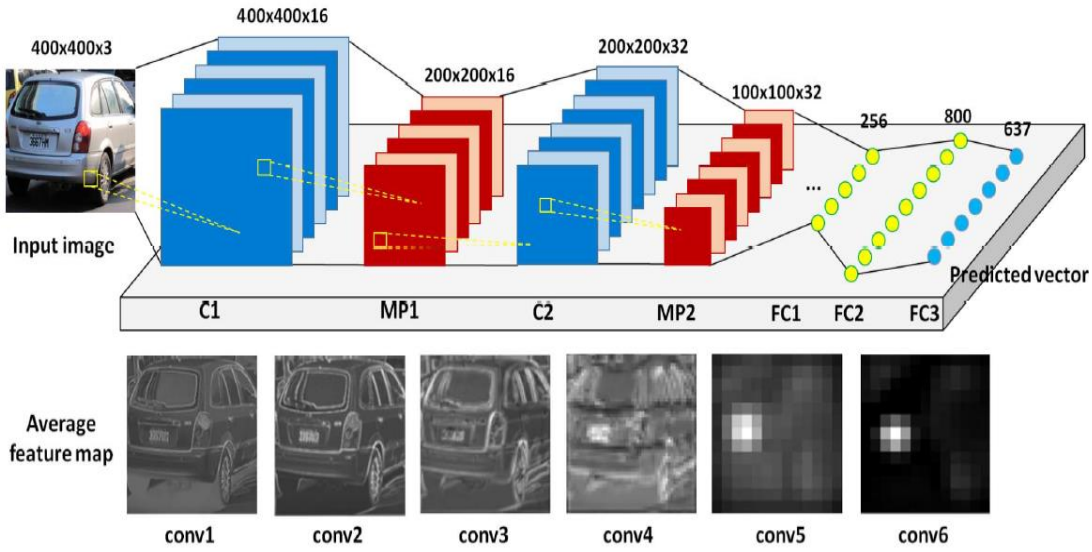



**Figura 23 Arquitectura de detección por regiones [85]**

La R-CNN solía ser muy lenta en el procesamiento de su imagen a tiempo real [86], demorando de 2 a 40 segundos [87], dando lugar a arquitecturas más robustas y rápidas como la arquitectura Faster R-CNN [89] obteniendo mejores tiempos de procesamiento (250 veces más rápida que la primera RCNN), al ser una suma de la antigua Fast RCNN y

una RPN (region proposal network). Se crea un algoritmo de detección de peatones y detección de vehículos basado el algoritmo You Only Look Once (YOLO) [61], con extracción de características optimizada. Posteriormente, se estudia para la detección de peatones, donde decidieron hallar las dimensiones de los anchors, dotando la experiencia a priori en vez del algoritmo de agrupamiento K-mean para el algoritmo original YOLOv2, junto a un análisis estadístico usando el valor inicial del cuadro de preselección usando minería dura negativa (hard negative mining).

Entre los algoritmo de detección se encuentran redes neuronales convolucionales basadas en regiones (R-CNN) [85], SPP-net [92], Fast R-CNN [93], Faster R-CNN [94], R-FCN (redes completamente convolucionales basadas en la región) [95]; generalmente obtienen una Región de interés (ROI) mediante la Búsqueda selectiva (SS) o la Red de propuesta de región (RPN) y luego clasifican cada región utilizando el modelo CNN para obtener la categoría y el nivel de confianza. En contra parte, se crea la arquitectura “de extremo a extremo” (end to end), entre ellas la YOLO y SSD (Detector de cuadro múltiple de disparo único) [96]. Estas, son arquitecturas de un solo canal para combinar la caja de objetos, mejorando la eficiencia de ejecución de la identificación, sin embargo, se debe mejorar la precisión del reconocimiento, especialmente para la identificación de objetivos pequeños.

En la actualidad, los métodos basados en la Propuesta de Región aún prevalecen, pero las ventajas en la velocidad de detección del método de extremo a extremo han presentado mejores resultados como se evidencia con la YOLO frente a las RCNN en la **Figura 24**.

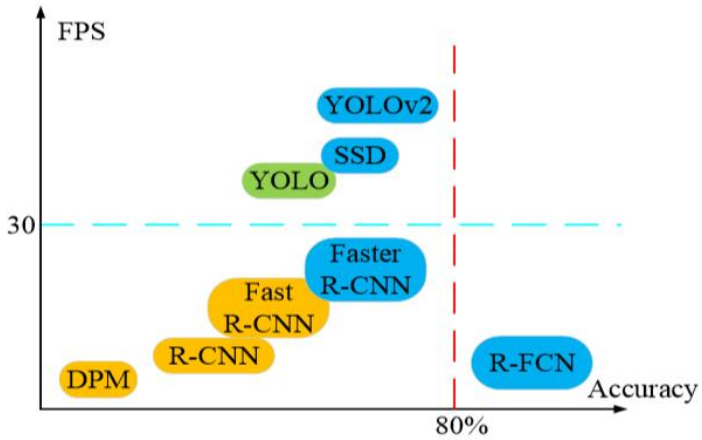



**Figura 24 Arquitectura de detección por regiones [97]**

En el Picasso data set, se realizó un comparativo entre diferentes arquitecturas (ver **Figura 25**), se presenta la arquitectura que compara la precisión vs sensibilidad (recall), mostrando mejores rendimientos evaluando puntuación o Score por parte de la YOLO, al detectar verdaderos positivos

En el gráfico de la curva de Precisión – Recall se evidencia una mayor precisión frente a la arquitectura RCNN, DPM, y YOLO, siendo esta última, seleccionada para el entrenamiento del trabajo.

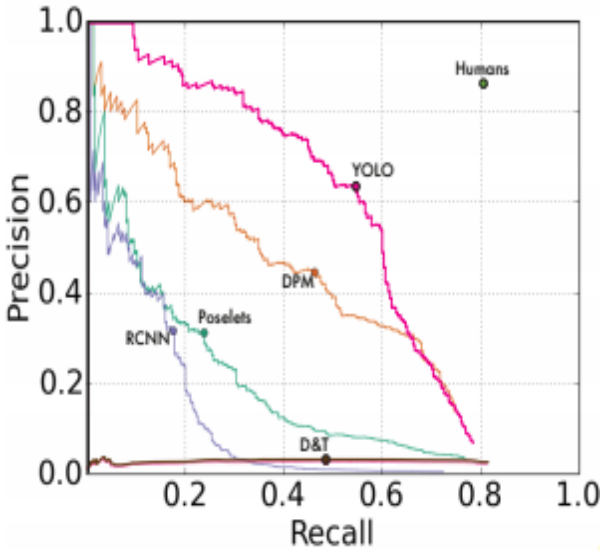



**Figura 25 Picasso data set precisión-recall [98]**

Estas han sido utilizadas en diferentes sistemas de robot móviles [99], con tarjetas embebidas para detectar personas entre otros objetos [61].

La arquitectura YOLO, se escoge por demostrar un buen rendimiento en velocidad de procesamiento en FPS, precisión y sensibilidad frente a arquitecturas como la Faster RCNN. Las CNN como algoritmos de DL, demostraron poder obtener los filtros a través de técnicas de convoluciones, permitiendo actualizar los pesos automáticamente como lo haría una red neuronal multicapa, además de presentar un mejor rendimiento que las técnicas clásicas de ML, gracias a los avances demostrados en los recientes concursos de imageNet.

## ▪ 2.3.2 Antecedentes en carros Autónomos

Los carros autónomos asemejan el comportamiento de un auto tripulado, siendo capaces de tomar decisiones, al percibir una aproximación del panorama por “visión artificial”, y señales tomadas a través de sensores láser Lidar, ultrasonido, cámaras, radares, GPS, los cuales son calculados computacionalmente, permitiendo la toma de decisiones según los OI, de acuerdo al mapeo existente de las carreteras en las que transita [100]. Empresas como BMW, Tesla, Renault, Mercedes Benz, Google, Daimler AG, Volvo, Nissan, Ford están empeñadas en alcanzar la autonomía total de los automóviles [101]. El primer prototipo en carro autónomo fue expuesto por la General Motors en 1939 direccionado a través del control de una tarjeta embebida, y expuesto en una feria de tecnología, aunque las mejoras son notables a partir de los años 80s [18]. Empresas como Mercedes-Benz, la Agencia de Proyectos de Investigación avanzada para la defensa (DARPA) y los laboratorios HRL, presentan prototipos funcionales para zonas agrestes y sin tráfico. Audi en 2014 crea un prototipo de coche que compitió en velocidad contra uno tripulado en zonas controladas [102]. En 2015 el Audi SQ5 con apoyo de Delphi tienen un auto que recorre 5400 km durante 9 días con pequeñas intervenciones humanas [103]. Este mismo año Google [18] pone a prueba 25 de sus diseños de coches a transitar autónomamente a una velocidad promedio de 40 km/h en California (ver **Figura 26**), utilizando el software de sus autos tripulados Lexus los cuales tienen varios millares de kilómetros recorridos.

**Figura 26 Auto de Google 2014 [18]**

Desde finales del año 2016, la empresa Nutonomydel, lanza el primer taxi autónomo del mundo en Singapur, Uber en Pittsburgh y San Francisco [104], sin embargo, en 2018 detiene la operación por un accidente donde una mujer es atropellada por un vehículo no tripulado [23]. Alguna de las ventajas en los coches autónomos está en la seguridad vial [105], el acceso para individuos en situación de discapacidad [106], eficiencia energética,

reducción de gases invernadero, tráfico con menor densidad de autos [107], optimización al estacionarse. El software utilizado requiere trabajo a tiempo real y el algoritmo debe ser de rápido procesamiento para responder a situaciones dinámicas. El machine learning, la probabilidad y el Deep learning ha permitido mejoras en “predecir y reconocer” comportamientos en las calles, para una óptima toma de decisiones para los coches autónomos, como lo viene realizando Nissan y Tesla (ver **Figura 27**). Uno de los obstáculos en promover los coches autónomos es la reglamentación por factores políticos, jurídicos, de infraestructura entre otros [108]. El Parlamento Europeo el 7 de julio de 2010 establece el marco para la implantación de los sistemas de transporte inteligentes, para un marco común en territorio europeo [109] mientras que el departamento de Vehículos a Motor de Nevada otorga la primera licencia para un coche no tripulado marca Toyota Prius con apoyo de Google [110] en estados unidos desde el 2012 extendiéndose hoy a california y florida [111].

**Figura 27 El sistema "Autopilot" del coche eléctrico Tesla Model S funciona con nivel de autonomía de nivel 3 a 4 solamente en autopistas y no en vías urbanas Tesla S [68]**

### Renault Symbioz conducción autónoma nivel 4 [112]

El nivel 4, libera al conductor de todas las tareas de conducción dinámica cuando se enciende el sistema de conducción automática del automóvil. Ya no es necesario que el conductor se concentre en el camino por delante. El automóvil puede moverse a una posición segura si no puede lidiar con un incidente inesperado por delante. En autopistas autorizadas o carreteras duales con una franja mediana, un automóvil autónomo de Nivel 4 como el automóvil de demostración SYMBIOZ puede ajustar la velocidad del vehículo de acuerdo con el automóvil que se encuentra al frente, permanecer en su carril incluso en las curvas, cambiar de carril (es decir, adelantar a otro automóvil o salir de la autopista) y funcionar solo en atascos. La automatización de conducción de nivel 4 actualmente no está

permitida por las normas de circulación francesas.

Los seis socios principales involucrados junto con los equipos de Renault en el desarrollo del auto de demostración SYMBIOZ proporcionaron información en sus áreas específicas de experiencia:

- LG ha estado involucrado en el desarrollo de las interfaces hombre-máquina.
- Ubisoft ha proporcionado experiencia de realidad virtual a bordo para el modo de conducción autónomo.
- Devialet ha desarrollado una nueva experiencia de usuario a través de un sistema de sonido avanzado.
- Sanef ha trabajado en la forma en que el automóvil se comunica con la infraestructura de la red vial.
- TomTom ha contribuido con su experiencia en geoposicionamiento.
- IAV ha proporcionado experiencia en ingeniería de conducción autónoma.

En conclusión, varios avances en CA han permitido la reducción de siniestros viales, mayor comodidad, seguridad y descongestión en carreteras, sin embargo, los costos asociados a estas tecnologías suelen ser altos, a lo que el crear un prototipo para cualquier carro en países con un nivel de poder adquisitivo menor como Colombia, podría aportar a mejorar la movilidad y reducir los accidentes viales.

### ▪ 2.3.3 Algoritmos para la conducción autónoma

En cuanto a investigación de procesamiento de imágenes para el movimiento autónomo en vehículos, se encuentra en la literatura carros autónomos utilizando cámaras capaces de reconocer personas, utilizando sensores de profundidad con el uso de la R CNN [113], buscando reducir siniestros viales. Se desarrollan algoritmos de detección de colisión con modelos geométricos [114], algunos especializados en evitar obstáculos [115], estimar la ubicación de agentes [36], y detectar la pista del carril [116].
El procesamiento de imágenes utilizando visión de máquina se ha estudiado para la limpieza autónoma en hogares [128], para exteriores [118].

**Segmentación semántica**

Arquitecturas complejas como la SegNet, usando segmentación semántica para la detección de peatones [117] [131], con una fiabilidad cercana al 95% usando cámaras RGBD. En [85] se evidencian investigaciones en coches autónomos y en [16] utilizando segmentación semántica, la cual clasifica y detecta objetos de interés evaluando pixeles de personas y señales de tránsito ver **Figura 28**, sin embargo, etiquetar las imágenes requiere un trabajo extenuante a diferencia de otras arquitecturas que encierran en un cuadro la región de interés como la YOLOv2 y la Faster RCNN.

**Figura 28 Seg-Net Escena de una carretera, imagen en color (izquierda) y los correspondientes píxeles etiquetados (derecha) [117]**

En cuanto a la simulación de vehículos autónomos se han desarrollado software especializados en el análisis de giros para la conducción autónoma centrándose en el modelado realista de la dinámica física del coche e inteligencia artificial especial mente difusa [113] [119]. El software TORCS (The Open Racing Car Simulator) es una herramienta de código abierto elegido para proporcionar el entorno virtual [120]. Otros simuladores especializados en robótica y de código abierto se encuentra V-Rep y Gazibo de fácil acople a Ross [121], y Matlab [142]. A continuación, se presentan diferentes estudios realizados en simuladores para la conducción autónoma, entre ellos el uso de redes neuronales, redes convolucionales, redes por reforzamiento, control neuro difuso, controladores PID, cinemática y ética en la conducción.

**Simulación de vehículo autónomo usando v-Rep bajo Ros [125]**

Para simular el sistema de dirección del vehículo real con aceleraciones finitas, éste se ha simulado en VREP como un sistema dinámico (con inercia y fricción) actuado por un motor servo-controlado en posición. Esto quiere decir que la posición del volante es controlada internamente por V-REP mediante un PID. El simulador recibe consignas de posición del volante de la misma manera que el vehículo real. La posición actual del volante determina el ángulo de dirección de una rueda directriz virtual central según el modelo bicicleta. Mediante una serie de relaciones matemáticas, se establece la orientación de las ruedas delanteras de tal manera que se cumpla la disposición Ackermann de las mismas para un centro instantáneo de rotación determinado. Los sensores utilizados en el vehículo son: un láser LIDAR 3D, una cámara estéreo y un GPS. Se ha creado un modelo de cada uno de ellos que simula su comportamiento [125]. La trayectoria se obtiene con una lista de puntos de paso objetivo, usando el algoritmo pure pursuit [126], para seguir la trayectoria discretizada. Este algoritmo publica en cada ciclo de control un comando de velocidad (tangencial y angular) que describe un radio constante en cada instante infinitesimal. Este radio hace que el robot se dirija hacia el siguiente punto objetivo. El look ahead (distancia que establece el punto que “persigue” el vehículo) se ajusta de tal manera que disminuye la velocidad en grandes curvaturas del carril al igual que haría un conductor. Esta estrategia permite que el algoritmo adapte la distancia objetivo para seguir correctamente la trayectoria [127]. Las tareas se definen utilizando un editor de redes de Petri, interpretadas y guardadas en un fichero XML, controlado mediante nodos desarrollados bajo el sistema ROS. Se ha mapeado el campus externo y ha logrado conducir por sus carriles siguiendo la línea central mediante el algoritmo de seguimiento de trayectoria [125]. La **Figura 29**, evidencia la estructura del software.

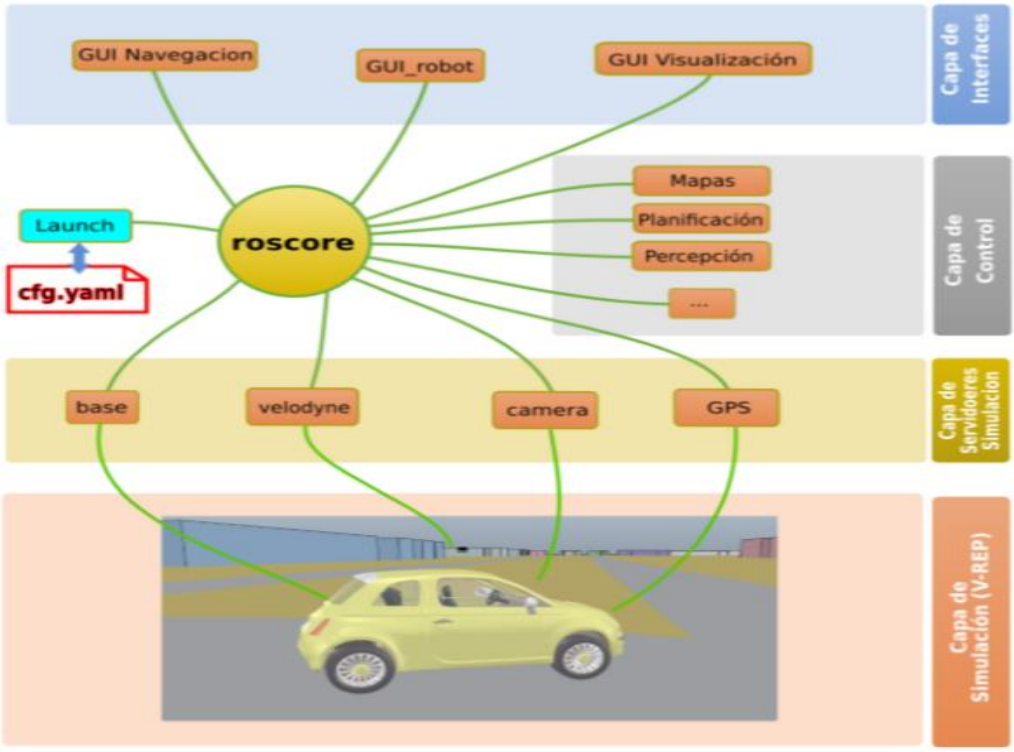



**Figura 29 Estructura de sistema Smart Car para simulación V-Rep bajo Ros [125]**

### Reconocimiento de carreteras utilizando Maquina de Vector Soporte (SVM) [8]

Este documento presenta el rendimiento de Support Vector Machines (SVM) para reconocer imágenes de la carretera. Todas las imágenes son recopiladas por un automóvil. Se dividen en 4 clases: gire a la izquierda, gire a la derecha, avance y pare. Usaron el 90% de ellos para entrenamiento y el 10% para probar el modelo. La precisión del modelo es de aproximadamente 70.77%. Las imágenes de unión están etiquetadas para girar a la izquierda y girar a la derecha y otras imágenes están etiquetadas para avanzar y detenerse. [122] en la **Figura 30** se aprecia el flujograma de procesamiento y toma de decisiones con el SVM.

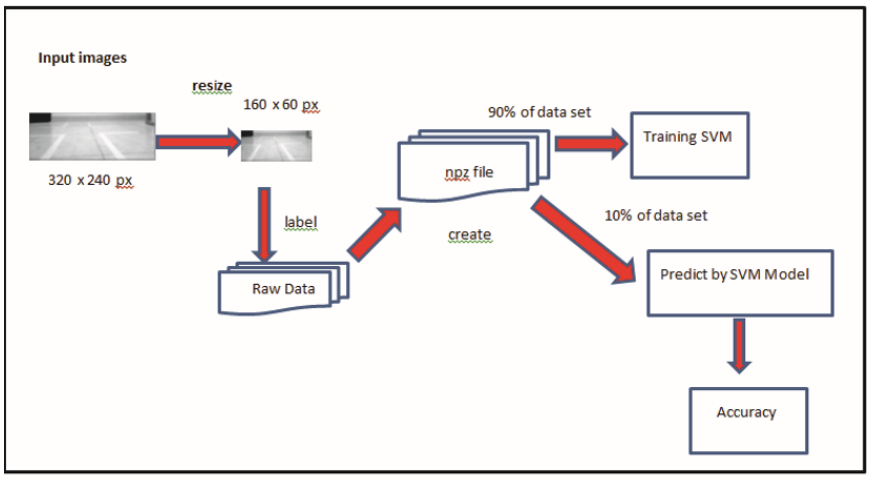



**Figura 30 Reconocimiento de carreteras con SVM [8]**

**Un sistema eficiente y confiable de asesoramiento de velocidad óptima en semáforos con luz verde, para automóviles autónomos [122]**

Un automóvil autónomo puede ser más conveniente, más seguro y menos intensivo en energía. Además, los sistemas Green Light Optimal Speed Advisory (GLOSA) reducen el tiempo de viaje y las emisiones de CO2. En este artículo, el autor propone un GLOSA novedoso, llamado sistema R-GLOSA para soportar autos autónomos. Suponen que un automóvil autónomo puede acceder a todos los horarios de tránsito ligero que encontrará en su ruta. La ruta se divide en cada segmento de acuerdo con las luces de tráfico. En cada segmento, un automóvil autónomo puede comunicarse con las unidades de carretera (RSU) distribuidas en las calles y luego optimiza la velocidad para llegar a la intersección cuando la luz es verde (ver **Figura 31**). La velocidad para la cobertura de cada RSU parte de la cinemática de física clásica conociendo las variables distancia, fracción de rozamiento y velocidad final e inicial para optimizar la velocidad y reducir os atascos. Dos enfoques superan a GLOSA en términos de velocidad óptima y tiempo de viaje. Según la densidad del vehículo, un automóvil autónomo calcula la velocidad óptima para llegar a la intersección cuando la luz es verde.

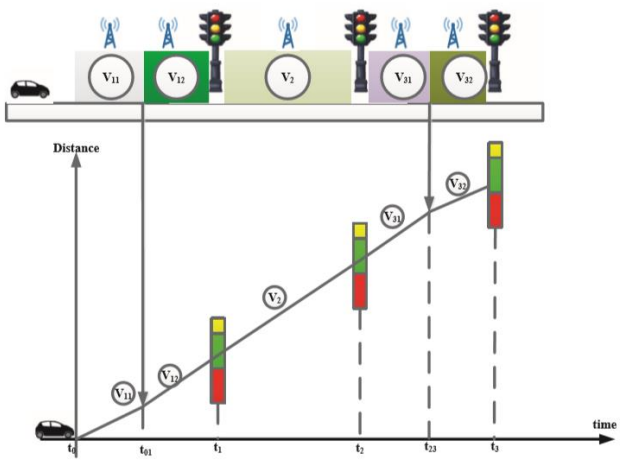



**Figura 31 Sistema multi segmentación GLOSA para semaforización verde [122]**

### Piloto autónomo utilizando lógica difusa y control PID [6]

Para este proyecto, el autor utiliza controladores basados en su mayoría, lógica borrosa, una técnica intuitiva que ha demostrado ser eficiente a la hora de extraer el conocimiento humano en procesos complejos, como la conducción de vehículos en entornos reales. El modularidad de la arquitectura presentada permite incluir nuevos dispositivos sensoriales y de comunicación, como los sensores RFID (radio frecuencia) para comunicarse entre otros coches. Además, se pueden ajustar tanto las reglas como las funciones de pertenencia de las variables lingüísticas presentadas para entornos urbanos y autopistas.

El controlador borroso presentó un buen funcionamiento, y el tiempo necesario para su ajuste es considerablemente inferior al resto. El controlador i-PI presenta un mejor resultado en estado estacionario, sin embargo, presenta muchas oscilaciones en el transitorio. Además, tiene problemas cuando el cambio en la referencia es negativo.

### Conducción autónoma basada en Q Learning simulado en Unity [123]

El aprendizaje reforzado profundo (DRL) combina el aprendizaje reforzado con el aprendizaje profundo [124]. En la práctica, el aprendizaje automático con imágenes como entradas se logra mediante redes neuronales convolucionales (CNN). Deep Q Network

(DQN) es una forma de aprendizaje reforzado en el que el resultado de un CNN no es clasificación, sino valores Q (recompensas) dadas, a las acciones resultantes de los estados de entrada, el cual fue un éxito en el dominio desafiante de los juegos clásicos de Atari 2600 [129]. La tasa de aprendizaje y el factor de descuento son los dos parámetros más importantes. El entorno de simulación se crea utilizando Unity (https://unity3d.com/). La interacción entre el programa Unity y Python es proporcionada por los módulos de conversión Unity-Python. El coche a partir de lo que reconoce decide si ir recto, o girar como se evidencia en la **Figura 32**.


**Figura 32 Conducción autónoma para Deep Q Learning en Unity [123]**

### Algoritmos de ética para coches autónomos [22]

En tiempos de accidentes inevitables, los vehículos autónomos enfrentan situaciones críticas de toma de decisiones en las que están en juego dos o más vidas. En tiempos como estos, los vehículos se enfrentan a dilemas éticos, es decir, minimizar el número de muertes. El autor utiliza un algoritmo de ordenamiento de montón mínimo basado en cola prioritario (priority queue based min-heap sorting algorithm), para clasificar las vidas de las personas de acuerdo con algunos parámetros dados. En este algoritmo, las operaciones de trabajan por nodos y valores semejantes a arboles de decisión, usa el término probabilidad de supervivencia, que indica la probabilidad de que una persona sobreviva [22], según variables como la edad de esas personas y el género. Los autores dejan a criterio del dueño del auto, poner valores de su elección a estas variables e insertar valores diferentes la razón para hacerlo, es que la opinión sobre este asunto puede variar de persona a persona. A medida que los algoritmos de optimización de fallas afectan algunas

de nuestras intuiciones éticas más profundas, diferentes personas tendrán diferentes juicios sobre cuál debería ser el curso correcto de las acciones. Las personas pueden estar en total desacuerdo con las respuestas adecuadas a dilemas éticos como los propuestos. Pseudocódigo del algoritmo [22]

1. Calcule la probabilidad de supervivencia para cada incidente.
2. Evaluar y generar una puntuación para cada incidente.
3. Cree un montón sin clasificar.
4. Crear un montón mínimo.
5. Elimine o inserte nuevos nodos si se producen nuevos incidentes.
6. Elija la raíz si el límite de tiempo excede.

**Conducción autónoma inteligente en un ambiente virtual con redes neuronales para movimiento transversal y fuzzy para movimiento longitudinal [7]**

Para este trabajo de simulación, el autor se basa en técnicas de inteligencia computacional y crea una red neuronal para determinar la dirección y un sistema de lógica difusa que controlar el freno y el acelerador. El sistema se evalúa en un entorno virtual simulado, pero está diseñado para funcionar e integrarse en un vehículo físico. [7]

El software TORCS (The Open Racing Car Simulator) fue elegido para proporcionar el entorno virtual [137]. Este software de simulación es una herramienta de código abierto ampliamente utilizada en la comunidad científica para proyectos de investigación relacionados con la inteligencia artificial [96], [139]. El modelado realista de la dinámica física del automóvil, la posibilidad de acceder al código fuente del simulador y la facilidad para establecer un protocolo de comunicación entre TORCS y un algoritmo inteligente desarrollado en lenguajes de programación como Python, son las razones de elección, el cual es conectado con un programa de conductor externo mediante el uso del software de competición Simulated Car Racing Championship, desarrollado por el Politécnico di Milano [140]. En la **Figura 33** se evidencia en paralelo el uso de la red neuronal y fuzzy por parte del autor.

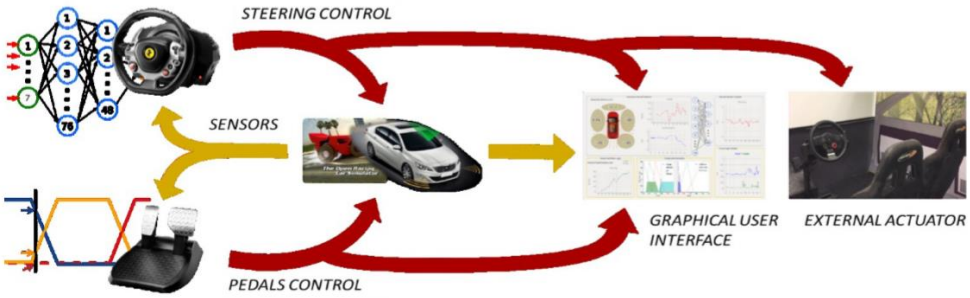



**Figura 33 Red neuronal movimiento transversal, fuzzy movimiento longitudinal [7]**

La arquitectura de la red neuronal utilizada se asemeja a la de un Perceptrón Multicapa (MLP) la cual se compone de dos capas ocultas de 76 neuronas la primera y 48 neuronas la segunda. La capa de entrada consta de siete neuronas, una para cada entrada (cinco mediciones de los grupos de sensores de obstáculos, el ángulo del eje de la pista y la distancia al centro de la pista) y la capa de salida consta de una sola neurona que representa la salida de La red neuronal (la dirección del automóvil).

Para el diseño del sistema de lógica difusa para movimiento longitudinal, el autor define los conjuntos difusos y sus funciones de membresía. Debido a que este componente tiene dos variables de entrada (distancia al borde de la pista y la velocidad del automóvil) y dos variables de salida (acelerador y freno), se construyeron un total de cuatro conjuntos difusos [7]. En **Figura 34** se evidencia la interacción de variables y en la **Tabla 6,** las reglas para la toma de decisiones en el control de aceleración vs freno según la distancia

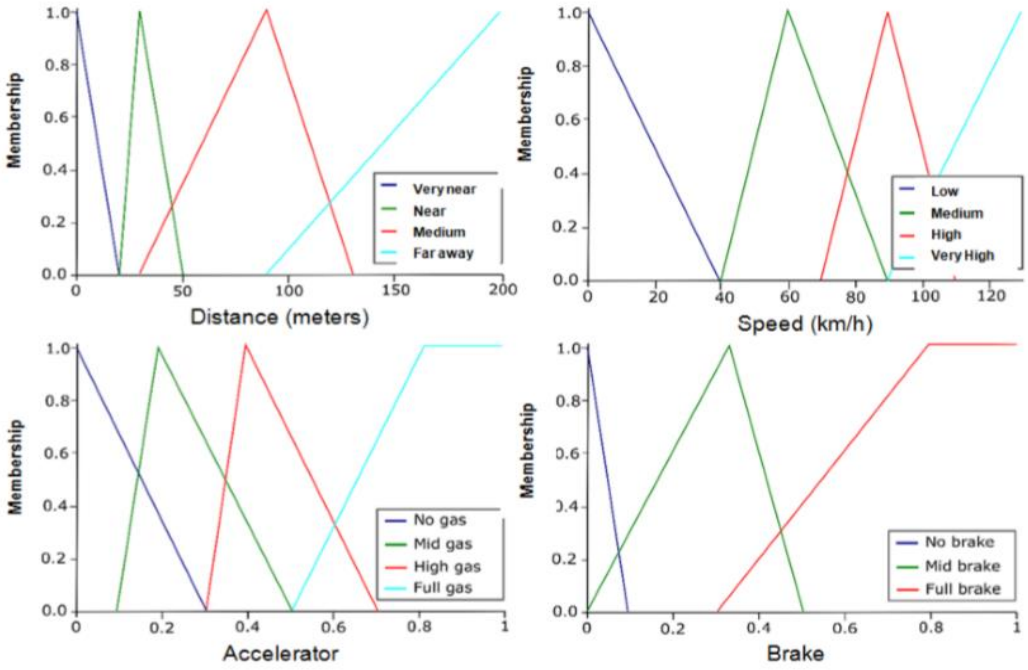



**Figura 34 Fuzzy de Entradas y salidas para control de aceleración y freno [7]**

**Tabla 6 Entradas y salidas Fuzzy para control de aceleración y freno [7].**

| Distance | Speed | | | |
|---|---|---|---|---|
| | *Low* | *Medium* | *High* | *Very High* |
| *Far away* | Full gas | Full gas | Mid gas | Mid gas |
| *Medium* | Full gas | Full gas | Mid gas | High gas |
| *Near* | High gas | High gas | No gas | No gas |
| *Very near* | High gas | High gas | No gas | No gas |

| Distance | Speed | | | |
|---|---|---|---|---|
| | *Low* | *Medium* | *High* | *Very High* |
| *Far away* | No brake | No brake | No brake | No brake |
| *Medium* | No brake | No brake | No brake | Mid brake |
| *Near* | No brake | No brake | Mid brake | Full brake |
| *Very near* | No brake | Mid brake | Full brake | Full brake |

## Reconocimiento de coches utilizando la Red YOLOv2 en Opencv [38]

El trabajo del autor tuvo como meta el lograr la clasificación y detección de objetos con el diseño de un sistema de reconocimiento automático de vehículos mediante el uso del aprendizaje profundo para facilitar el desarrollo de una aplicación de seguridad vehicular. El sistema se basó en un detector de objetos desarrollado con técnicas de aprendizaje supervisado y planteado con redes neuronales artificiales convolucionales, haciendo uso de la plataforma Caffe y sus librerías, así como también de librerías en Opencv y scripts desarrollados en Python [38]. En la **Tabla 7,** se evidencia la respuesta de imágenes de entrada vs las de salida.

| Situación | Imagen de entrada | # de vehículos | Imagen de salida | # de vehículos detectados |
|---|---|---|---|---|
| Vehículos cercanos |  | 1 |   | 1 |
| Vehículos lejanos |  | 1 |   | 1 |

**Tabla 7 Reconocimiento de autos con arquitectura YOLO [38].**

### Reconocimiento de peatones y automóviles utilizando la Red YOLOv2 [97]

En este trabajo se propuso un sistema de detección basado en el YOLOv2 [61], para reconocer peatones y automóviles utilizando estrategias importantes como la minería dura y negativa, con la que mejoran la precisión de detección. Luego, utilizan la base de datos KITTI [132] para entrenar la red, utilizando la Darknet-19 como un extractor de características la cual tiene 19 capas convolucionales y 5 capas de agrupación máxima siendo semejante a la red GoogleNet y a la Resnet [133] que combina características de alta resolución con características de baja resolución, para agregar características más finas. Para hallar los Anchors de los recuadros no utilizaron la agrupación K-means para seleccionar automáticamente los mejores cuadros iniciales, sino información a priori como se ve en la **Figura 35**. Los Anchors para peatones, fueron con altura: ancho 3: 1, y la radio del borde del vehículo, altura: ancho 2: 3. Las proporciones de píxeles de la altura y el ancho del vehículo están entre [0.36, 0.5] y [0.72, 0.75], debido a diferentes observaciones visuales desde el costado de los vehículos y detrás o adelante de los vehículos.

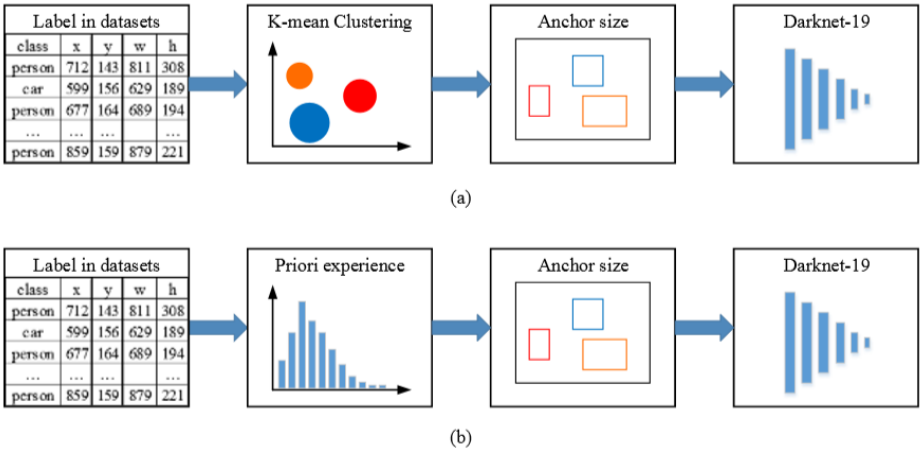



**Figura 35 Anchors por K-means vs experiencia previa en YOLOv2.**

Para las etiquetas de peatones y vehículos usando el conjunto de datos KITTI y el conjunto de datos de peatones Caltech de acuerdo con el conocimiento a priori, rrealizando una comparación entre las arquitecturas Faster RCNN, YOLOv2 y una mejora a la YOLO como se mencionó anteriormente, en las que evidenció mejores resultados reconociendo peatones y automóviles en precisión y velocidad frente a la YOLOv2 (ver **Tabla 8**). La Faster RCNN cuenta con mejores resultados en precisión, pero al ser lenta no la hace tan precisa como el uso de la YOLOv2 para la conducción autónoma.

| Item | Time cost* | FPS | Accuracy | |
|---|---|---|---|---|
| | | | Pedestrian | Vehicle |
| Faster RCNN | 0s | 0.5 | 65% | 75 % |
| YOLO V2 | 128s | 20 | 43.33% | 59.57% |
| PROPOSED | 0s | 22 | 45% | 61.34% |

***The time of obtaining the anchor sizes**

**Tabla 8 Comparativo en reconocimiento de peatones y automóviles entre tres arquitecturas con KITTI [97]**

Otro más cercano, evalúa 5 objetos de interés, con más de 30,000 imágenes del data set COCO, evidenciando buenos resultados y modificando el LR como el número de épocas evalúa la precisión vs recall , pero sin identificar la luz del semáforo [144].

Finalmente, unos más arriesgados, hacen la estimación de profundidad, utilizando una mona cámara y el algoritmo de la YOLOv2, donde relaciona posición frente a la geometría de los bounding boxes a través de una regresión obteniendo un 80.4% de [145].

En el estado del arte, se presentan investigaciones que aportan a la conducción autónoma, entre ellas la segmentación semántica con SegNet aunque el etiquetado toma más tiempo. Otras simulaciones bajo Ros y V-Rep utilizando CNN que clasifica imágenes virtuales pero que se alejan a la aplicación real. Otros avances utilizando máquina de vector soporte para reconocer las líneas de las carreteras con cámaras, asemejándose al modo crucero, pero incapaz de reconocer OI. Propuestas con aprendizaje utilizando redes neuronales multicapa y control de lógica difusa para el frenado y acelerado del automóvil, presentó buenos resultados, a lo que en este proyecto solo se centró en la utilización de la cinemática convencional con fines de dar realismo en la simulación. Simulaciones basadas en Q learning y desarrolladas en Unity, la cual al igual que los juegos de video con aprendizaje reforzado a partir de recompensas, utiliza una CNN, sin embargo, esta solo considera la clasificación de agentes con una CNN, mas no la detección como lo haría la arquitectura YOLO o Faster RCNN siendo más realista la propuesta que plantea este trabajo. Finalmente desarrollos con la arquitectura YOLOv2 y OpenCV para detección de automóviles etiquetando sus propias imágenes junto a otros desarrollos para la detección de automóviles y peatones utilizando la base de datos KITTI, la cual se compara con la faster RCNN, evidenciado por parte de la YOLO, mejores resultados en la velocidad de procesamiento de FPS (casi 40 veces mas rápida) y una precisión semejante, a lo que a diferencia de esta investigación (dos OI), en este trabajo se consideran 6 objetos de interés, y se desarrolla una CNN extra para diferenciar el color del semáforo, dándole una mayor aplicabilidad al algoritmo. La propuesta para este trabajo se realiza con herramientas como Matlab y el uso de un simulador de fácil acople como VREP, permitiendo la utilización de imágenes más realistas, la utilización de una base de datos creada para el contexto latinoamericano (a diferencia de KITII que es contexto europeo), y una mayor cantidad de OI, aportando así al estado del arte.

# CAPITULO 3. ELABORACIÓN DE BASE DE DATOS

El presente capitulo, desarrolla la elaboración de dos bases de datos, una diseñada para comparar la capacidad de clasificación entre dos algoritmos, y una segunda base de datos, para el entrenamiento de la arquitectura que clasifica y detecta los OI.

Para la **primera base de datos**, se toman alrededor de 200 imágenes para 7 clases seleccionadas (carros, señal de pare, bicicletas, motos, personas, semáforo verde y semáforo rojo), almacenando un total aproximado de 1400. Esta base de datos se utiliza para el entrenamiento de la sección 5.2, 5.3 y 5.4 del presente trabajo.

La elaboración de la **segunda base de datos,** utiliza una interfaz gráfica en Matlab con la cual se etiquetan seis (6) categorías divididas en: carros, señal de pare, bicicletas, motos, personas y semáforos. Se realizan y almacenan, varias grabaciones desde la misma cámara para evitar problemas de resolución. Los videos se obtienen en formato mp4, se fragmentan en más 3000 imágenes, convirtiéndolos a frames, y posteriormente etiquetando las 6 categorías mencionadas previamente. Esta segunda base de datos se conformó por 2421 imágenes para entrenamiento, y 605 para validación. El uso de estas bases de datos es utilizado, a partir de la sección 5.5, para el entrenamiento de la arquitectura de aprendizaje profundo, la YOLOv2 basada en regiones, la cual se utiliza como herramienta fundamental en la realización del trabajo, “asistente para la conducción” en la detección y clasificación de los seis agentes.

## ▪ 3.1 Creación de base de datos #1 para evidenciar el algoritmo con mejor comportamiento en reconocimiento.

En la elaboración de la base de datos, fueron seleccionadas 7 clases, con el fin de acotar el trabajo. Se realizan y almacenas varios videos en Bogotá que contienen los OI, se convierte a frames y de cada fotograma se recortan los agentes que interesan. Se almacenan aproximadamente 200 recortes por cada una de las siete clases, para un total de 1400 imágenes, las cuales se categorizan en 7 carpetas con los respectivos títulos: **bicicletas, carros, Motos, pare, personas, Semáforo rojo, Semáforo verde**, como se evidencia en la **Figura 36**. Los objetos de interés seleccionados se toman a partir del criterio de factor de

riesgo en siniestros viales. Se agrupa en el folder "**semáforo rojo"**, imágenes de semáforos en luz roja y amarilla considerando los dos colores como una señal de alerta para detenerse, por el contrario, el grupo de "**semáforo verde"** se considera como una señal de continuar**.** La señal de "**pare**" se consideró como un aviso de precaución que implica detenerse totalmente. No se tuvo en cuenta las señales de velocidad, debido a que el criterio máximo en la simulación será inferior a 30 km/h, apto por la OMS. Se agrupan en la clase **"carros"**, automóviles, buses y camionetas sin distinción, y se considera un factor relevante al haber aproximadamente 1.5M circulando en Bogotá. La clase "**motos", "bicicletas" y "personas"**, se consideró relevante por ser los más afectados en siniestros viales de acuerdo a los registros de la secretaria de movilidad de 2015 [29]. Mas detalles de los criterios de análisis en toma de decisiones para la simulación, se realiza en el capítulo 6.


**Figura 36 Selección y recorte manual de data set para entrenamiento**

Los recortes deben tratar de mostrar la mayor cantidad de features para evitar el sobre entrenamiento (overfiting) es decir, las imágenes recortadas, deben almacenarse desde diferentes ángulos para que el aprendizaje del algoritmo tenga una mayor extracción de filtros y características. En la **Figura 37** se presentan algunas de las imágenes de la base de datos #1, las cuales se redimensionan a 227 x 227 pixeles, ya que serán las que se van a utilizar para los algoritmos el entrenamiento de algoritmos de ML y posteriormente un nuevo entrenamiento de DL con la CNN ALEXNET, la cual requiere de este criterio de entrada para realizar el entrenamiento (ver más en el capítulo 5).

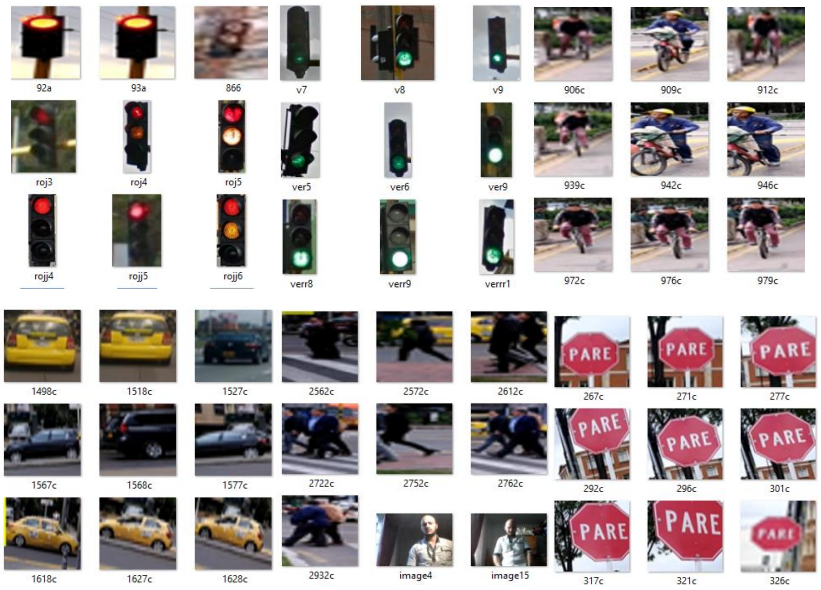



**Figura 37 Imágenes almacenadas para data set**

## ▪ 3.2 Creación de base de datos #2 con etiquetado de objetos de interés para entrenamiento de la arquitectura YOLOv2.

En esta sección se utiliza una interfaz gráfica de Matlab para etiquetar seis (6) categorías, divididas en **carros, señal de pare, bicicletas, motos, personas y semáforos**.

Se realizan grabaciones con la misma cámara para evitar problemas de resolución, las cuales serán utilizadas para extraer imágenes por frames, y posteriormente etiquetar los objetos de interés para generar una base de datos para entrenamiento y para validación, las cuales serán detectadas y clasificadas utilizando una arquitectura basada en regiones. Para este caso concreto se utilizará la arquitectura YOLOv2 la cual será entrenada y evaluada a partir de la sección 5.5, utilizando el grafico Precisión-Recall para localización de objetos de interés.

Cada grabación se dio en formato mp4, tuvo una duración de 2 minutos en promedio, y las imágenes se redimensionan a una resolución de 224 x 224 pixeles, debido a que la red neuronal convolucional RESNET50, que utiliza la arquitectura YOLOv2, requiere este formato en dimensión como entradas, para poder procesar la información. Los frames tomados, presentaban situaciones cotidianas desde un automóvil (ver anexo “herramientas e instrumentos), haciendo diferentes maniobras humanas, entre ellas se consideró importante adquirir información del auto detrás de automóviles y motos a una distancia menor a 10 metros para mantener el reconocimiento de caracteres, y se realizan en calles

congestionadas, y autopistas importantes entre ellas, la avenida 26, la carrera 30 NQS, la calle 100, la carrera 11, procurando poder obtener información de varios agentes en un ambiente real. Los videos obtenidos se realizan en un ambiente controlado, es decir, la mayor cantidad de fotogramas cuenta con buena iluminación, no hay lluvia y se realiza con la misma cámara. También se efectuaron tomas del auto detenido frente a semáforos en luz roja, luz verde y señal de pare donde se evidencia el buen y mal uso de las cebras, y de algunos peatones y automóviles que atraviesan las vías estando la luz de semáforo en verde entre otros fenómenos considerados imprudencias en las vías. Para obtener las imágenes, se adquieren frames cada 700 milisegundos aproximadamente, obteniendo un total de 2421 imágenes para entrenamiento y 605 imágenes para validación. Esta base de datos con frames resultantes, es etiquetada manualmente, como se muestra en la **Figura 38** utilizando un toolbox de Matlab bautizado –ImageLabeller- donde se categorizan los seis (6) ROI Label (region de interés).

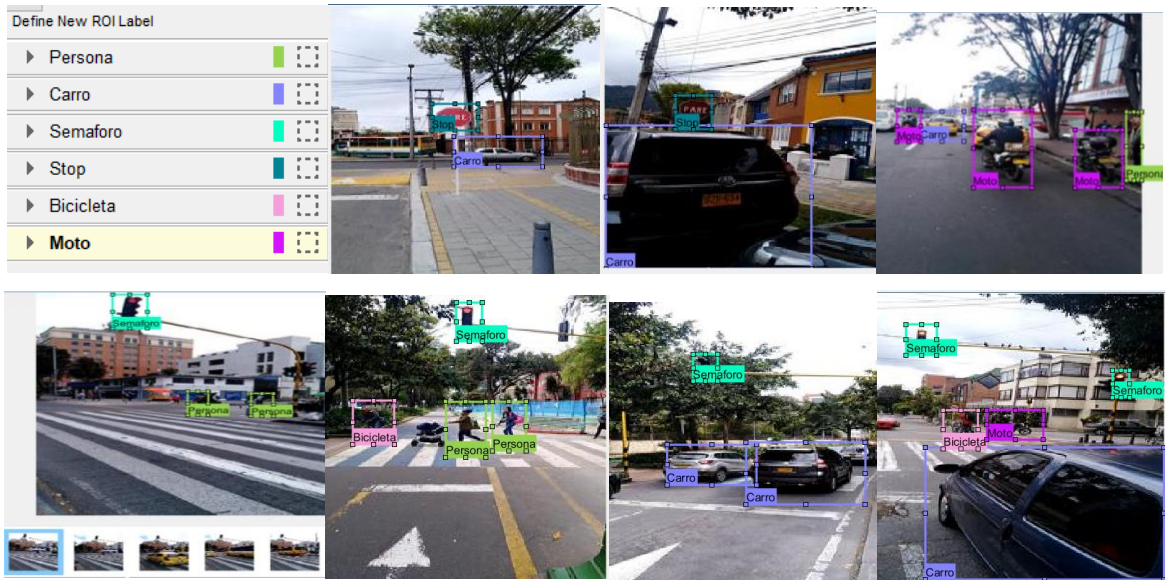



**Figura 38 Etiquetas de ROI de Data set utilizando ImageLabeler**

La información de las 6 etiquetas obtenidas de los 2421 frames, se extraen como data set, donde en la misma imagen, existan objetos redundantes. Por ejemplo, en un mismo fotograma, pueden hallarse, dos semáforos en rojo, tres automóviles, y 3 personas entre otras, las cuales se encierran en cuadrados (bounding boxes) la cual reúne 4 nodos o esquinas, en coordenadas de la ubicación del cuadro que encierra el objeto de interés.

De estas 2421 imágenes se logra etiquetar una cantidad promedio de 1039 semáforos, 634 personas,1482 automóviles, 155 señales de pare o stop, 247 bicicletas y 395 motos (ver **Tabla 9**), los cuales se obtienen a una distancia estimada menor a 10 metros visualmente, y posteriormente se redimensionas a [224 224] pixeles para la evaluación de la CNN.

**Tabla 9 Set de datos etiquetado por cuadros en tabla de pixeles**

| ImageFileName 2421x7 | LABEL | | | | | |
|---|---|---|---|---|---|---|
| 1. | 2.personas | 3. Carros | 4. semáforos | 5. Stop | 6. Bicicletas | 6.Motos |
| D/foldertesis/imagen.jpg | 634 | 1482 | 1039 | 155 | 247 | 395 |

Todos los semáforos (en luz verde y roja), se etiquetan como un mismo ROI (ver **Figura 39**), ya que en el capítulo 5.5, se pretende utilizar otra CNN que trabaja en paralelo, la cual analiza el grupo “semáforo”, y evalúa si está en estado verde o rojo.

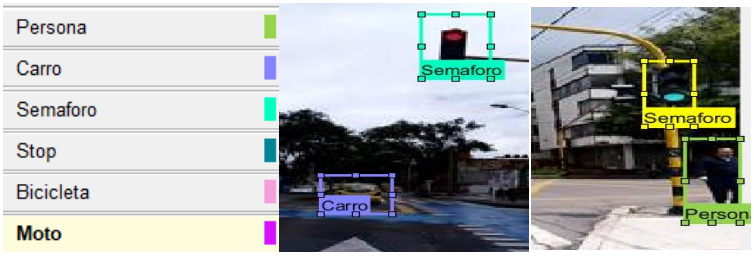



**Figura 39 Etiquetas de semáforos sin diferenciar color**

En conclusión, ejemplificado en la **Figura 40**, se generan tres bases de datos para diferentes aplicaciones, la primera para la utilización de algoritmos que clasifican las imágenes, la segunda base de datos, para la implementación a eventualidades con varios objetos de interés que se presentan utilizando ROI, siendo esta aplicable para algoritmos de aplicación y detección de OI basada en regiones y la tercera para el entrenamiento de semáforos que se verá más detalladamente en el capítulo 5.

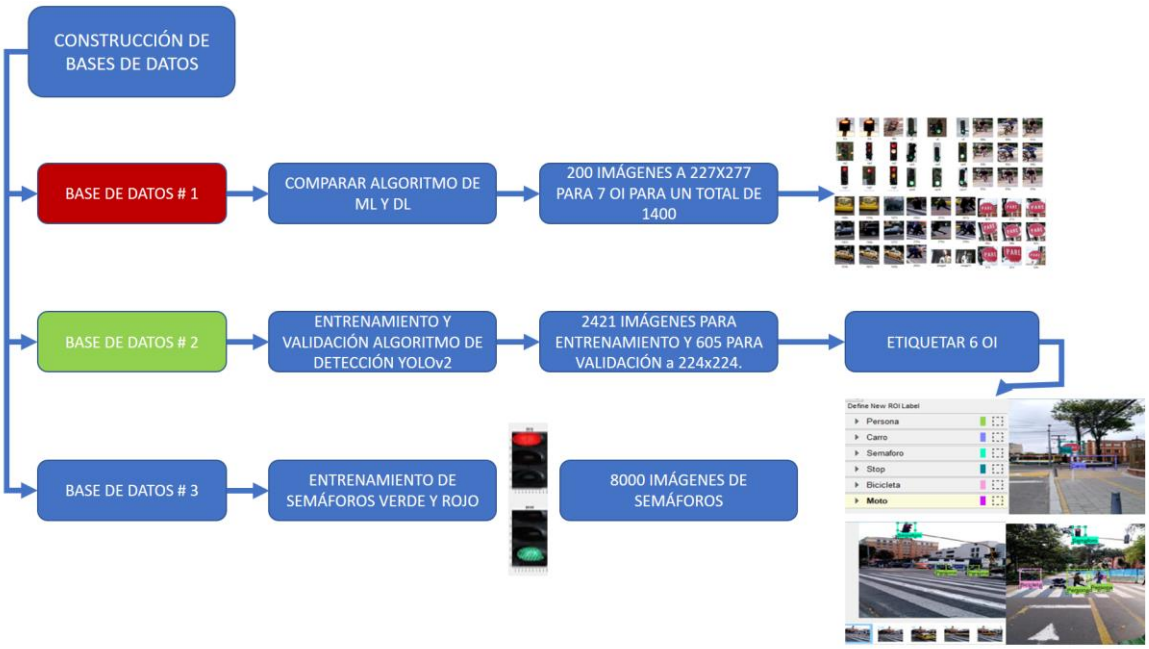



**Figura 40 Creación de bases de datos**

# CAPITULO 4. DESARROLLO DEL AMBIENTE VIRTUAL DE SIMULACIÓN

Para la creación del ambiente virtual se utilizó V-Rep, el cual es un software libre y de fácil acople a Matlab. Se crea un ambiente de simulación que presenta imágenes reales de las calles de Bogotá, mostrando el comportamiento longitudinal para un móvil de características de micro car, y en paredes(walls) se pondrá como textura, imágenes seleccionadas para evaluar diferentes casos de comportamiento longitudinal del coche ante la detección de algún objeto cuando se realice el entrenamiento. Este trabajo no pretende demostrar características físicas, sin embargo los cálculos de cinemática se aproximan para darle realismo a la simulación.

## ○ 4.1 Creación del ambiente virtual

V-REP es un software de simulación multiplataforma desarrollado por Coppelia Robotics GmbH, el cual permite la personalización completa de la simulación mediante diferentes enfoques (complementos, plugin, scripts, etc) y soporta cuatro motores dinámicos/físicos [141]. V-REP permite importar fácilmente diversos formatos de mallas, por lo que las carreteras y los edificios se han cargado mediante archivos OBJ.

Para recrear el entorno virtual se inicia con la importacion de un robot diferencial pionner.Desde este se realizan pruebas del movimiento rotacional de los dos motores. En la parte posterior del robot se ubica una camara RGB-D la cual tomará la informacion del entorno como se evidencia en la **Figura 41**.

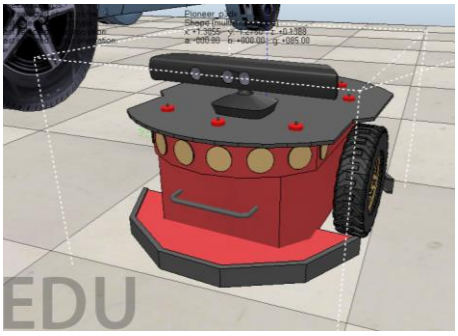



**Figura 41 Uso del Pioneer y Kinect**

El segundo paso es la importacion del prototipo del carro en formato .**OBJ,** el cual cumple con caracteristicas de un coche micro car (**Figura 42**), el cual cumplia con caracteristicas semejantes a 1.50 mt alto, 1.50mt ancho y 3mt largo, con un diametro de llantas de aproximadamente 15'' (38.1 centimetros o 0.381 metros), es decir un radio de 0,19 metros.

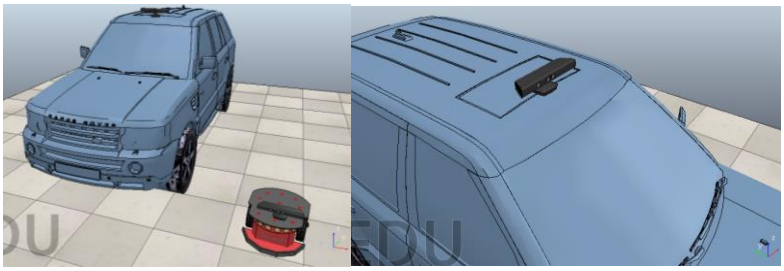


**Figura 42 Importación coche de característica micro car formato .OBJ**

En esta sección se crea el suelo para que el auto cuente con gravedad, y se agrega una pared o Wall. Los walls se pueden modificar en dimensión, textura y su geometría. Al frente del automovil se ubica una pared a la cual, se le agrega una imagen extraida de la base de datos #2 creada en la seccion 3.2, para simular un ambiente mas realista, visualizando el correcto funcionamiento de la camara y el robot. En la pared, se presenta dos de los objetos de interés (**persona y semáforo**), la cual es observada por el Kinect (cámara RGB-D) ubicada sobre el coche, mientras el robot diferencial observa el automóvil. En este caso se presentan dos ventanas por cámara, las cuales, al ser cámaras estéreo, obtienen los componentes en RGB y otro que corresponde a la profundidad, caracterizándose por mostrar una silueta de color azul si se encuentra lejos, y de un color rojizo al acercase frontalmente según su rango de visión como se aprecia en la **Figura 43**.

**Figura 43 Visualización del entorno micro car- Pioneer y cámaras de profundidad**

Considerando que la cámara de profundidad es un Kinect, se caracteriza con un alcance máximo de visión y profundidad a 8 metros de distancia, y la resolución a 224x224 pixeles, ya que, a partir de las características de pixeles, se va a entrenar la arquitectura de la red convolucional YOLO utilizando la resnet50. El criterio de ubicación de la cámara sobre el micro car, considera un alto de 1.5 metros, suficiente para obtener una vista panorámica aceptable que tome del suelo a las luces de tráfico y el punto de contacto del frente (capo) del coche vs el del Kinect, ya que el criterio de parada dependerá de la distancia de la punta del coche vs el objeto de interés. Para conocer el desfase de distancia del Kinect $Dk$ frente al borde del capo se obtiene a través de una resta obtenida de las coordenadas $X$ que se encuentran como posición en el espacio. Ver **Figura 44**.

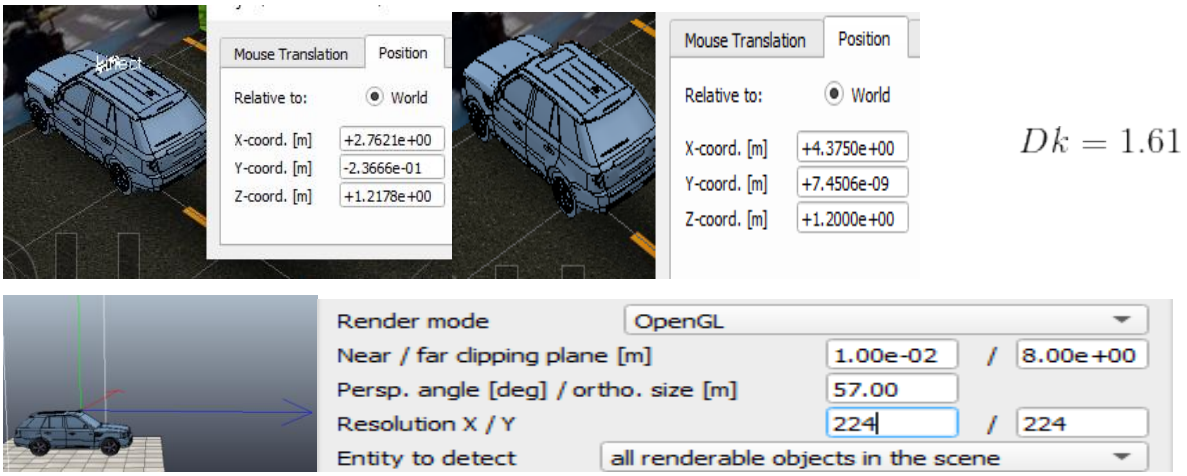



**Figura 44 configuración**

**y localización de cámara de profundidad**

Esto quiere decir que, el auto tiene un largo aproximado de 1.61 metros de la ubicación del Kinect a la punta del capo. Si la distancia en la que el Kinect detecta es de 8 metros, pero el punto de referencia o punta de contacto del capo es de 1.61 metros de desfase, se concluye que realmente el coche tendrá una capacidad de detección por parte del Kinect a partir de 6,39 metros. La simulación pretende visualizar un movimiento angular mayor en las ruedas al encontrarse lejos de un OI, el cual se irá reduciendo a partir de que se acerca. Para aproximar los cálculos como criterio de frenado, se aproximará a 6,50 metros, es decir se tomará de referencia a 1.5 metros la distancia de la cámara al capo del móvil.

En este punto se le implementan cuatro motores, uno a cada rueda del carro simulado, ver **Figura 45**, los cuales variaran la velocidad angular según la distancia a el OI, desacelerandose si se acerca y aumentando si se aleja.


**Figura 45 Adherir 4 motores, uno para cada rueda en V-Rep**

El control de velocidad desde Matlab, se realiza de acuerdo con la velocidad angular y no a la velocidad lineal, siendo $w = VL/r$, donde $VL$ es la velocidad lineal y $r$ el radio de las ruedas del móvil. Para este caso se utilizará una llanta de aproximadamente 15 pulgadas (rin + perfil del neumático tipo Bridgestone), que corresponde a los carros micro car como el Fiat Zastava Topolino. El procedimiento de conversión de unidades para pulgadas a centímetros es:

$$diametro.llanta = 15inch * \frac{1m}{39{,}3inch} = 0{,}381m \qquad r = 0.1905\ m$$

Este punto concluye en observar dos cámaras RGB-D, una encargada de visualizar lo que observa el auto (en este caso concreto una calle con una señal de pare, por otro lado, una cámara que visualiza el comportamiento de las ruedas, ya que la pared se ira acercando o alejando, según este comportamiento las llantas variaran la velocidad ver **Figura 46**. Se agregan varias paredes que corresponden a diferentes escenarios de decisión los cuales se describen en la sección 4.3.


**Figura 46 visualización de cámara del coche y de cámara que supervisa el movimiento angular de las ruedas**

## ○ 4.2 Obtención de Imágenes de prueba para simulación

Se extraen imágenes de las calles de Bogotá, como criterio en la toma de decisiones para diferentes situaciones, ver **Figura 47**. Por ejemplo, "¿Qué haría un conductor, si el semáforo está en estado verde, pero hay un peatón cruzando?", si el coche está atascado en un trancón, como varia la velocidad angular, según la distancia de los coches o motos localizados al frente. Esta toma de decisiones es implementada en el simulador y desarrollada en la sección 6.

**Figura 47 Ilustración fotogramas para detección en simulación**

## ○ 4.3 Creación de interfaz gráfica para visualizar videos y la cámara del simulador

Para que el algoritmo YOLOv2 que se entrena en la sección 5.5 pueda ser visualizado, fue necesario crear una interfaz gráfica (GUIDE) de Matlab, desde donde correrá el algoritmo de la YOLOv2 y como OPC el software V-Rep, capaz de utilizar cámaras RGB-D, necesarias para las pruebas del comportamiento del coche en movimiento longitudinal

frente a los agentes de interés, permitiendo ver la interacción en tiempo real entre el entorno real y virtual. La interfaz grafica creada, cuenta con dos opciones, una para obtener los videos mp4. capturados en la seccion del capitulo 3 y otra para ver a tiempo real lo que visualizan las camaras del simulador. La opcion “Browse”, será la encagrada de al hacer click, ir a la carpeta donde se podra cargar un video con formato mp4, un boton de “Start” para empezar el video , y otro de “stop” para detenerlo y no generar errores como se evidencia en la **Figura 48**, por otra parte se creó un boton boleano de “ENCENDER”, el cual funciona como conector matlab – v-rep, el cual al recibir el click, cambie de color a estado verde y empiece a visualizar lo que ve la camara RGB-D de la simulación.

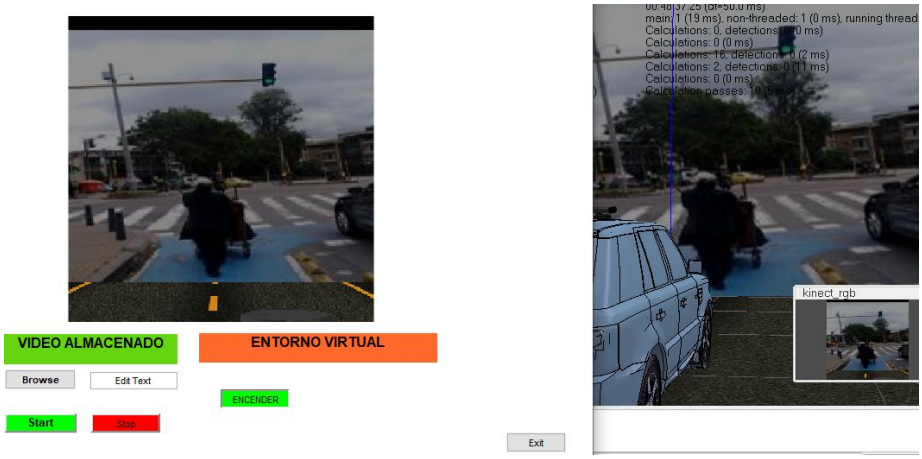



**Figura 48 Visualización de interfaz gráfica de Matlab, lo que visualiza v-Rep**

## ○ 4.4 Reacción del móvil frente a obstáculos de interés

### Parámetros de distancia y velocidad

En la **Tabla 10** se describen dos coeficientes de rozamiento en llantas cuando la calzada esta húmeda y seca.

**Tabla 10 Parámetros de coeficiente de fricción de rozamiento**

| | | | |
|---|---|---|---|
| Jones and Childers informa de unos coeficientes de fricción de alrededor de **0.7** para calzada seca y de **0.4** para la calzada mojada. | **Coeficiente de Fricción de rozamiento** | **húmedo** | **seca** |
| | | 0,4 | 0,7 |
| | | **Promedio 0,55** | |

La magnitud de la fuerza de rozamiento entre dos cuerpos en contacto es proporcional a la normal entre los dos cuerpos, es decir:

$Fr = a = \mu * g$ donde $a$ es la aceleración, que es igual a lo que conocemos como coeficiente de rozamiento $\mu$ ,multiplicado por $g$ que es gravedad obtenido en la **Ecuación** 17.

$$a = \mu * g \qquad (17)$$

Considerando un coeficiente de rozamiento $\mu$ promedio de la llanta en 0,55 como se ve en la **Tabla 10** y la gravedad $g$ una constante de -9,81, obtenemos una aceración negativa de

$$a = -5.4$$

Teniendo como criterio de parada o velocidad final igual a cero, una distancia de 2 metros con respecto a los OI, es decir 4.5 metros (de la resta 6.5m – 2m) desde que detecta el objeto hasta detenerse totalmente. Se halla la velocidad máxima a la que el coche deberá moverse, y así conocer la distancia en la que debe iniciar el frenado. Para esto se acude a la **Ecuación 18** de velocidad en una determinada posición.

$$V_0^2 = V_f^2 - (2 * a * d) \qquad (18)$$

Siendo $V_0^2$ la velocidad lineal inicial, la cual es desconocida, igual a $V_f^2$ la velocidad final que en este caso será igual a (0) cero, $a$ representa la aceleración que en este caso será la fuerza del coeficiente de rozamiento -5.4 hallado en la **Ecuación 17** y la distancia $d$ ,igual a 4.5 metros, que necesita el automóvil en detenerse totalmente, se obtiene : $V_f^2 = 0$, $a = -5.4 \quad d = 4.5m$

Para obtener la velocidad inicial, o la velocidad lineal máxima ($VLm$), se obtiene de la **Ecuación 19** dada por

$$VLm = \sqrt{(2 * a * d)} \qquad (19)$$

$$VLm = \sqrt{|-10.8| * 4.5}$$

$$VLm = 6.968m/s$$

Como criterio de velocidad inicial máxima, se considera le velocidad lineal de 6.968 m/s. A partir de esta velocidad, irá disminuyendo proporcionalmente durante 4,5 metros hasta detenerse.

La velocidad angular máxima ($Wm$) está dada por la **Ecuación 20**, donde $VLm$ es la velocidad lineal máxima y $r$ el radio de las ruedas del móvil el cual es de 0.1905 metros, siendo $Wm$ igual a 36.58 rad/s.

$$w = \frac{VLm}{r} \qquad (20)$$

$$w = (\frac{6.968m}{s})/0.1905m = 36.58 \text{ rad/s}$$

En conclusión, como se ejemplifica en la **Figura 49**, el uso de V-Rep como ambiente virtual de fácil acople a Matlab, permite desarrollos más realistas al poder importar imágenes y plasmarlas en paredes simulando un ambiente cercano a la realidad en condiciones controladas. El uso de motores facilita el control de la velocidad angular en las ruedas para cálculos de la cinemática de frenado y acelerado. El importar fácilmente cámaras de profundidad como el KINECT, y el poder caracterizar la resolución de la adquisición de capas RGB como la caracterización de la distancia, permite cálculos que se acercan a la realidad según los componentes utilizados.

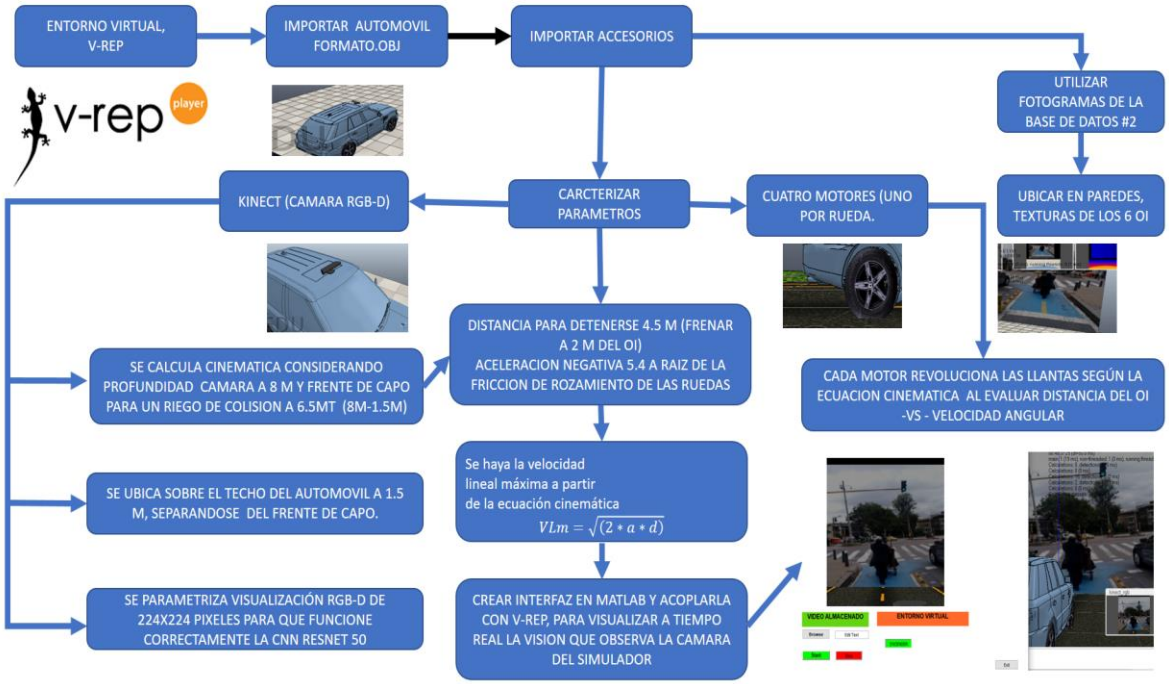



**Figura 49 Creación de ambiente virtual acople con Matlab**

# CAPITULO 5. EVALUACIÓN, SELECCIÓN E IMPLEMENTACIÓN DEL ALGORITMO DE CLASIFICACIÓN Y DETECCIÓN.

Para la selección del algoritmo a utilizar (técnicas de ML o DL), se utiliza la base de datos #1, creada en el capítulo 3.1 para el entrenamiento de las 7 clases, carros, señal de pare, bicicletas, motos, personas, semáforo verde y semáforo rojo, donde cada clase fue separada manualmente en carpetas. La selección del algoritmo se realiza a partir de tres etapas. Primero, se evalúa la posibilidad de clasificación a partir de técnicas de supervisión utilizando técnicas de **extracción manual** de características en algunos objetos de interés (semáforos verdes, rojos y personas). La segunda etapa se realiza con el entrenamiento de varios algoritmos clásicos de ML utilizando el data set # 1 creado y finalmente en una tercera etapa, se realizó un entrenamiento con la misma base de datos, utilizando redes neuronales convolucionales. La respuesta de clasificación será evaluada a partir de matrices de confusión y probada con reconocimiento a tiempo real usando una cámara. El trabajo enfatiza en el reconocimiento de señales y agentes de tránsito de Bogotá. Se ejemplifica en la **Figura 50** el comportamiento de entrenamiento y validación para la etapa dos y tres de este capítulo, donde hay una base de datos como entrada, se extraen filtros y se obtiene al finalizar un modelo entrenado, el cual se evalúa con imágenes de prueba.

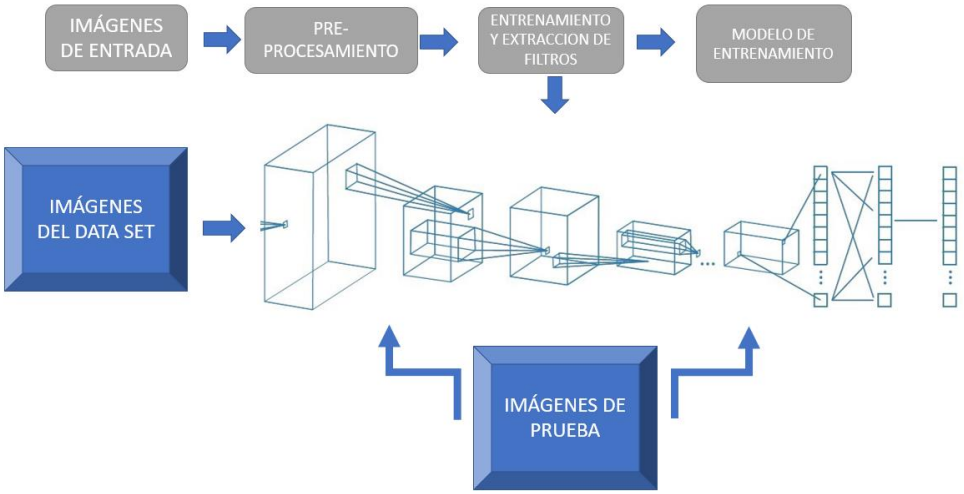



**Figura 50 Proceso de entrenamiento y validación de CNN. Fuente propia**

## ○ 5.1 Extracción de características utilizando procesamiento de imágenes

**Etapa 1**

Inicialmente se hace una evaluación de procesamiento de imágenes para revisar cómo atacar el problema. En este caso se toman dos imágenes de 100 x 350 pixeles de semáforos. Se pasa un vector fila por la coordenada Y 70, para evidenciar los componentes RGB en el semáforo rojo y se repite el proceso para el semáforo verde desde la coordenada Y del pixel 260 respectivamente como se evidencia en la **Figura 51**. Se aprecian dos gráficos, el primer grafico evidencia más componente R (rojo), en el segundo grafico se evidencian más componentes verdes. De cualquier forma, se evidencia una forma arcaica de extraer caracteres, las cuales son más útiles para el procesamiento de imágenes frente a la necesidad a cumplir la cual es el reconocimiento de patrones.

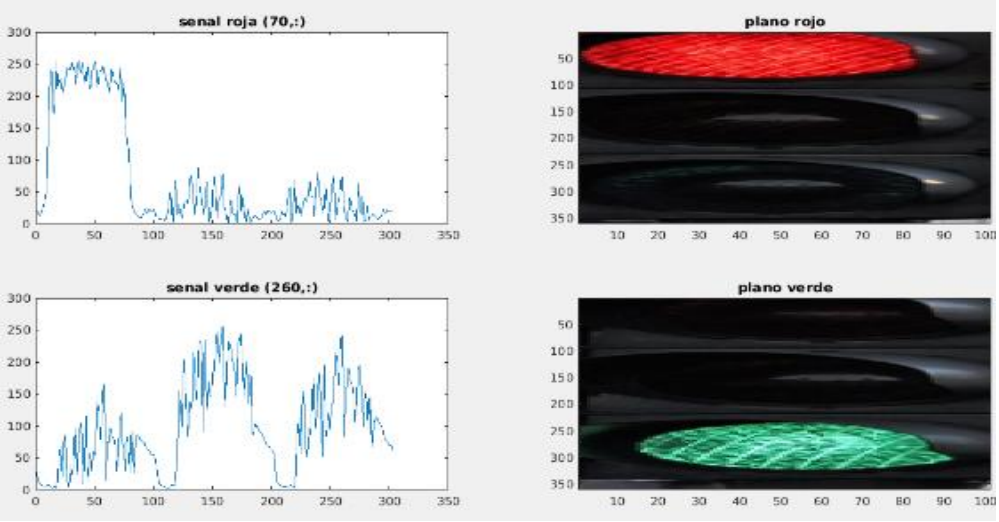



**Figura 51 Diferencia de color por Procesamiento de imágenes**

Posteriormente se realiza la extracción de características de una forma más practica utilizando la herramienta filter matches de Matlab. Como se evidencia en la **Figura 52** se puede visualizar a la izquierda de la imagen, líneas de color amarillo que corresponde a la cantidad manual de características extraída de un semáforo verde (arriba) y un semáforo rojo (abajo), los cuales se encierran en un recuadro de color verde en la sección derecha

comentando cual es el objeto detectado. Desafortunadamente, cuando se modificaba la imagen en brillo o en rotación ya no la reconocía el objeto de interés.

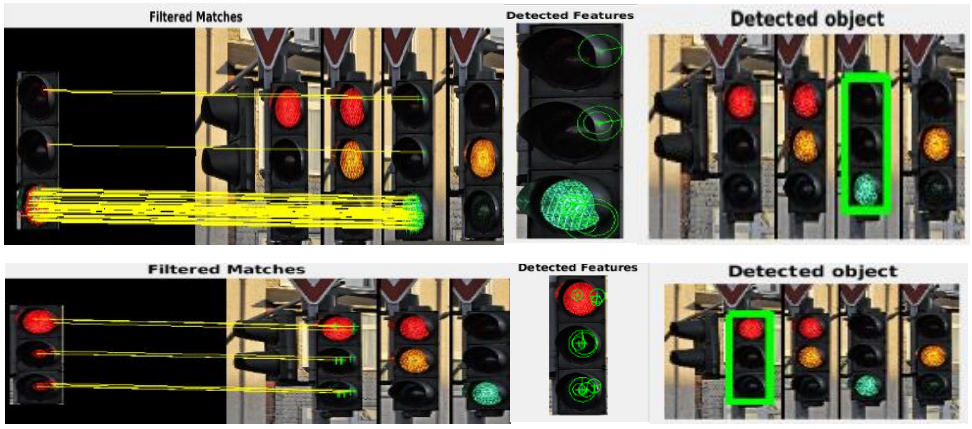



**Figura 52 Experimento con características en semáforos con Filter Matches**

Se realizan el mismo ejercicio de extracción de caracteres para rostros utilizando el Filter Matches y el algoritmo de viola Jones como se presenta en la **Figura 53** sin embargo este también evidencia que al moverse el sujeto ya no cumplía con el reconocimiento esperado

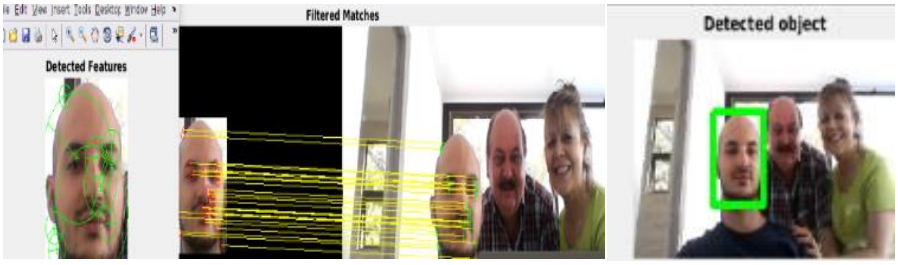



**Figura 53 Extracción de características para personas**

En conclusión, por procesamiento de imágenes y extracción manual de caracteres, no es suficiente como alternativa para el trabajo, ya que, al alterarse la imagen, deja de reconocer las características, por esto se acuden a el entrenamiento de algoritmos sofisticados de ML en la sección 5.2 y 5.3.

## ○ 5.2 Reconocimiento y clasificación de imágenes, utilizando técnicas clásicas de ML

**Etapa 2**

Una vez evidenciada la no practica utilización de la extracción de características manual utilizando el filter Matches para reconocer y clasificar imágenes de video, se realiza la implementación de varias técnicas clásicas de Machine learning con la base de datos creada en la sección 3.1, obteniendo automáticamente decenas de miles de características para cada etiqueta como se presenta en la **Figura 54** para los 7 agentes.

```
* Image category 1: Motos          from 199 images in image set 1...done. Extracted 10652 features.
* Image category 2: bicicletas     from 194 images in image set 2...done. Extracted 22744 features.
* Image category 3: carros         from 199 images in image set 3...done. Extracted 19823 features.
* Image category 4: pare           from 189 images in image set 4...done. Extracted 18108 features.
* Image category 5: personas       from 187 images in image set 5...done. Extracted 18148 features.
* Image category 6: semaforo rojo  from 183 images in image set 6...done. Extracted 4264 features.
* Image category 7: semaforo verde from 181 images in image set 7...done. Extracted 5142 features.
```

**Figura 54 Extracción de características en 7 clases con Classification learner**

Se utiliza la app de **Classification learner** de Matlab donde se evalúan diferentes algoritmos de ML, entre ellos son testeados los algoritmos *Árbol de decisión, discriminante lineal, máquina de vector soporte cuadrático, KNN vecino más próximo, y gaussiano con SVM.* Una vez extraídas las imágenes para evaluar las 7 clases se obtiene un Accuracy(precisión) por cada técnica utilizada como se observa en la **Figura 55**, para árbol de decisión un 55.6%, el discriminante lineal 82%, el KNN 82% y el gaussiano por Bayes un 16,5%. En este caso se escoge el de mayor precisión con un **84.2%**, para este caso, el algoritmo de SVM cuadrático, el cual tuvo un tiempo de procesamiento de 4.08 segundos.

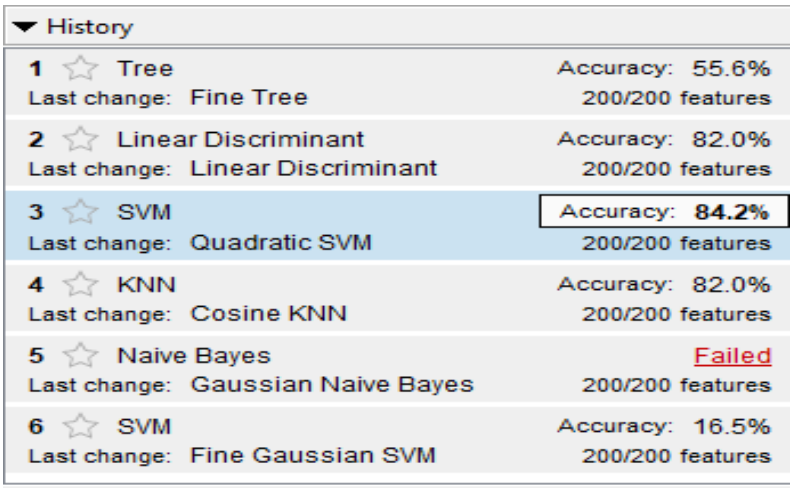



**Figura 55 Selección de técnicas de Machine Learning para la clasificación de 7 agentes**

Tomando un criterio de validación del 20% aleatorio de las imágenes de la data set, se obtiene una matriz de confusión de las 7 clases de transito obteniendo dos resultados correspondientes a falsos positivos (color rojo) y verdaderos positivos (color verde). Ver **Figura 56**

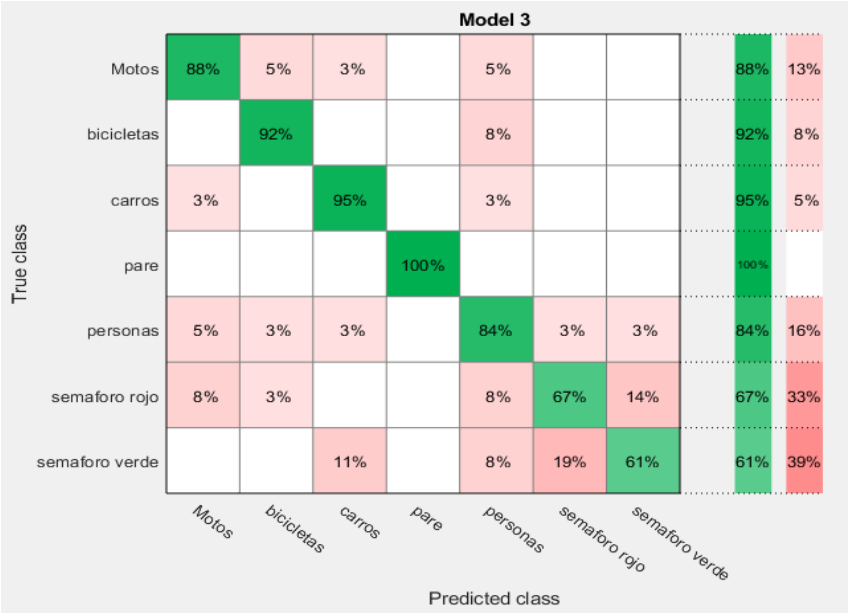



**Figura 56 Matriz de confusión falsos positivos - verdaderos positivos con ML**

posteriormente se computan los resultados en la **Tabla 11** y se comprueba en video la fiabilidad exponiendo imágenes en la cámara, evidenciando el comportamiento de reconocimiento, para este caso un semáforo en verde tuvo un 61% de precisión y un coche un 95% como se ve en la **Figura 57**

**Tabla 11 Datos de imágenes por cámara a tiempo real con ML**

| CLASE | Verdaderos positivos | Falsos positivos |
|---|---|---|
| | SVM | SVM |
| MOTOS | 88% | 12% |
| BICICLETAS | 92% | 8% |
| CARRO | 95% | 5% |
| PARE | 100% | 0% |

| | | |
|---|---|---|
| **PERSONAS** | 84% | 16% |
| **S. ROJO** | 67% | 33% |
| **S. VERDE** | 61% | 39% |


**Figura 57 Visualización y reconocimiento de imágenes por cámara a tiempo real con ML**

Una vez se obtiene como mejor clasificador de los siete agentes al algoritmo SVM con un 84.2% de precisión (técnica clásica de ML), se procede en la sección 5.3 realizar el entrenamiento de los mismos objetos de interés con una técnica de aprendizaje profundo para posteriormente comparar el de mejor precisión.

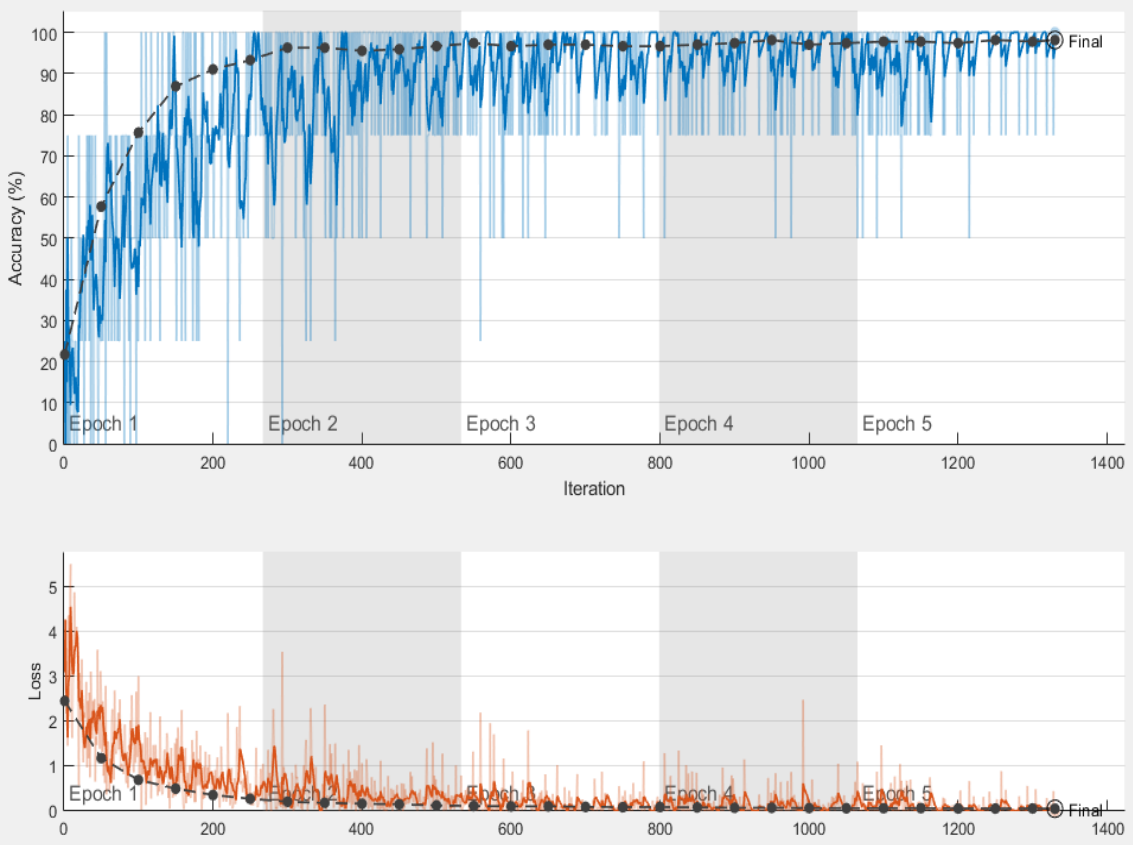
## ○ 5.3 Reconocimiento y clasificación de imágenes, utilizando Deep learning con una CNN.

**Etapa 3**

Una vez realizado el experimento de reconocimiento con técnicas clásicas de machine learning, se continua con un segundo algoritmo utilizando técnicas de Deep learning. De acuerdo a la literatura los avances obtenidos en el concurso ImageNet 1000, hay superioridad en precisión, por parte de las redes neuronales convolucionales en reconocer patrones. En este caso se utiliza la base de datos #1 para el entrenamiento de los 7 objetos de interés. Se verifica que todas las imágenes estuvieran redimensionadas a 227x227 pixeles para poder hacer uso de la transferencia de conocimiento, en este caso utilizando la CNN Alex Net, compuesta por 25 capas con los pesos de filtros aprendidos, lo que

reducirá el tiempo de aprendizaje para los nuevos OI. Se modificó su antepenúltima capa (#23), llamada Softmax, encargada de obtener las siete (7) clases de las imágenes almacenadas de la base de datos #1 y no las mil(1000) categorías pre aprendidas.

En este caso se utilizan como parámetros, una tasa de aprendizaje (learning rate) de 1^-5 para alcanzar el mínimo global, un minibatch de cuatro (4) (para reducir el tiempo de aprendizaje), y cinco (5) épocas que demuestran un buen comportamiento al evidenciar que el error tiende a cero y el Accuracy al 100% desde la 3ra época en la iteración 800 donde ya se estabiliza la curva de aprendizaje. Ver **Figura 58**




| RESULTS | | | |
|---|---|---|---|
| Validation accuracy | | 98.13% | |
| Elapsed time | | 12min-43 sec | |
| **Training Time** | | **Validation** | |
| Epoch | 5 | frequency | 50 iteration |
| iteration | 1330 | harware Resourse | GPU Nvidea 920Mx |
| Iteration per epoch | 266 | Learning rate | 1E-05 |

**Figura 58 Datos de entrenamiento precisión y perdidas en DL con Alex Net**

Así como se realizó con las técnicas clásicas de machine learning, para esta técnica de CNN también se obtiene la matriz de confusión ejemplificado en la **Figura 59**, y **Tabla 12,** la cual evidencia un porcentaje de precisión mayor al 95% en todas las categorías el cual utilizó 80% de la data set para entrenamiento y el 20% de imágenes aleatoria para la validación y resultados.

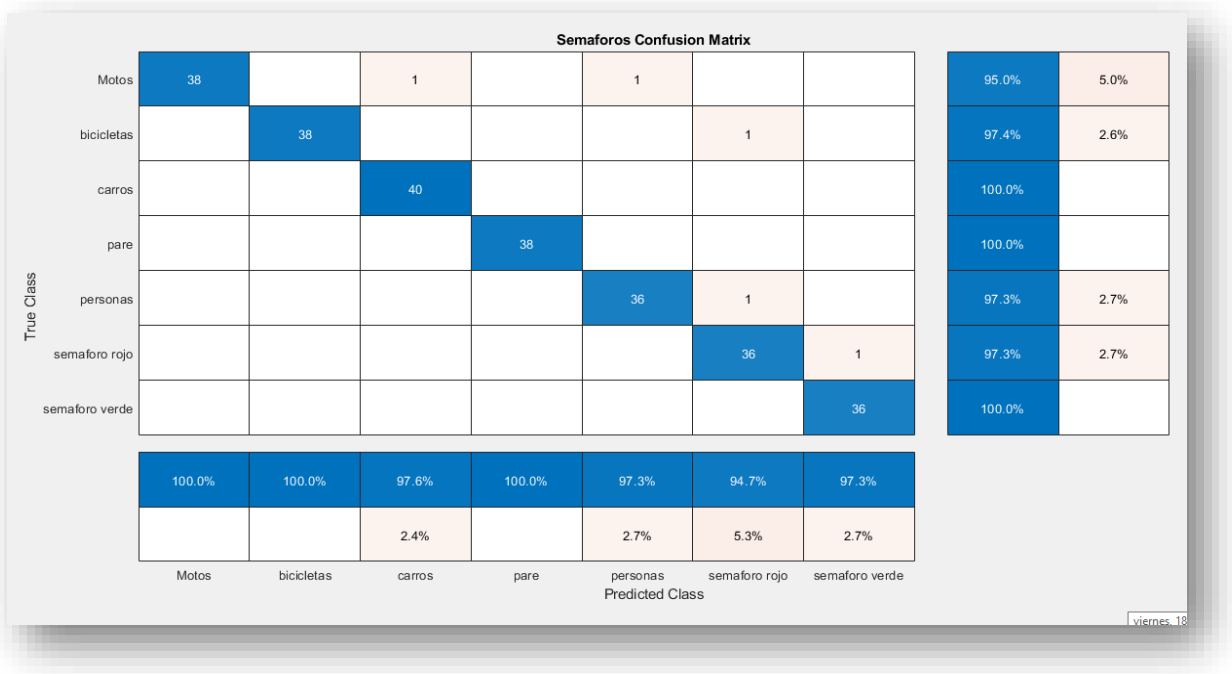



**Figura 59 Matriz de confusión falsos positivos - verdaderos positivos con DL**

**Tabla 12 Datos Matriz de confusión falsos positivos - verdaderos positivos con DL**

| CLASE | Verdaderos positivos | Falsos positivos |
|---|---|---|
| | CNN ALEXNET | CNN ALEXNET |
| MOTOS | 95,0% | 5,00% |
| BICICLETAS | 97,4% | 2,60% |
| CARRO | 100,0% | 0,00% |
| PARE | 100,0% | 0,00% |
| PERSONAS | 97,3% | 2,70% |

| | | |
|---|---|---|
| **S. ROJO** | 97,3% | 2,70% |
| **S. VERDE** | 100,0% | 0,00% |

Se computa la información de la matriz en la **Tabla 12**, posteriormente se evidencia por ejemplo que identificando personas tuvo un 97,3% y señales de pare del 100% al utilizar la validación. En la **Figura 60** se evidencian las 25 capas que conforman la taxonomía de esta CNN Alex Net, ganadora en 2012 en el concurso ImageNet. Se realizan pruebas en video a tiempo real (ver **Figura 61**), demostrando una respuesta rápida y confiable donde varia en menor grado los histogramas. En este caso se hace el experimento exponiendo dos imágenes una de semáforo verde y una señal de pare.

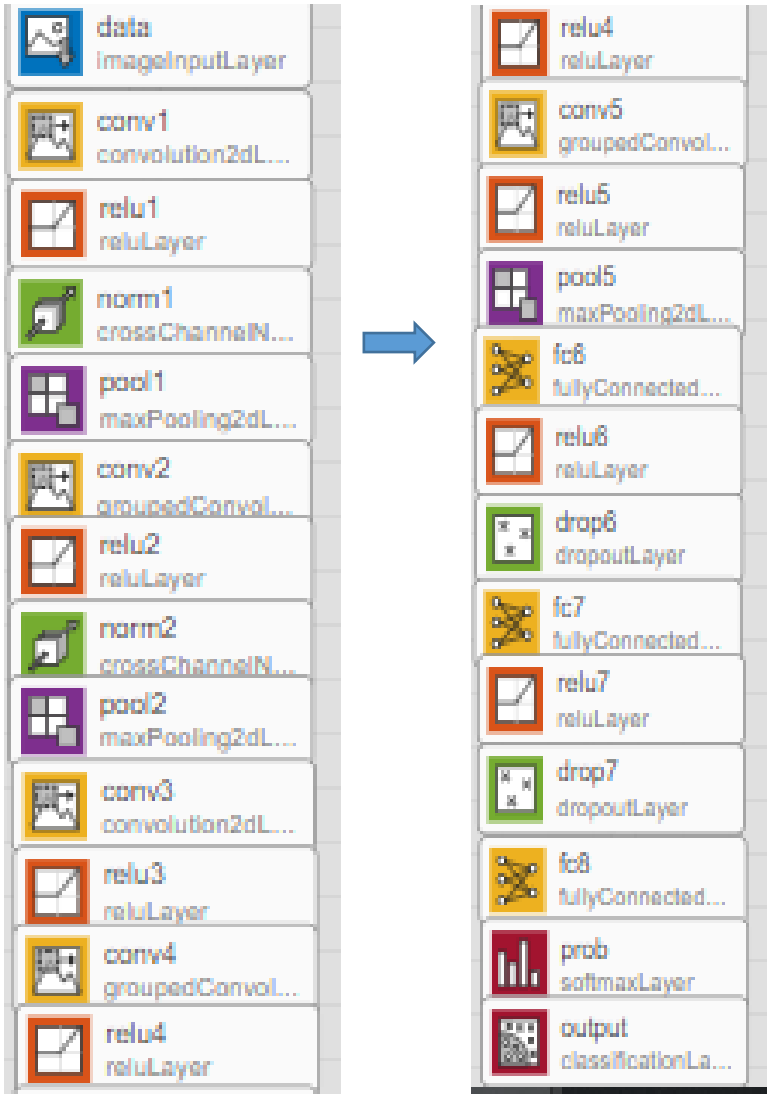


**Figura 60 Capas CNN Alex Net fuente: propia**

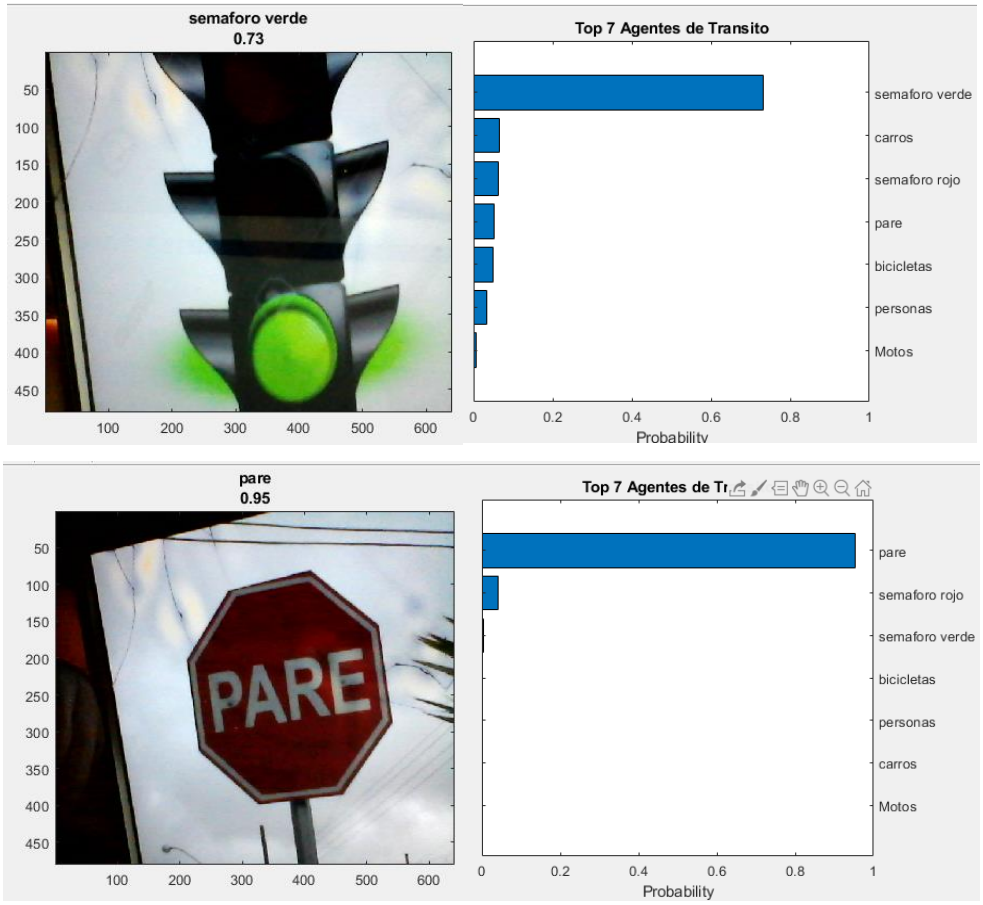



**Figura 61 Visualización y reconocimiento de imágenes por cámara a tiempo real con DL**

Los ejemplos realizados anteriormente fueron con el fin de recrear una evaluación en reconocimiento de agentes utilizando técnicas clásicas de aprendizaje de máquina y redes neuronales convolucionales apalancado con transfer learning[4].

[4] EL transfer learning, es la utilización de redes neuronales convolucionales creadas y que se pueden descargar con AlexNet, GoogleNet, o Res Net.

## ○ 5.4 Evaluación y selección de algoritmo de clasificación según las matrices de confusión con la base de datos #1.

En la **Tabla 13** se aprecian dos categorías (Verdaderos positivos y Falsos positivos), los cuales muestran porcentualmente su grado de efectividad en clasificar cada clase, tomando como parámetro que el índice es positivo, si el porcentaje obtenido en verdaderos positivos se acerca al 100% y en falsos positivos al 0% obtenidos en la matriz de confusión. Los algoritmos presentados son dos, el de ML (SVM) y de DL (Alex Net). Los datos del algoritmo de ML, se desarrolla en la sección 5.2, y el algoritmo de DL en la sección 5.3, el cual compara al de mejor rendimiento.

**Tabla 13 Comparación de rendimiento en reconocimiento de agentes entre DL y ML utilizando matriz de confusión**

| | VERDADEROS POSITIVOS | | | FALSOS POSITIVOS | | |
|---|---|---|---|---|---|---|
| **CLASE** | **ML** | **DL** | **DIFERENCIA (DL-ML) /ML)** | **ML** | **DL** | **DIFERENCIA ML-DL** |
| **MOTOS** | 88% | 95,00% | 7,95% | 12% | 5,00% | 7% |
| **BICICLETAS** | 92% | 97,40% | 5,87% | 8% | 2,60% | 5% |
| **CARRO** | 95% | 100,00% | 5,26% | 5% | 0,00% | 5% |
| **PARE** | 100% | 100,00% | 0,00% | 0% | 0,00% | 0% |
| **PERSONAS** | 84% | 97,30% | 15,83% | 16% | 2,70% | 13% |
| **S. ROJO** | 67% | 97,30% | 45,22% | 33% | 2,70% | 30% |
| **S. VERDE** | 61% | 100,00% | 63,93% | 39% | 0,00% | 39% |
| **Precisión general** | ML<br>DL | 84,20%<br>98,13% | **16,54%** | | | |

En esta etapa se demostró superioridad en las redes convolucionales frente a técnicas clásicas de ML. La técnica de DL ilustra la superioridad en la clasificación de imágenes para las 7 categorías con la CNN frente a la técnica de ML que utilizó el algoritmo de SVM, el cual se evalúa a partir de las dos matrices de confusión. La comparación realizada detectando **verdaderos positivos**, se calcula como ((DL-ML) /ML) para cada categoría, demostrando por ejemplo que reconociendo personas la CNN, evidencia una precisión de un 15,83% superior, frente a la técnica SVM de machine learning. Para el cálculo de la diferencia obteniendo falsos negativos, se genera una resta, evidenciando en qué

porcentaje se redujo la clasificación errónea de DL frente al ML, mejorando por ejemplo en un 7% al clasificar motos la CNN Alex Net, frente a la técnica de ML, el SVM.
La precisión total de la CNN, fue de 98.13%, frente a la mejor técnica de ML clásica, en este caso el SVM, el cual obtuvo un 84.2%, superándola en un **16,5%** en precisión general.

## ▪ Requerimientos del sistema

Una vez se obtiene el algoritmo de clasificación, se revisan los parámetros necesarios para que el algoritmo a utilizar clasifique y detecte los agentes, para esto se es necesario las consideraciones técnicas de software y hardware a utilizar.

De acuerdo a [116] los requerimientos parten de los estándares ISO / IEC / IEEE 29148: 2011, como requisito que garantice un ciclo de vida del sistema. El presente proyecto se enfoca principalmente en el diseño de un asistente de conducción para un coche utilizando reconocimiento y detección de agentes de tránsito, en base a esto acelera o frena. Por tal es requisito, revisar las prioridades para probar compatibilidad. Ver **Tabla 14**:

**Tabla 14 Requerimientos de arquitectura**

| | | Prioridad | | |
|---|---|---|---|---|
| Numeral # | **Software** | alta | media | baja |
| 1 | Requiere un sistema operativo de distribución libre | | x | |
| 2 | Requiere un software de tratamiento de un gran bloque de imágenes y archivos de texto. | x | | |
| 3 | Requiere un software de programación compatible con el software de tratamiento de imágenes. | x | | |
| 4 | Requiere un Data set compatible con la arquitectura de red neuronal convolucional. | x | | |
| 5 | Requiere una arquitectura de red neuronal convolucional con bajo consumo computacional. | x | | |
| 6 | Requiere un software de tratamiento de imágenes para representar la salida del detector | x | | |
| | **Hardware** | | | |
| 7 | Se necesita un equipo que contenga GPU | x | | |

- **Selección software y hardware**

Una vez realizada la tabla de prioridades, se considera la evaluación a continuación. De acuerdo al numeral 1, el sistema operativo a utilizar es Windows gracias a la facilidad en uso y acople entre Matlab y V-Rep.

Se escoge Matlab 2019A, debido a que es un entorno que facilita el entrenamiento, cuenta con las librerías de la arquitectura **YOLOv2** y soporta la **interoperabilidad** con marcos de deep learning de código abierto a través de las capacidades de importación y exportación de ONNX. Sumado a ello presenta mejores resultados en la velocidad de procesamiento, frente a librerías como Tensor Flow o Caffe2 que se pueden visualizar en la **Figura 62**, además de facilitar la creación de base de datos, con el uso de la app ImageLabeller,

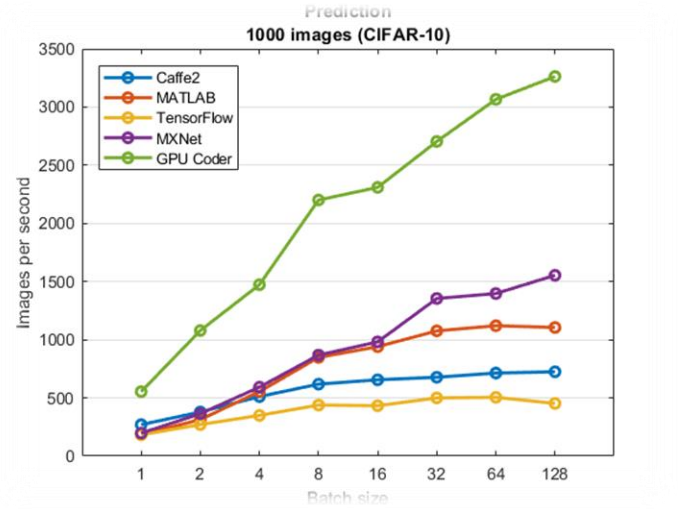



**Figura 62 Predicción en CIFAR -10 utilizando Matlab vs Tensor Flow**

La referencia en cuanto al numeral 4, según el Data set Pascal VOC [61], ha demostrado en diversos trabajos la detección de objetos más centrados en las imágenes, facilitando un gran volumen de información frente a otros data set [136]. La unión más común de datos de Pascal es el uso de trainval 2012 para entrenamiento y validación [61].

Como numeral 5 se evaluaron diferentes arquitecturas de red para detección de objetos en tiempo real que generen un bajo coste computacional, recursos limitados de memoria y procesamiento. Las arquitecturas evaluadas fueron la Faster RCNN y la YOLOv2. Inicialmente se realiza un entrenamiento utilizando la arquitectura de la red Faster RCNN, sin embargo se observa que el procesamiento es ineficiente, ya que requiere de 1.98 segundos de procesamiento por frame para detectar los objetos de interés, lo que la hace

ineficiente para la implementación a tiempo real, sumado a ello las regiones de entrenamiento en las intercepción sobre el objeto, no son las esperadas como se visualiza en la **Figura 63**, por lo que se decide tomar la arquitectura YOLOv2 siguiendo también el estado del arte.

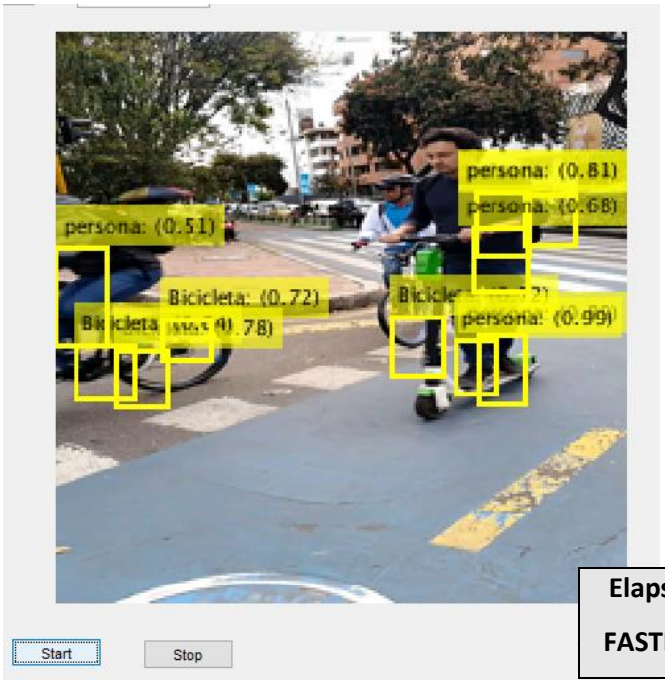



**Figura 63 Velocidad Procesamiento en video utilizando red Faster RCNN**

Para el numeral 6 las imágenes etiquetadas utilizando ImageLabeller son de fácil exportación como tabla, para posteriormente ser evaluada. Como numeral 7 se requirió un computador con GPU, tarjeta gráfica NVidia 920MX, ya que un computador con solo CPU no podría realizar estas operaciones, y el entrenamiento podría ser enormemente más demorado.

## ○ 5.5 Implementación de la red Neuronal Convolucional YOLOv2 con la base de datos #2

Debido a la gran cantidad de herramientas y flexibilidad que ofrece Matlab, para evaluar resultados, utilizar diferentes arquitecturas y un etiquetado sencillo con el uso de la interfaz ImageLabeler. Una vez se conoce la precisión de las técnicas de DL, frente a las de ML, se implementó la arquitectura YOLOv2 la cual es entrenada con la base de datos # 2 creada en el capítulo 3.2. El rendimiento de la red YOLOv2 entrenada se midió en función de sus

métricas de calidad y velocidad. La arquitectura YOLO utiliza filtros de 64, 192, 128, 256, 512 y 1024 en sus capas, las cuales realizan operación de reducción y convolucionales para obtener las activaciones y parámetros de la red. Esta base de datos, se entrena con la red profunda **RESNET50** debido a sus filtros de 3x3 haciéndola liviana, una arquitectura de 170 capas que da una alta fiabilidad y procesamiento rápido debido a las capas residuales. En este proceso la Resnet50 clasifica la categoría de las 6 clases, mientras que la arquitectura YOLO se encarga de la detección, encerrando en recuadros (ROI) a los OI que se desean localizar. Una vez se obtiene el entrenamiento, se anexa en paralelo el entrenamiento de la CNN Alex Net, la cual tiene como única función la clasificación de semáforo en estado **verde y rojo** (ver **Figura 64**) sobre la arquitectura de deteccion. La arquitctura se utiliza para procesar videos formato mp4 y lo recibido a travez de la camara RGB-D del simulador de V-Rep, el cual se acopla con Matlab (software que almacena el detector YOLO).

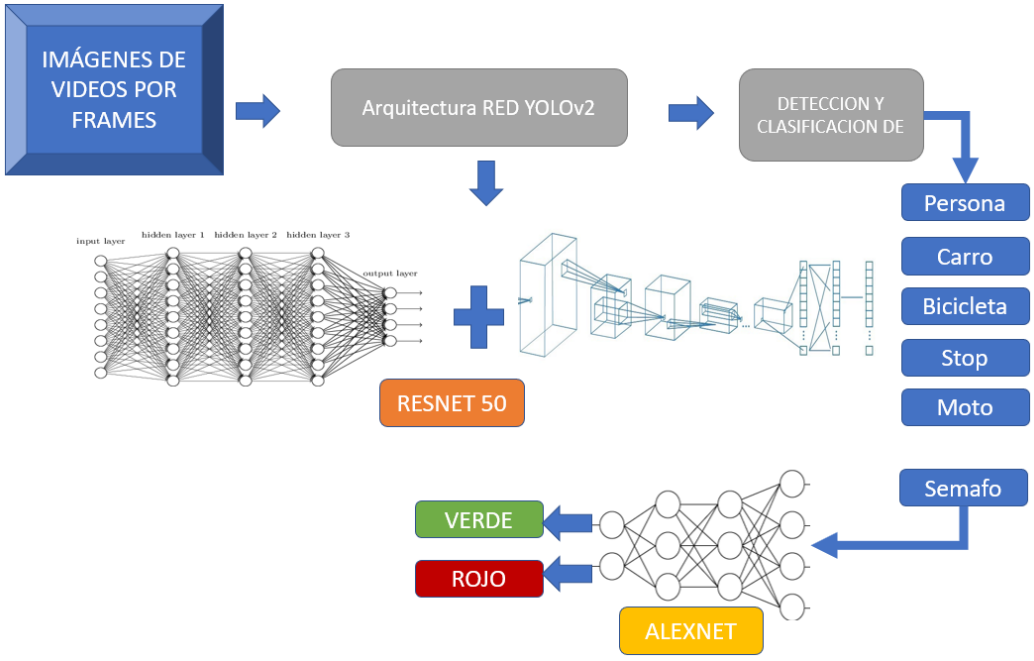



**Figura 64 Arquitectura de reconocimiento YOLO con Alex Net. Fuente Propia**

La entrada de la red divide la imagen en 7x7 recuadros, en donde la red neuronal RESNET 50 es aquella que procesa las imágenes. Para el entrenamiento de la arquitectura, se toman las 2421 imágenes extraídas de la base de datos #2 creado en la sección 3.2, la cual contiene más de 4000 etiquetas, las cuales deben ser redimensionadas a 224x224x3,

consideramos las 6 etiquetas, se establecen las dimensiones de los Anchors es decir posibles características en alto y ancho de los ROI, se llaman 2 minibatch, y se utilizan 30 épocas las cuales duran 11 horas y 45 minutos entrenando. (ver **Tabla 15)**

Como se visualiza en la **Figura 64**, el código se desarrolla de la siguiente forma. Se carga la base de datos con el tamaño de entrada de la base de datos. Luego el número de categorías que en este caso serán 6 y en la activation layer se utiliza la capa de activación número 40 para extraer desde ahí las características, para ahí implementar la YOLO. Posteriormente se proponen unos Anchors, que en este caso **son [43 59;18 22; 23 29; 84 109**], resultado de buscar una proporción en los datos de base x altura en los recuadros. Para el entrenamiento en Matlab de la arquitectura, se deben describir: El nombre de la arquitectura, que en este caso es yolov2Layer, numero de clases = 6, Anchor, la CNN a utilizar =resnet50, y en que parte de esa CNN entra la YOLO, que en este caso es la Relu 40 (la cual corresponde a una función de activación de la CNN resnet50, y se recomienda ubicar a la mitad o el final de la CNN. Esta arquitectura de CNN, se inspecciona utilizando la app "**deep network designer**" y así conocer las capas de esta u otra CNN, por ejemplo, para la renet18 seria res5b_relu).

**Tabla 15 Número de parámetros y operaciones de las capas convolucionales**

| Nombre de las capas | N Capas | Resolución Imágenes |
|---|---|---|
| **conv 1** | 1 | 224x224 |
| **conv 2** | 1 | 112x112 |
| **conv 3, conv 4, conv 5, conv 6** | 4 | 56x56 |
| **conv 7, conv 8, conv 9, conv 10, conv 11, conv 12, conv 13, conv 14, conv15, conv 16** | 10 | 28x28 |
| **conv 18, conv 19, conv 20, conv 21, conv 22** | 6 | 14x14 |

| | | |
|---|---|---|
| **conv 23, conv 24** | 2 | 7x7 |

Bounding boxes o cuadros delimitadores Joseph Redmond YOLO. La dimensión de la malla x (5+la cantidad de clases). Para cada recuadro se le asignan los valores centro de masa del objeto, dimensiones del cuadro delimitador, 1 o 0 si se encuentra y una dimensión por cada clase, es decir 6 clases y 5 para un total de 11. La IOU es la división del área ocupada sobre los recuadros de la malla donde se encuentra, y se tendrá en cuenta si supera el 50%. Sin embargo se mejora con el non Max suppression, el cual con apoyo del IoU, toma los recuadros que tomen el área de probabilidad, y para evitar gran cantidad de boxes para cada clase, solo extrae el de mayor probabilidad, este se puede arreglar aún más con los Anchors. Los Anchors (anclajes), representan una multiplicación a toda la dimensión, en este caso si al haber 4 anclajes, se obtiene un arreglo de 4 x 11. Este busca encajar en el pc al que pertenezca para evitar confusiones. Los Anchors mejor seleccionados son a través del algoritmo vecino mas próximo, el cual no se utilizó en este trabajo.

Las operaciones que realiza la red neuronal YOLOv2, influyen en el costo computacional. Gracias a que las imágenes disminuyen en resolución, el número de operaciones también. La reducción de hiperparámetros con las capas de Pooling, son capaces de controlar el sobreajuste presentadas en la **Tabla 16**.

**Tabla 16 Activaciones de las capas Pooling**

| Nombre de las capas pooling | N Capas | Resolución Imágenes |
|---|---|---|
| **Pool 1** | 1 | 224x224 |
| **Pool 2** | 1 | 112x112 |
| **Pool 6** | 1 | 56x56 |
| **Pool 16** | 1 | 56x56 |

- **Acople entre el GUIDE y el entorno virtual**

Para poder identificar varios elementos al mismo tiempo y demarcando el ROI, se requirió la utilización de arquitecturas la YOLOv2 (You Only look Ones) [98] la cual se escogió debido a sus buenos resultados a tiempo real. En la **Figura 65**, se evidencia como ejemplo en una imagen, el reconocimiento de tres OI, semaforos, personas y carros (en la parte de la izquierda esta la interfaz creada en un GUI. de matlab y a la derecha, la simulacion de v-rep). El guide en matlab funciona inicialmente tomando un frame (im) redimensionado a 224 x 224,que asignan diferentes variables. El detector se representa como [bbox, score, label]=detect (detector, im); los cuales se dividen en bbox= Representa el cuadro que encierra la etiqueta, Score= Es la puntuación en semejanza a la categoría, Label= hace referencia a la clase a la que pertenece. Luego entra en un ciclo for con una excepción para que aúnque no reconozca nada no genere error. En el for el video es procesado como frames , arrojando los label que halla.


**Figura 65 Visualización cámara RGB-D desde Matlab**

- **Pruebas en Video de reconocimiento sin discriminar color de semáforo.**

Continuando con las pruebas del rendimiento de la arquitectura YOLOv2 a tiempo real en videos, presenta incompleta la información de semáforos, es decir la red solo detecta el semáforo, mas no diferencia el color como se ve en la **Figura 66**, en donde se logra ver que identifica carros, motos y semáforos correctamente sin discriminar color.

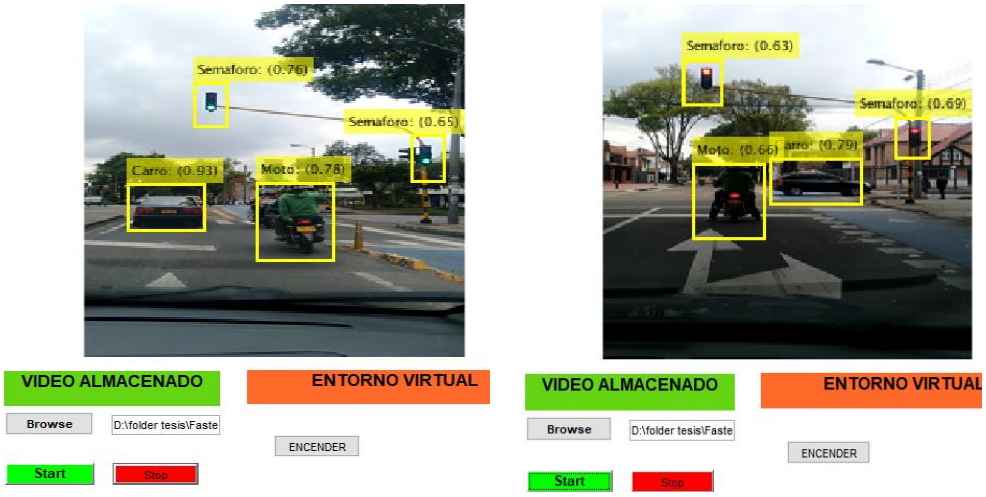



**Figura 66 Detección sin identificación de semáforo**

- **Entrenamiento de CNN Alex Net para reconocimiento de color de semáforos.**

El algoritmo de detección YOLO detecta 6 agentes, uno de ellos es semáforo, pero no se discrimina si es verde o rojo, por tal motivo se debe entrenar una CNN externa que trabaja pen paralelo, para corregir la clasificación de los semáforos. Para reducir la probabilidad de errores durante el entrenamiento, se anexan alrededor de 8000 imágenes entre semáforos verdes y rojos, utilizando la CNN de Alex Net redimensionando las imágenes a 227 x 227, estas dimensiones son debido a que es la única forma que la red Alex Net funcione, se establecen 5 épocas de entrenamiento, un LR de -0.00001, obteniendo un Accuracy del 99,34% al final del entrenamiento. Una vez entrenada esta red, se incluye dentro de la red YOLOv2, entrenada en la sección 5.4 como se ejemplificó en la **Figura 67**.

utilizando el 20% como validación como se ve en las imágenes de semáforos verdes y semáforos rojos de la matriz de confusión

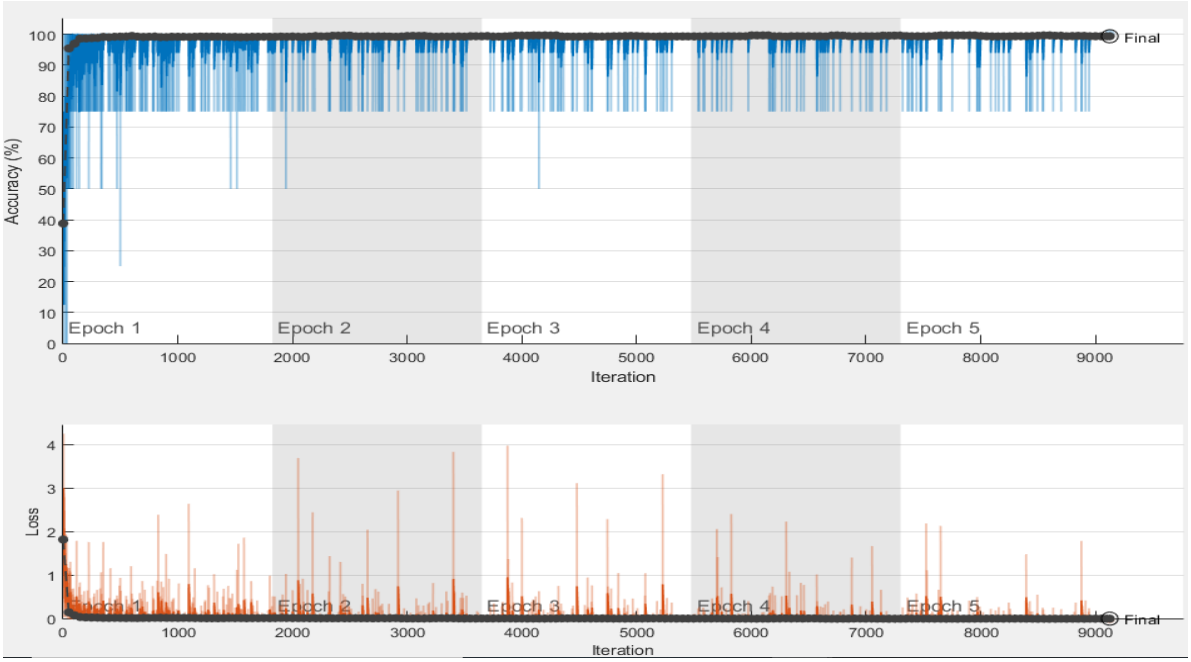



**Figura 67 Precisión y perdidas para CNN verde y rojo**

En la **Figura 67** se evidencia que antes de cumplir las primeras 1000 iteraciones en la época uno, se estabiliza la precisión, como también sucedió con las perdidas, estas tienden a cero. Se consideraron parámetros (ver **Tabla 17**) las 5 épocas para generar un aprendizaje con una tasa de error mínima, es por esto que se obtuvo una precisión mayor al 99% para identificar semáforos verdes o rojos. De esta CNN entrenada se obtiene una matriz de confusión, la cual evidencia el alto grado de precisión en verdaderos positivos como se evidencia en la **Figura 68**

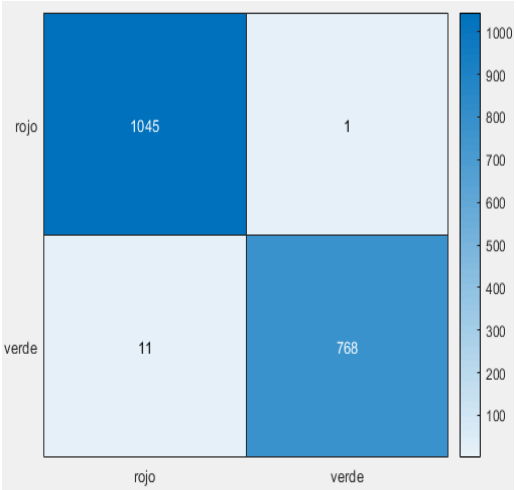



**Figura 68 Matriz de confusión rojo-verde**

**Tabla 17 Hiperparámetros de entrenamiento red rojo-verde**

| RESULTS | | | |
|---|---|---|---|
| Validation accuracy | | 99.34 | |
| Elapsed time | | 165min-5 sec | |
| **Training Time** | | **Validation** | |
| Epoch | 5 | frequency | 50 iteration |
| iteration | 9125 | hardware Resourse | GPU Nvidea 920Mx |
| Iteration per Epoch | 1825 | Learning Rate | 1E-05 |

En vista de verificar los buenos resultados en video, son expuestas desde un celular imágenes de semáforos, y se comprueba la categoría “rojo” con un 100%, verde con un 97% como se evidencia en la **Figura 69**. Una vez entrenada la red, esta se concatena con la arquitectura YOLOv2, previamente entrenada, de tal forma que la arquitectura de detección detecta “Semáforo”, y la CNN entrenada escribe el label que corresponde a rojo, o verde. En la **Figura 70** se evidencia el comportamiento de la red para clasificar los semáforos es decir, ya se puede apreciar el label “semáforo” y el color  al que pertenece (verde o rojo) detectándolo correctamente.

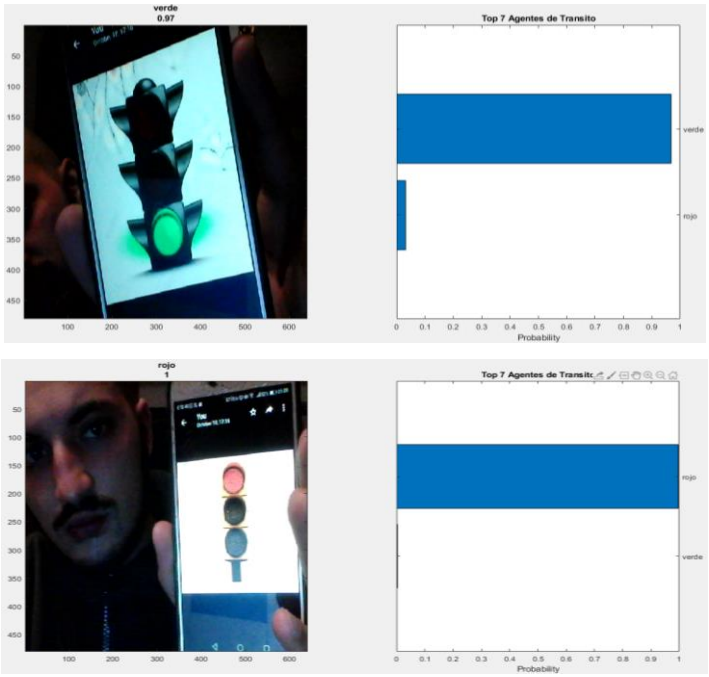



**Figura 69 Detección verde-rojo en video para semáforos con CNN**


**Figura 70 Clasificación y detección de semáforos por color**

# ● CAPITULO 6. ANÁLISIS Y RESULTADOS

## ○ 6.1 Comportamiento del móvil simulado en trayectoria longitudinal frente a obstáculos de interés.

En este capítulo se presentará el comportamiento del coche simulado en V-Rep a partir de la detección de la red YOLOv2 entrenada, y se evaluará con métricas de calidad con gráficos de precisión vs recall utilizando el data set #2 de entrenamiento y validación.

### ▪ 6.1.1 Toma de decisiones a partir de la cinemática.

Para este caso se toman varias escenas en las que se visualizara el comportamiento de la distancia obtenida frente al comportamiento de la velocidad angular del coche. La toma de decisiones se ejemplifica en la **Figura 71**, en donde se presenta un flujograma de actividad en el simulador al enfrentarse a el reconocimiento de los seis agentes entrenados.

Durante una vía en movimiento se asumirá que el auto que va adelante cambia su velocidad constantemente, para efectos del tráfico. Considerando este comportamiento se incluyó una función polinómica en el grafico ++ que muestra una aproximación a el comportamiento que deberá llevar el coche considerando la distancia como variable independiente.

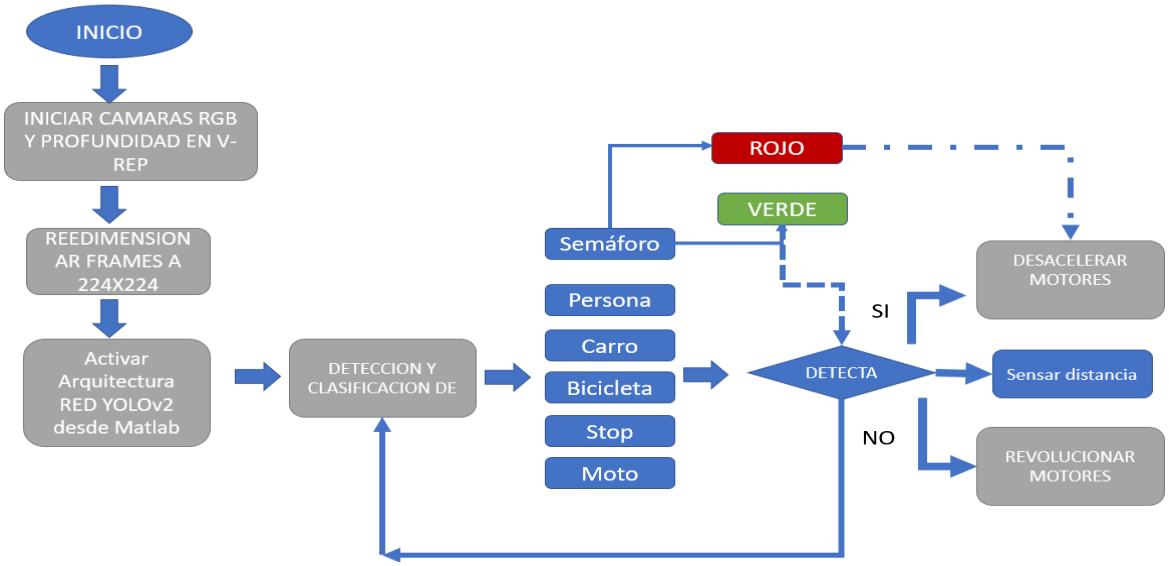



**Figura 71 Acople del simulador con la arquitectura YOLO- Fuente propia**

El sensor al usar es una cámara de profundidad RGBD, la cual tiene una adquisición de datos máxima de distancia a 8 metros. Dado que la cámara RGB-D se ubica a 1,5 metros del frente del capo (zona de contacto), significa que realmente se obtiene la distancia máxima de los

**Tabla 18 Comportamiento Distancia VS Velocidad**

| Distancia Metros (d) | 6,5 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|
| Velocidad angular rad/s | 36,58 | 29,87 | 24,39 | 17,24 | 0 | 0 | 0 |
| (VL)Velocidad Lineal m/s | 6,97 | 5,69 | 4,65 | 3,28 | 0 | 0 | 0 |
| (VL)Velocidad en km/h | 25,09 | 20,48 | 16,72 | 11,83 | 0 | 0 | 0 |

objetos de interés, a partir de 6,5 metros (8m-1.5m). El auto, aunque se encuentre de colisionar del objeto de interés a 6,5 metros, se toma como criterio de parada a dos (2) metros de distancia, es decir el automóvil desde que detecta el objeto de interés hasta que se desacelera totalmente, puede desplazarse un máximo de 4,5 metro, como se evidencia en la (**Tabla 18),** yendo de 6,5 m a 2m. De acuerdo a estas condiciones iniciales y la cinemática calculada para el movimiento uniformemente acelerado con un coeficiente de fricción hallado, se toma el **radio el cual es de 0.1905 m** de la **Ecuación 21**, para hacer los cálculos de la velocidad angular y poder visualizar el movimiento de las ruedas.:

Ejemplo velocidad angular W= VL/r donde el radio (r) será = 0.1905 m

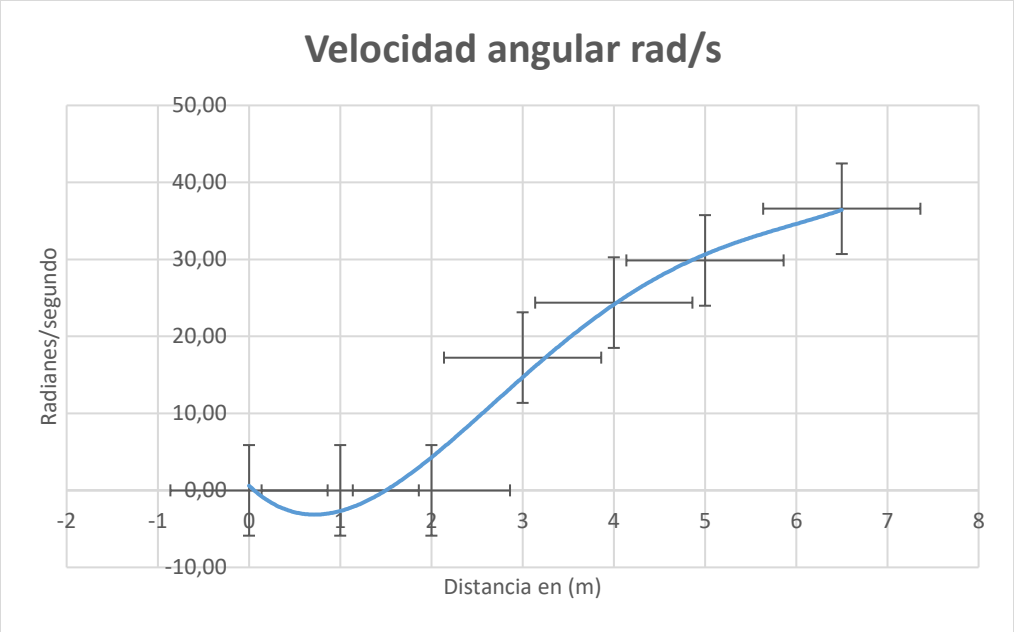



**Figura 72 Función polinómica de velocidad frente a la distancia**

$$w = 0{,}1002x^4 - 1{,}7349x^3 + 9{,}6282x^2 - 11{,}287x + 0{,}6116 \qquad (21)$$

$$R^2 = 0{,}9772$$

En todos los casos a continuación se podrá visualizar dos partes, una lo que ve el sujeto y otra lo que ve la cámara RGBD. Se tomarán dos momentos para evidenciar el comportamiento proporcional de la distancia del objeto de interés frente a la velocidad angular, acelerando al encontrarse más lejos o desacelerando al estar más cerca. Como se presenta en la **Ecuación** 21, es una función polinomial de cuarto orden, que se adapta en un 97.72% al movimiento del coche en velocidad – distancia ejemplificado en la **Figura 72**.

**Caso 1**
**El auto enfrente de una señal de pare**

En este caso el auto detecta la **señal de pare** a 4.82 metros, y su velocidad angular es de 28.97 rad/s. Se observa el Kinect_depth de color azul representando que se encuentra lejos de los objetos de interés. **Ver Figura 73**

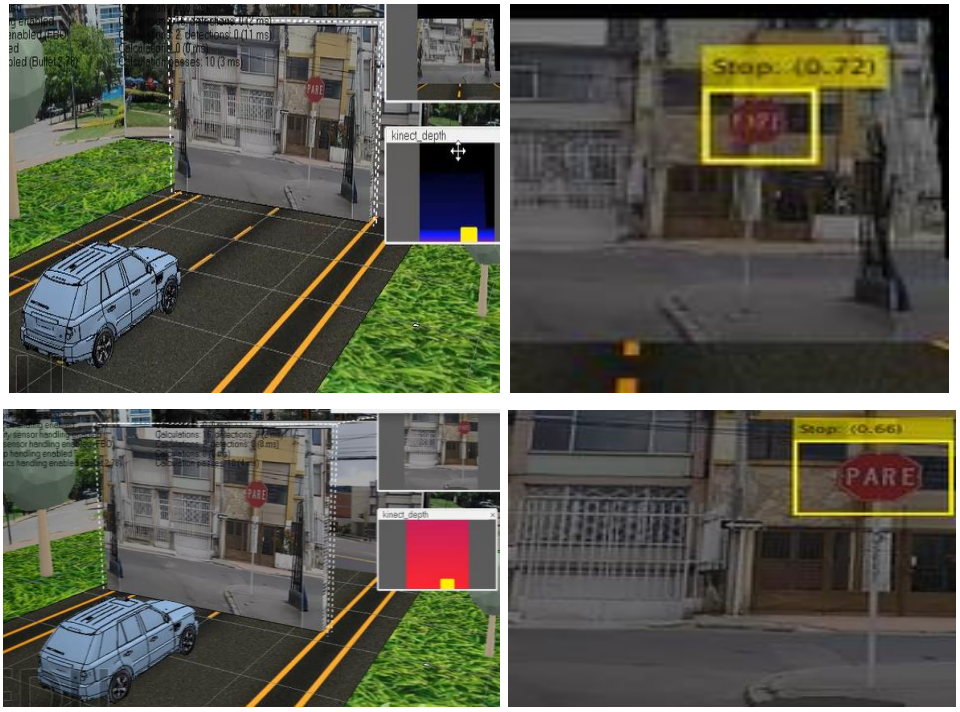



**Figura 73 Caso 1 Visualización velocidad-distancia a partir de la detección pares**

El coche una vez empieza a acercarse empieza a desacelerarse hasta obtener una distancia máxima de 2 metros, al igual el Kinect_depth de profundidad visualiza el color rojo es decir está más cerca. En este punto la velocidad angular debe ser cero rad/s.

**Tabla 19 Visualización velocidad-distancia a partir de la detección pares**

| Objeto detectado | | momento 1 | momento 2 |
|---|---|---|---|
| **PARE** | distancia (m) | 4.82 | 1,9 |
| | velocidad (rad/s) | 28.97 | 0 |

**Caso 2**

**El auto frente a un semáforo en luz verde**

El auto detecta la luz verde, ver **Figura 74**, por consiguiente, aunque conoce la distancia y la ubicación del objeto de interés, en este caso valores en distancia por 5,36 m y en el segundo caso 2, 33 metros del semáforo. El coche no se detiene jamás y permanece con la velocidad lineal máxima establecida en las condiciones iniciales explicadas en el capítulo 4, la cual es 6,97 m/s o en velocidad angular 36,58 rad/s.


**Figura 74 Visualización velocidad-distancia a partir de la detección Semáforo verde**

**Tabla 20 Visualización velocidad-distancia a partir de la detección Semáforo verde**

| **Objeto detectado** | | **momento 1** | **momento 2** |
|---|---|---|---|
| **SEMAFORO VERDE** | distancia (m) | 2.33 | 5.36 |
| | velocidad (rad/s) | 38.58 | 36.58 |

**Caso 3**
**El semáforo está en rojo**

El automóvil al encontrar que el semáforo posee la luz roja, (ver **Figura 75**) la cual significa detenerse, va desacelerando el movimiento de las ruedas. En la imagen se visualiza a una distancia de 3.97metros y su velocidad angular se encuentra a 24,21 rad/s.

Al acercarse y llegar a la distancia igual o menor a 2 metros, debe haberse detenido totalmente. En este caso al obtener la distancia es de 1.83 metros, y la velocidad angular es cero.

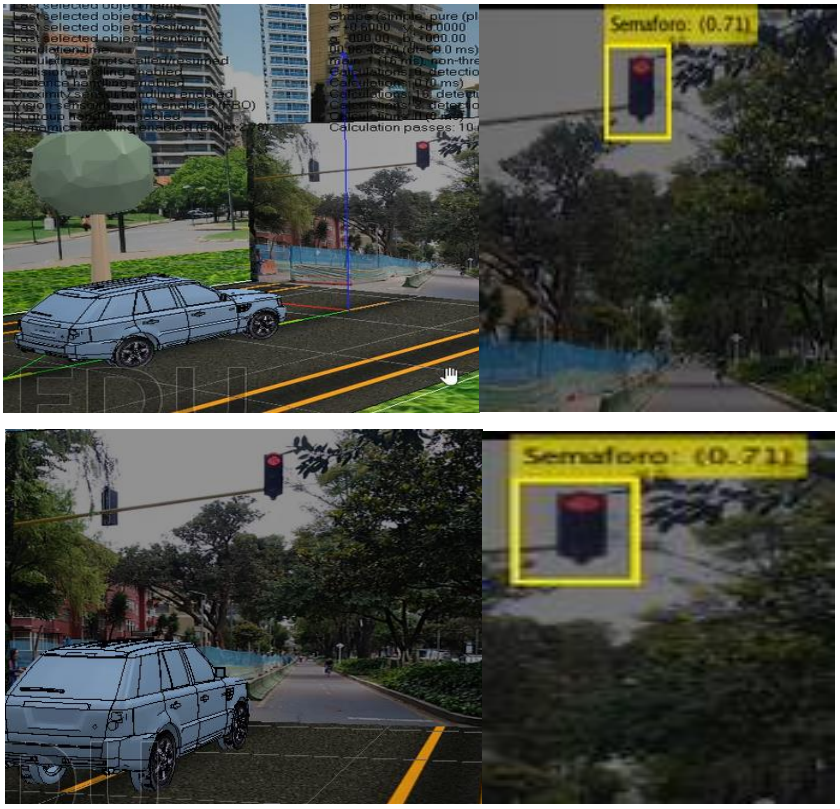



**Figura 75 velocidad-distancia a partir de la detección de motos, autos y semáforo en verde**

**Tabla 21 velocidad-distancia a partir de la detección de motos, autos y semáforo en verde**

| Objeto detectado | | momento 1 | momento 2 |
|---|---|---|---|
| **SEMAFORO ROJO** | velocidad (rad/s) | 24.21 | 0 |
| | distancia (m) | 3,57 | 1,987 |

### Caso 4
### Estar en movimiento detrás de un coche o una moto.

Posible situación en trancón (**Figura 76**). Como se estipula en los antecedentes, Bogotá pertenece a la 3ra ciudad que más tiempo demora en atascos seguido de Moscú y Estambul para lo que es útil el uso del Traffic Jam Assist (TJA), es decir el auto es parcialmente

autónomo en atascos ya que la velocidad es mínima. Para este caso se considera que el coche debe variar su velocidad de acuerdo a la distancia a la que se encuentre con el objeto de interés al frente, es decir, la velocidad angular aumentara si los objetos de interés se alejan. En este cuarto caso, el automóvil observa un semáforo en verde, un coche y una moto. Se toman cuatro datos, de cuatro momentos donde varia la distancia y velocidad angular como se evidencia en la **Tabla 22**


**Figura 76 velocidad-distancia a partir de la detección de motos, autos y semáforo en verde**

**Tabla 22 Evaluación de distancias y velocidades angulares, con respecto a 3 objetos.**

| **Objeto detectado** | | **momento 1** | **momento 2** | **momento 3** | **momento 4** |
|---|---|---|---|---|---|
| **SEMAFORO VERDE** | distancia (m) | 3,55 | 5,55 | 2,79 | 2 |
| **CARRO** | distancia (m) | 3,22 | 5,42 | -- | 1,987 |
| **MOTO** | distancia (m) | 3,20 | 5,45 | 2,50 | 1,987 |
| | **velocidad (rad/s)** | 18.89 | 31.9 | 12.19 | 0 |

Se evidencia que el **semáforo está en verde**, es decir el **coche puede continuar avanzando**, sin embargo, se encuentra una moto y un carro con los que podría colisionar, debido a esto se debe detener y movilizarse con respecto al movimiento que el objeto más cercano y justo al frente detecta. En el caso uno la distancia es 3,20 metros y la velocidad angular es de 18,89 rad/s, en el caso dos se aparta aumentando la distancia a 5,42 m y la velocidad angular a 31,9 rad/s. A raíz de que el/los objetos de interés aumentan su distancia la velocidad también. Posteriormente en el momento 3, disminuye la distancia a 2,5 metros

y la velocidad se reduce a 12.19 rad/s. Finalmente se obtiene una distancia de 1,90 metros a lo que el auto responde a una velocidad angular de 0 rad/s, es decir se detiene.

**Caso 5**
**Semáforo en verde con persona**

El coche debe avanzar (**Figura 77**) ya que el semáforo se encuentra con la luz en verde, pero encuentra una persona aparentemente imprudente, cruzando la cebra. Para este momento, se encuentra el semáforo a 4.76 metros y la persona a 3.75 m del sensor, a lo que la velocidad del coche responde a una velocidad angular de 22,82 rad/s

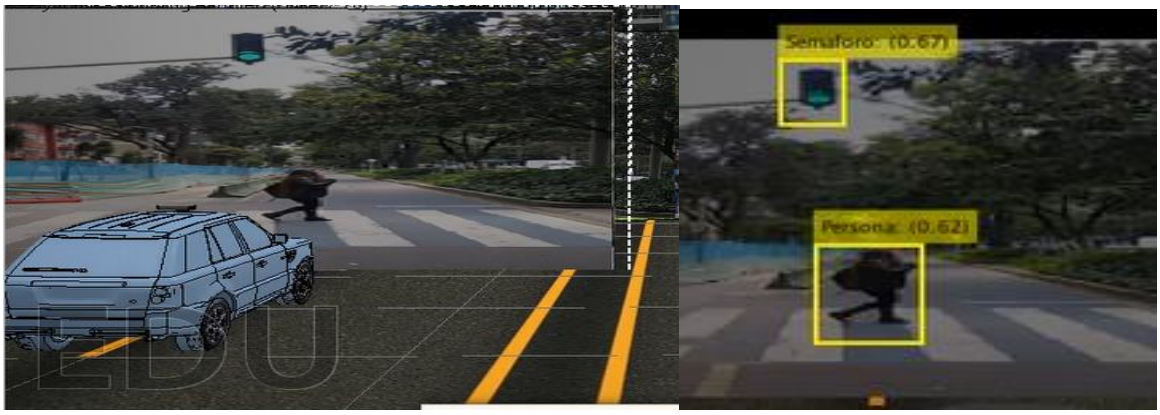



**Figura 77 velocidad-distancia a partir de la detección una persona y un semáforo en verde**

El semáforo continuó en verde y el coche al estar a una distancia, en este caso de una distancia menor a 1,90 metros al objeto de interés (persona), la velocidad angular es cero hasta que la persona u objeto detectado ya no se encuentre obstaculizando el movimiento longitudinal (**Figura 78**).


**Figura 78 Velocidad 0 a partir de la detección una persona y un semáforo en verde a una distancia menor de dos metros.**

**Tabla 23 velocidad a partir de la detección de una persona y un semáforo en verde a dos distancias**

| Objeto detectado | | momento 1 | momento 2 |
|---|---|---|---|
| SEMAFORO VERDE | distancia (m) | 4,14 | 2,3 |
| PERSONA | distancia (m) | 3,75 | 1,903 |
| | **velocidad (rad/s)** | 22,82 | 0 |

En la **Figura 79** se presenta la visualización de la simulación en el software V-Rep (izquierda) y aparecen los datos en distancia de los dos objetos de interés detectados (el semáforo y la persona), con la velocidad angular de las ruedas.

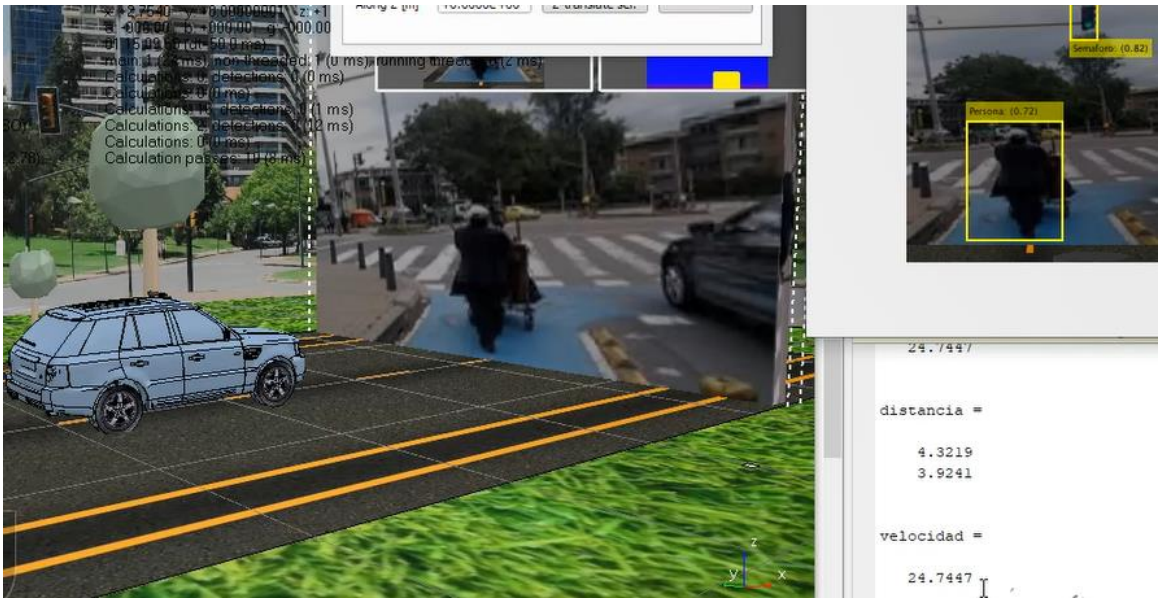



**Figura 79 Visualización a tiempo real distancia – velocidad**

En conclusión, como se ejemplifica en la **Figura 80**, el comportamiento en velocidad de las ruedas, variará según la distancia a la que sean percibidos los objetos de interés. La cámara de profundidad redimensiona la pixelación de entrada de las imágenes para poder procesarla con la arquitectura YOLOv2 y la ConvNets Resne50. La velocidad angular varia según la cinemática calculada como variable independiente la distancia.

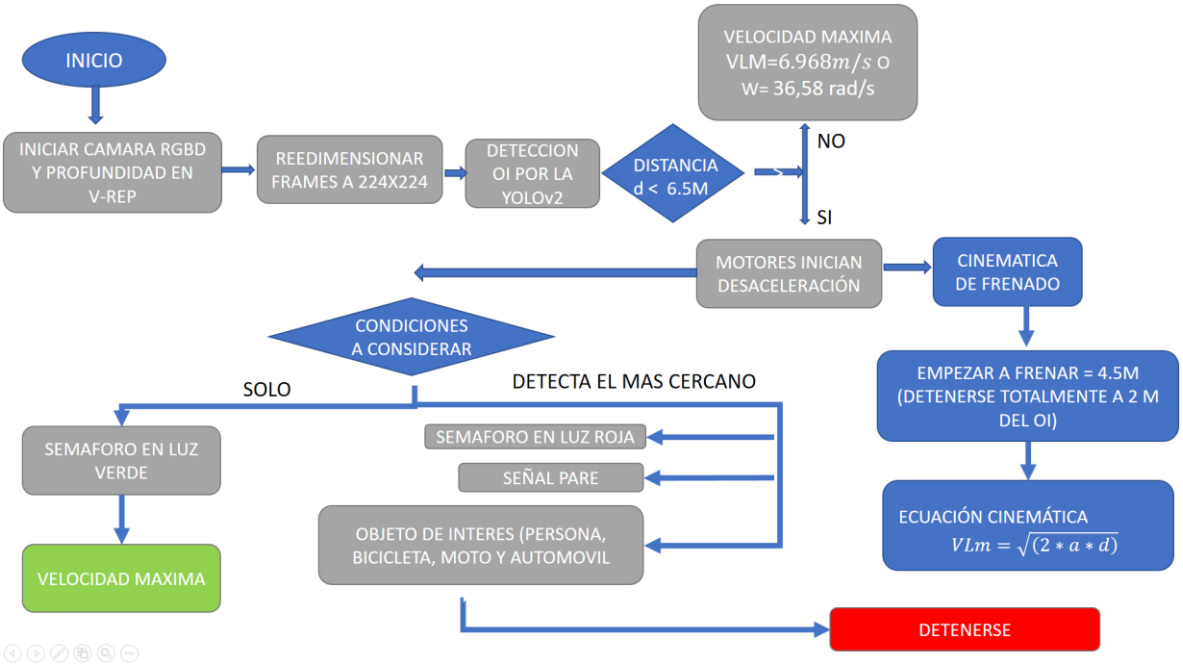



**Figura 80 Comportamiento del simulador frente la detección de los objetos de interés.**

## ○ 6.2 IMPLEMENTACIÓN DE MÉTRICAS DE CALIDAD

En la implementación de métricas de calidad fueron seleccionadas en función a las 6 clases seleccionadas, persona, carro, pare, semáforo, moto y bicicleta. La **Tabla 24** muestra la matriz de contingencia, la cual ilustra el significado de VP (verdadero positivo), FP (falso positivo), FN (falso negativo) y VN (verdadero negativo). El VP representa el número de clases detectados correctamente, los FP los objetos detectados que no corresponden con las etiquetas realizadas manualmente [138], el FN es la cantidad de objetos para cada clase que la red neuronal no pudo detectar y VN ocurre cuando en un cuadro no aparece un objeto etiquetado, y la red reporta correctamente tal situación. Por dicha razón, en este contexto se evalúa la detección calidad en términos de disminuir FP y FN [138].

La primera herramienta utilizada en el reconocimiento se conoce como *precisión* (o valor predictivo positivo), definido como VP / (VP + FP) capaz de medir la confiabilidad y

veracidad de la detección en proporción a los VP detectados [138]. Considerando VP + FP como la cantidad de objetos detectados en cada clase, la precisión indica porcentualmente las detecciones correctas. La segunda métrica es el recall o también llamada *sensibilidad,* definida como VP / (VP + FN), la cual básicamente determina en cantidad porcentual el número de objetos encontrados correctamente [138].

**Tabla 24 Situación real predicha por la red en calidad**

| | | Predicho por la red | |
|---|---|---|---|
| | | Objeto detectado | Objeto no detectado |
| Situación real | Objeto actual | Verdadero positivo (VP) | Falso Negativo (FN) |
| | Objeto no actual | Falso Positivo (FP) | Verdadero Negativo (VN) |

A continuación, se realiza la presentación en la **Figura 81**, que ilustra la precisión vs recall (sensibilidad), para el data set de datos entrenado el cual reúne alrededor de 2421 imágenes extraídas de 20 videos y 605 imágenes para el test extraídas de 6 videos.

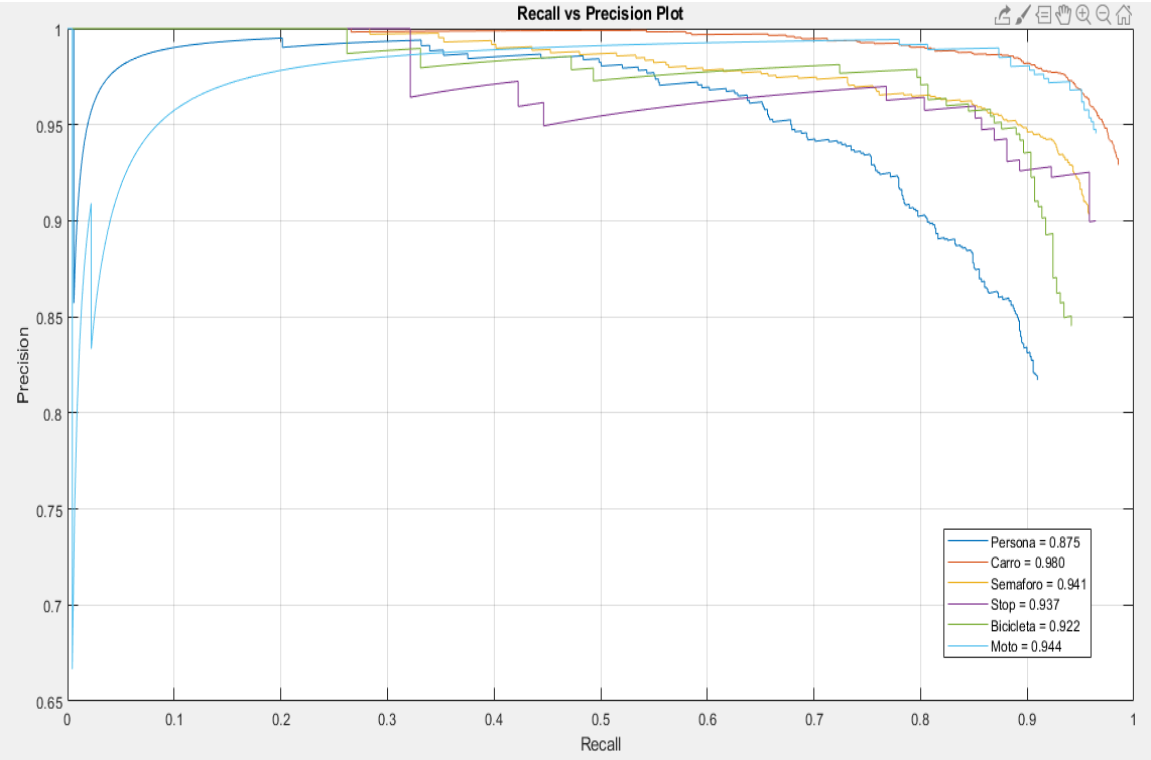



**Figura 81 Precisión - Recall Data set de entrenamiento**

**Tabla 25 Precisión - Recall del Data set de entrenamiento**

| En el reconocimiento y detección de los objetos de interés del **DATA SET DE ENTRENAMIENTO**, con 2421 imágenes, se obtienen los siguientes resultados en el grafico precisión – Recall para las 6 categorías siguientes: | |
|---|---|
| CARRO | 98,00% |
| MOTO | 94,40% |
| SEMAFORO | 94,10% |
| STOP | 93,70% |
| BICICLETA | 92,20% |
| PERSONA | 87,50% |

El resultado de precisión – Recall con la base de datos de entrenamiento tuvo un promedio general de 93.3%, visto en la **Figura 82**. Una vez efectuado el entrenamiento de las redes y visto u comportamiento con la data set de entrenamiento, se procede a

realizar su validación con un nuevo data set para comprobar el desempeño de cada una de las clases. En esta etapa de prueba se evidencia una respuesta de precisión-Recall del 60% en general, donde no se incluyen las bicicletas, ya que el reconocimiento de este agente tuvo varios problemas de reconocimiento, al confundirse con personas y motos.

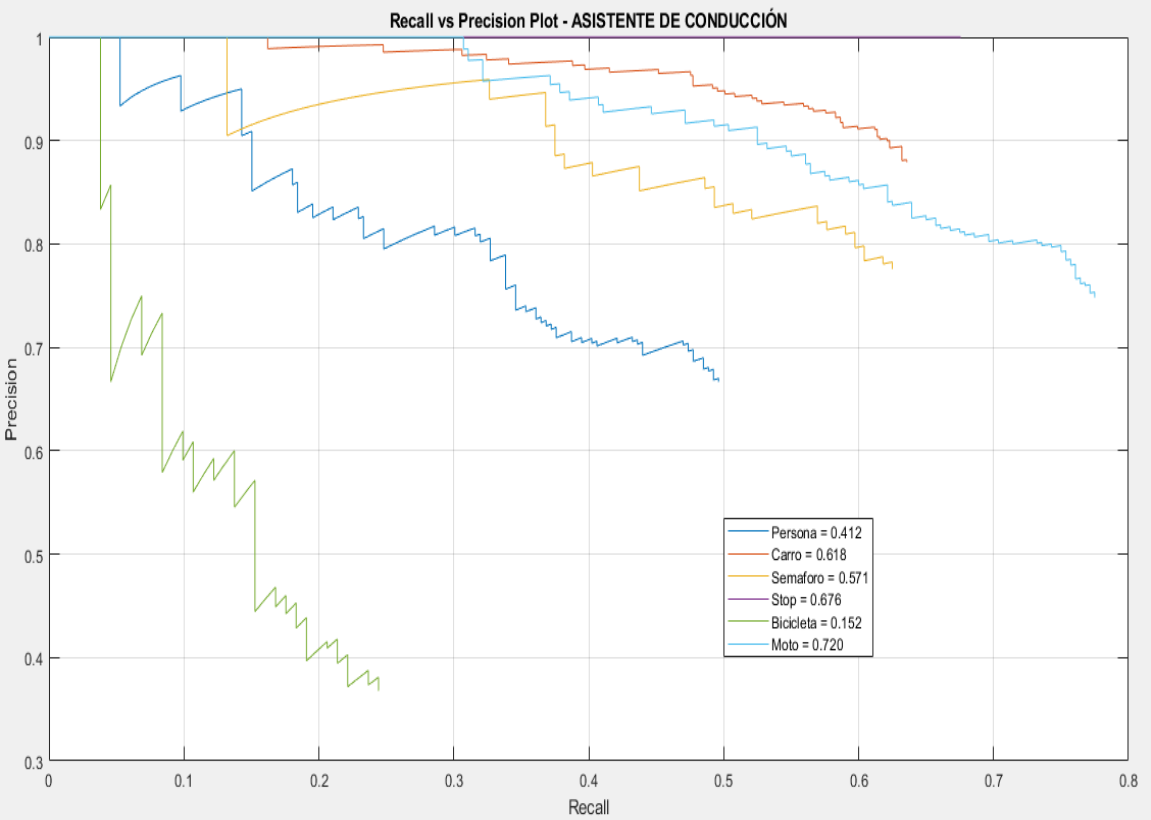



**Figura 82 Precisión - Recall Data set de prueba**

**Tabla 26 Precisión - Recall Data set de prueba**

| En el reconocimiento y detección de los objetos de interés del DATA SET DE PRUEBA (25%), 605 imágenes, se obtienen los siguientes resultados en el grafico precisión – Recall para las 6 categorías siguientes: | |
|---|---|
| CARRO | 62% |
| MOTO | 72% |
| SEMAFORO | 57% |
| STOP | 68% |
| BICICLETA | 15% |
| PERSONA | 41% |

**Un clasificador** con alta precisión será asertivo en la detección, pero su sensibilidad será baja siendo inversamente proporcionales [**15]**, La curva precisión-recall muestra la compensación entre asertividad y sensibilidad a diferentes umbrales de detección. La alta precisión enfatiza una baja tasa de falsos positivos detectados, la sensibilidad es alta si aumenta la tasa de falsos positivos detectados.

Los puntajes obtenidos del grafico muestran si el clasificador devuelve resultados precisos (alta precisión), y si cuenta con resultados positivos (alta sensibilidad) [**138]**, El área debajo de la curva de precisión-recall (AP) será entonces un indicador de calidad del detector, ya que tendrá un valor mayor si es alta la sensibilidad y la precisión de acuerdo a cada clase, y la medición promedio de esta detección bautizada como mAP [138].

## ▪ 6.4 imágenes etiquetadas manualmente frente a las detectadas por la YOLOv2.

Todas las imágenes que se presentan a la **izquierda representan las Figuras etiquetadas** manualmente, y **a la derecha, las que reconoce la red YOLO v2.**

Para el reconocimiento y comparación entre la detección y el etiquetado manual, se encerrará en un recuadro de diferente color según la clase como se presenta a continuación:

- Red YOLO: Amarillo
- Bicicleta: Azul
- Moto: Blanco.
- Carro: Rojo
- Persona: Cian
- Semáforo: Magenta
- Stop: Verde

**Verdaderos Positivos**

En la **Figura 83** se puede apreciar un buen reconocimiento de semáforos, coches, personas, señal de pares, motos y bicicletas utilizando imágenes de una base de datos con 600 imágenes, la cual es diferente a la utilizada para el entrenamiento, lo que indica una buena detección. Por otra parte, en la **Figura 84** detecta un carro en una posición no habitual, ya que éste se visualiza de lado. En la imagen de abajo la otra imagen es capaz de incluso detectar el bus de color rojo, el cual no fue etiquetado manualmente, sin embargo, la red logra clasificarlo correctamente

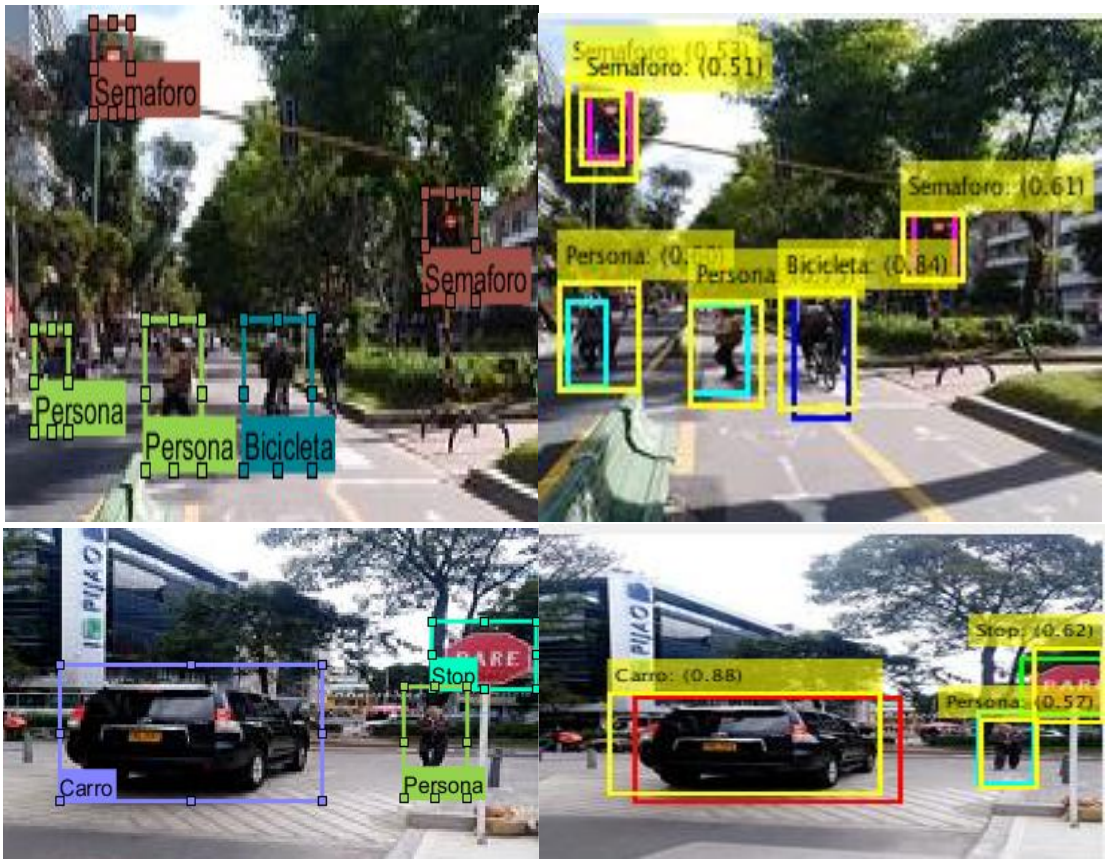



**Figura 83 Ejemplo de (VP) verdadero positivo.**

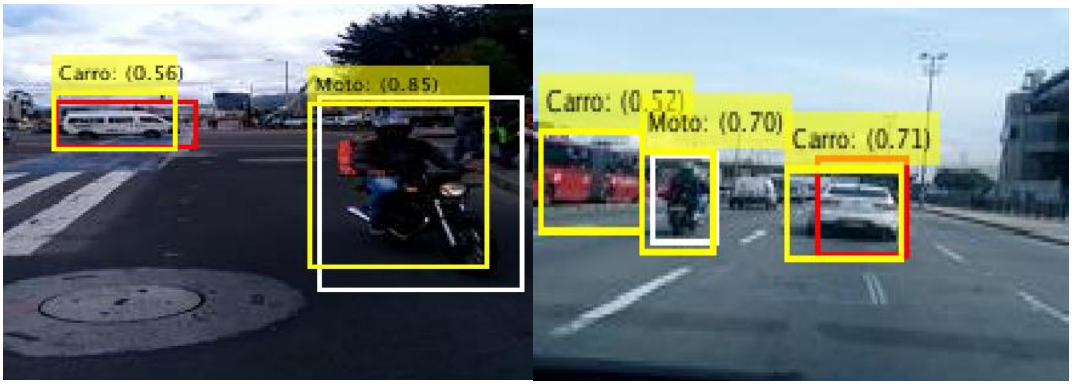



**Figura 84 Ejemplo de VP no habituales**

### Falsos Positivos

En la **Figura 85** se puede apreciar dos imágenes diferentes a las utilizadas para el entrenamiento, que el automóvil lejano es detectado como una moto, al igual que algunas personas se confunden por bicicletas en la segunda imagen. Esto se puedo dar en el primer caso debido a que la imagen utilizada fue en un escenario con lluvia y nubloso lo que pudo generar confusión por la red. En el segundo caso el objeto de interés “bicicleta”, se etiqueto cuando una persona está andando en ella, sin diferenciar que hay una persona encima de la bicicleta, por esto pudo generar error el entrenamiento al clasificar personas o bicicletas.

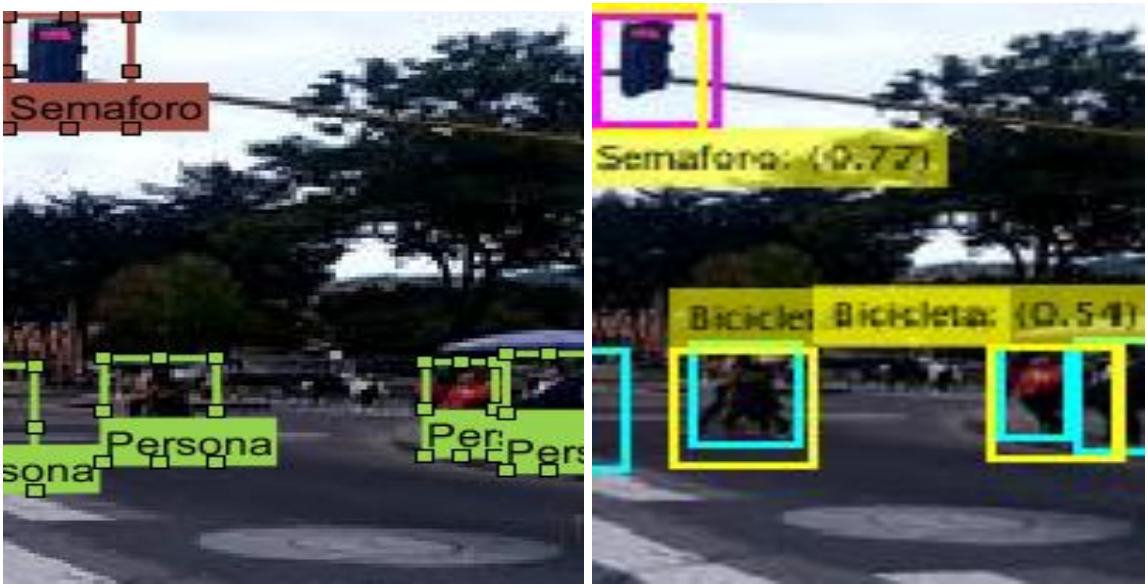



**Figura 85 Ejemplo (FP) Falsos Positivos**

### Falsos Negativos

En la **Figura 86** se puede apreciar dos imágenes diferentes a las utilizadas para el entrenamiento, en la izquierda las etiquetas esperadas y en la derecha lo que reconoce la red de color amarillo. Se evidencia donde la persona muy cerca no es capaz ser detectada por la red, en el segundo caso el automóvil de la derecha. La posible explicación es debido a que la data set de etiquetas con categoría personas se realizó desde un coche a una distancia mayor de dos metros, lo cual complica la detección de rasgos al estar tan cerca. Dado que la trayectoria de las personas es de naturaleza estocástica [61], las personas fueron capturadas en diferentes tamaños en los frames del image labeler. Si las personas están cerca de la cámara, se capturan en un tamaño de píxel mayor y tienen características de imagen más ricas. Esto se puede solucionar aumentando la base de datos del entrenamiento. En el caso del coche no se detecta, sin embargo, es posible que aumentando el data set y mejorando el brillo y resolución de etiquetas, se detecte correctamente.


**Figura 86 Ejemplo (FN) Falsos Negativos**

### Ejemplo de etiquetado múltiple

En la **Figura 87** se puede apreciar una imagen diferente a la utilizada para el entrenamiento, donde aparece, aunque bien detectado y clasificado el ROI de una moto, se duplican los recuadros que lo detectan. Este fenómeno sucede en menor proporción frente a otras redes y se soluciona mejorando los parámetros de los Anchors y bounding boxes, mejorando las etiquetas y diferenciando más efectivamente el ancho y el alto de las categorías utilizadas.

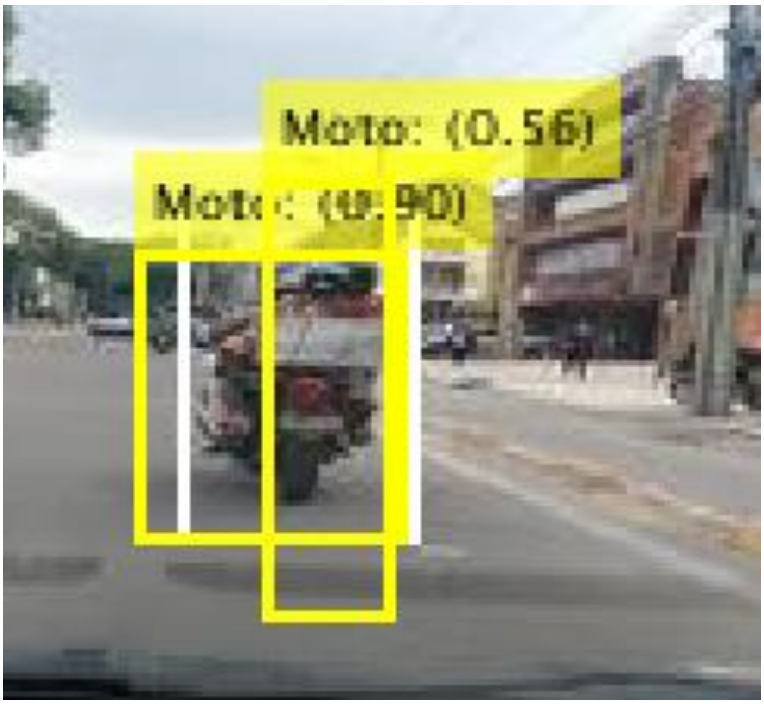



**Figura 87 Ejemplo de etiquetado Múltiple**

### Ejemplo detección de objetos inexistentes

En esta sección se evidencia que el algoritmo confunde en dos ocasiones las sombras de un árbol con personas, como se observa en la **Figura 88**, esto es debido a las etiquetas realizadas a larga distancia y baja resolución en la categoría “personas”.


**Figura 88 ejemplo de detección en objetos inexistentes**

**Detección de objeto lejano**

En la **Figura 89** se pueden apreciar dos imágenes diferentes a las utilizadas para el entrenamiento, donde no se alcanza a identificar algunos semáforos y motos que si fueron etiquetados. Esto sucede debido al ser la distancia lejana y la red Yolo al no obtener un enmallado más fino para detectar pequeños objetos. Los objetos que se encuentran lejos de la cámara se capturan en un tamaño de píxel más pequeño y tienen características de imagen de menor calidad. Esto puede mejorarse utilizando una cámara de mejor resolución y aumentando la base de datos de entrenamiento.

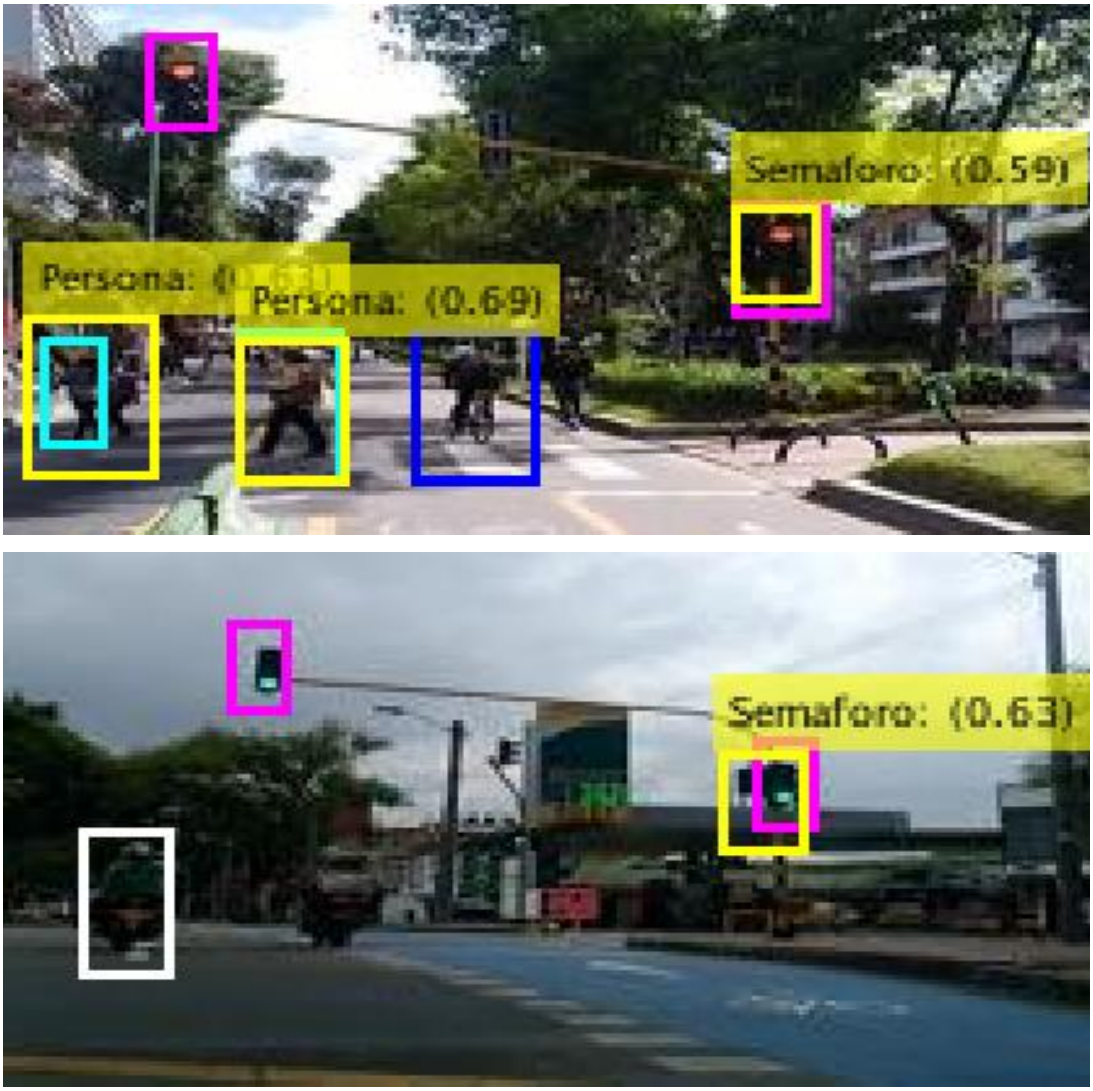



**Figura 89 Detección de objeto lejano.**

- **Comparativa gráficos precisión – Recall de la base de datos entrenada frente a la base de datos de test.**

**Tabla 27 comparación Precisión - Recall**

| **El porcentaje que disminuyó la red en detectar las seis categorías de la data set entrenado con 2450 imágenes, frente al data set de prueba con 605 imágenes es:** | | | |
|---|---|---|---|
| **Categoría** | Resultados Precisión-Recall **Entrenado** | Resultados Precisión-Recall **test** | Diferencia = (**test - entrenado) /entrenado** |
| **CARRO** | 98% | 62% | 36,7% |
| **MOTO** | 94% | 72% | 23,7% |
| **SEMAFORO** | 94% | 57% | 39,4% |
| **STOP** | 94% | 68% | 27,4% |
| **BICICLETA** | 92% | 15% | 83,7% |
| **PERSONA** | 88% | 41% | 53,1% |

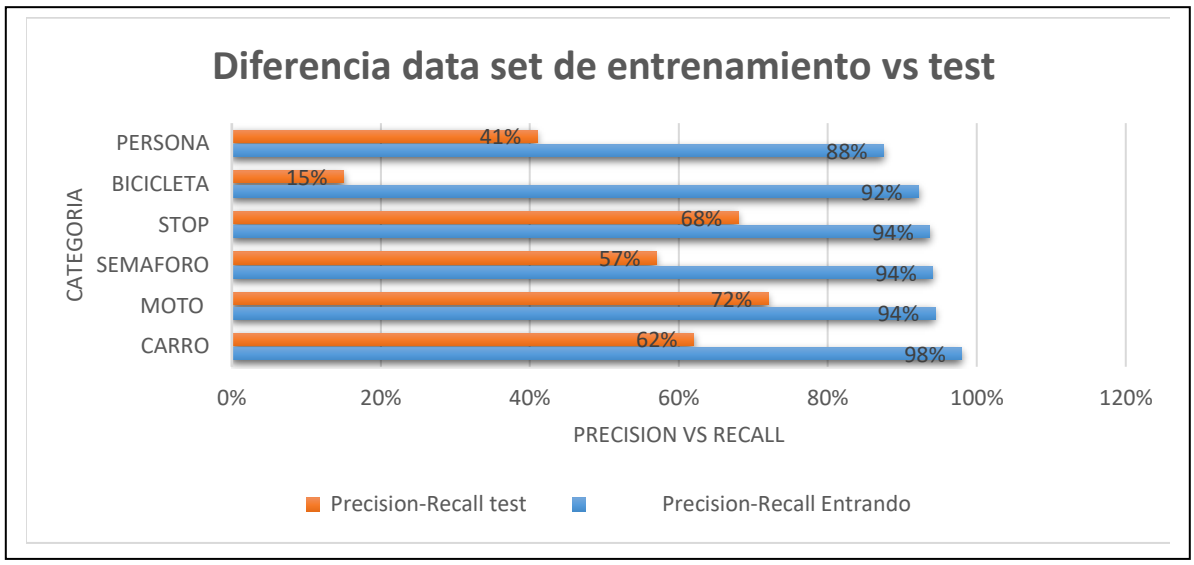



**Figura 90 de barras comparación Precisión – Recall**

.

En la **Tabla 28**, se aprecia la diferencia en resultados obtenidos por los gráficos de precisión Recall de la **base de datos del entrenamiento frente a la base de datos de prueba** o **validación**, donde se presentan mejores resultados en las categorías carro, moto, stop y semáforo obteniendo resultados mayores al 62%. Para la categoría bicicleta, semáforo y persona, no obtuvo tan buenos resultados al obtener resultados inferiores al 57%. Esto puede solucionarse aumentando la base de datos y mejorando el etiquetado, o probando otra CNN. En la **Figura 90**, se expone la diferencia en barras, del entrenamiento entre la validación de precisión y recall mostró que el reconocimiento en carros fue del 98% en el entrenamiento y de 62% con la base de datos de validación, obteniendo una diferencia del 36,7%, stop 94% frente a 68% en validación, persona 88% frente a 41%, moto 94% frente al 72% por nombrar unos casos.

## ▪ 6.5 Implementación de velocidad, utilizando TIC TOC para la medición del tiempo de procesamiento

Se implementa la función TIC TOC en el procesamiento y se evidencia que, para reconocer 3 categorías, en este caso dos (2) carros y una (1) señal de Stop, toma 129 milisegundos, lo que significan 8 FPS (frames per seconds) procesos por segundo.

En este caso se pone a prueba el procesamiento de la YOLO y además la ALEXNET encargada de diferenciar el color del semáforo. Como se observa en la **Figura 91** el tiempo en procesar 126 milisegundos, en los que se identifican tres (3) agentes de interés, un carro con un 87% de Accuracy, una moto con un 81% y un semáforo con un 64% respectivamente. Posteriormente entre a categorizar la red ALEXNET entrenada para diferenciar el color del semáforo como se evidencia en y se categoriza como “verde”, lo cual es correcto. Se puede concluir que se procesa en este caso 8 FPS, sola no es capaz de identificar si el semáforo es “rojo” o “verde”.

La velocidad de la YOLO entrenada se refiere al tiempo consumido por dicha red neuronal al realizar la detección de todas las clases entrenadas. Para el cálculo de la velocidad vale comentar 3 datos importantes. El hardware utilizado, las dimensiones de las imágenes y el peso de la arquitectura de la red. El procesador utilizado es un Intel i5 7200U @ 2.50GHz (4 CPU) se desarrolla en un computador ASUS referencia X441U utilizando una tarjeta de

video NVidia Gforce GT920MX 2GBDDR3 y memoria RAM de 8GB. La velocidad es una métrica importante ya que uno de los objetivos del presente trabajo es minimizar los accidentes a través de un buen uso de un asistente de conducción. La red neuronal empleada, fue creada para procesar más de 40FPS, en nuestro caso debido a la tarjeta gráfica de poca capacidad, se alcanzan 8 FPS, sin embargo, es de buena respuesta para tiempo real ya que toma alrededor de 124 milisegundos clasificando y detectando las imágenes entrenadas

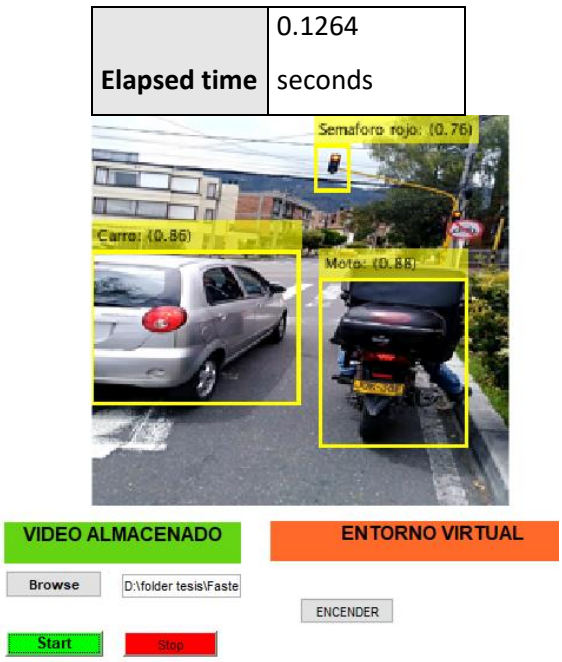



**Figura 91 Tiempo de procesamiento YOLO**

**Tabla 28 Tiempo de procesamiento YOLO vs Faster Rcnn**

| **Arquitectura** | **Elapsed time Seconds** |
|---|---|
| YOLOv2 | 0,1265 |
| Faster RCNN | 1,98 |
| Diferencia de la Yolo en veces más rápido | **15,7** |

En velocidad la Faster RCNN necesitó 1,98 segundo de procesamiento por frame, frente a la YOLOv2, la cual necesito de 0.1264 segundos por frame, haciéndola 15 veces más rápida ver. La YOLO es una opción óptima ya que procesa una sola vez la imagen, obteniendo los cuadros delimitadores, así como las clasificaciones de categorías de objetos, realizando un único procedimiento en paralelo, siendo esta la razón por la que se escoge YOLOv2 como detector.

# CAPITULO 7. CONCLUSIÓN Y TRABAJOS FUTUROS

- Los motivos obtenidos del observatorio nacional de Seguridad Vial y el RUNT, los siniestros viales que pudieron ser prevenidos, trae a consideración para este trabajo, la utilización de herramientas tecnológicas que presten a los conductores una asistencia a la conducción. Los siniestros viales, son causalidades producto de imprudencias, entre ellos están principalmente: incumplir las normas de tránsito, no mantener distancia, el exceso de velocidad, el tráfico denso y no ser visible en la vía. Conociendo estos efectos negativos, se identifica la necesidad de la creación de un software que funcione como asistente a la conducción, capaz de observar eventos en las calles, clasificando y detectando la distancia que separa al automóvil de un objeto de interés o de señal de tránsito, sirviendo de apoyo visual para que el conductor se abstenga de cometer las faltas de tránsito.
- La estructura de la YOLO es una arquitectura capaz de detectar y clasificar varios OI en un ROI con una sola CNN lo que permite aplicabilidad para el presente trabajo. Esta arquitectura se escoge por demostrar un buen rendimiento en velocidad y precisión frente a arquitecturas basada en regiones la Faster RCNN.

- Las CNN como algoritmos de DL, demostraron poder actualizar los filtros a través de técnicas de convoluciones permitiendo actualizar los pesos automáticamente como lo haría una red neuronal multicapa, además de presentar un mejor rendimiento que las técnicas clásicas de ML. La arquitectura YOLO, procesa las imágenes de entrada a través de una grilla que genera cuadros delimitadores con bounding boxes, identificando según el mapa de probabilidad la clase. La CNN (en este caso Resnet50), genera un score según el label o etiqueta de la imagen clasificando y detectando los OI seleccionados, la cual tiene buen rendimiento, por su arquitectura de capas residuales y filtros de 3x3 que permite un procesamiento más eficiente.

- Varios avances en CA han permitido la reducción de siniestros viales, mayor comodidad, seguridad y descongestión en carreteras, sin embargo, los costos asociados a estas tecnologías suelen ser altos, a lo que el crear un prototipo para cualquier carro en países con un nivel de poder adquisitivo menor como Colombia,

podría aportar a mejorar la movilidad y reducir los accidentes viales.

- La SAE cuenta con (6) niveles de autonomía. En este trabajo se consideró el desarrollo del nivel dos, debido a las características que lo relacionan, entre ellas, dar asistencia a la conducción, mantener un movimiento longitudinal, proponer un sistema con una cámara estéreo, capaz de reconocer la distancia de los OI que se pueda implementar para un automóvil convencional en Colombia.
- En el estado del arte, se presentan investigaciones que aportan a la conducción autónoma, entre ellas la segmentación semántica con SegNet, aunque el etiquetado toma más tiempo. Otras simulaciones bajo Ros y V-Rep utilizando CNN que clasifica imágenes virtuales pero que se alejan a la aplicación real. Otros avances utilizando máquina de vector soporte para reconocer las líneas de las carreteras con cámaras, asemejándose al modo crucero, pero incapaz de reconocer OI. Propuestas con aprendizaje utilizando redes neuronales multicapa y control de lógica difusa para el frenado y acelerado del automóvil, presentó buenos resultados, a lo que en este proyecto solo se centró en la utilización de la cinemática convencional con fines de dar realismo en la simulación. Simulaciones basadas en Q learning y desarrolladas en Unity, la cual al igual que los juegos de video con aprendizaje reforzado a partir de recompensas, utiliza una CNN, sin embargo, esta solo considera la clasificación de agentes con una CNN, mas no la detección como lo haría la arquitectura YOLO o Faster RCNN siendo más realista la propuesta que plantea este trabajo. Finalmente desarrollos con la arquitectura YOLOv2 y OpenCV para detección de automóviles etiquetando sus propias imágenes junto a otros desarrollos para la detección de automóviles y peatones utilizando la base de datos KITTI, la cual se compara con la faster RCNN, evidenciado por parte de la YOLO, mejores resultados en la velocidad de procesamiento de FPS (casi 40 veces más rápida) y una precisión semejante, a lo que a diferencia de esta investigación (dos OI), en este trabajo se consideran 6 objetos de interés, y se desarrolla una CNN extra para diferenciar el color del semáforo, dándole una mayor aplicabilidad al algoritmo. La propuesta para este trabajo se realiza con herramientas como Matlab y el uso de un simulador de fácil acople como VREP, permitiendo la utilización de imágenes más realistas, la utilización de una base de datos creada para el contexto latinoamericano (a diferencia de KITII que es contexto europeo), y una mayor cantidad de OI, aportando así al estado del arte.

- Se presenta la manera en que se generan dos bases de datos para diferentes aplicaciones, la primera para la utilización de algoritmos que clasifican las imágenes, y la segunda base de datos, para la implementación a eventualidades con varios objetos de interés que se presentan utilizando ROI, siendo esta aplicable para algoritmos de aplicación y detección de OI basada en regiones.
- Para la realización de la simulación se concluye que el uso de V-Rep, como ambiente virtual de fácil acople a Matlab, permite desarrollos más realistas al poder importar imágenes y plasmarlas en paredes simulando un ambiente cercano a la realidad en condiciones controladas. El uso de motores facilita el control de la velocidad angular en las ruedas para cálculos de la cinemática de frenado y aceleración. El importar fácilmente cámaras de profundidad como el KINECT, y el poder caracterizar la resolución de la adquisición de capas RGB como la caracterización de la distancia, permite cálculos que se acercan a la realidad según los componentes utilizados.
- En la comparación de algoritmos, se demostró superioridad en las redes convolucionales frente a técnicas clásicas de ML. La técnica de DL ilustra la superioridad en la clasificación de imágenes para las 7 categorías con la CNN frente a la técnica de ML que utilizó el algoritmo de SVM, el cual se evalúa a partir de las dos matrices de confusión. La comparación realizada detectando verdaderos positivos, se calcula como ((DL-ML) /ML) para cada categoría, demostrando por ejemplo que reconociendo personas la CNN, evidencia una precisión de un 15,83% superior, frente a la técnica SVM de machine learning. Para el cálculo de la diferencia obteniendo falsos negativos, se genera una resta, evidenciando en qué porcentaje se redujo la clasificación errónea de DL frente al ML, mejorando por ejemplo en un 7% al clasificar motos la CNN Alex Net, frente a la técnica de ML, el SVM. La precisión total de la CNN fue de 98.13%, frente a la mejor técnica de ML clásica, en este caso el SVM, el cual obtuvo un 84.2%, superándola en un **16,5%** en precisión general.
- Las redes neuronales convolucionales como clasificadores y las arquitecturas creadas para detección han generado confianza para aplicaciones en proyectos académicos e industriales en el reconocimiento y procesamiento de imágenes gracias a la precisión y sensibilidad semejante a la visión humana sin ocupar gran cantidad recursos computacionales.

- El algoritmo, logra que el conductor se detenga totalmente ante un semáforo en rojo, un pare, y mantenga distancia frente a los objetos de interés (sean carros, bicicletas, motos o personas), evitando el cometer imprudencias.

- El tiempo de procesamiento, puede mejorarse con la utilización de una GPU de mayor capacidad.

- En la aplicabilidad, se aprecia la diferencia en resultados obtenidos por los gráficos de precisión Recall de la base de datos del entrenamiento frente a la base de datos de prueba o validación, presentando mejores resultados en las categorías carro, moto, stop y semáforo obteniendo resultados mayores al 62%. Para la categoría bicicleta, semáforo y persona, no obtuvo tan buenos resultados al obtener resultados inferiores al 57%. Se expone la validación de precisión y recall demostrando que el reconocimiento en carros fue del 98% en el entrenamiento y de 62% con la base de datos de validación, obteniendo una diferencia del 36,7%, stop 94% frente a 68% en validación, persona 88% frente a 41%, moto 94% frente al 72% por nombrar unos casos. Se debe aumentar la cantidad de imágenes para que la base de datos de entrenamiento #2 mejore la precisión de clasificación al extraer más características. Por otro lado, la calidad de las etiquetas mejora, utilizando la base de datos Kitti o Pascal, ya que los resultados en el gráfico de precisión vs sensibilidad, no superó el 50% reconociendo personas y bicicletas.

- Para trabajos futuros, se puede implementar el algoritmo de detección desarrollado en este trabajo, integrado en una tarjeta embebida para un automóvil.

- La capacidad de extracción automática de features en imágenes utilizando aprendizaje con redes neuronales convolucionales, permite una mejor precisión según la calidad y cantidad de imágenes que conforman la base de datos de entrenamiento.
- La arquitectura YOLO y la CNN Resnet50, entregan buenos resultados en velocidad para aplicaciones en tiempo real, gracias al bajo coste computacional, permitiendo su utilización en tarjetas embebidas en aplicaciones por ejemplo de seguridad, movilidad o medicina.
- En velocidad, la Faster RCNN necesitó 1,98 segundos de procesamiento por frame, frente a la YOLOv2 la cual necesito de 0.1264 segundos por frame, siendo esta

segunda, 15 veces más rápida. La YOLO es una opción óptima ya que procesa una sola vez la imagen, obteniendo los cuadros delimitadores, así como las clasificaciones de categorías de objetos, realizando un único procedimiento en paralelo, siendo esta la razón por la que se escoge YOLOv2 como detector.

- La utilización del software de Matlab permite la importación y etiquetado de gran cantidad de imágenes y datos, con la instrucción datastore y el toolbox ImageLabeler, facilitando su procesamiento y análisis con pocas líneas de código.
- El software Matlab, facilita el análisis de varias aplicaciones en aprendizaje supervisado, gracias al uso del tool box Classification Learner, el cual extrae características y evalúa varios algoritmos de aprendizaje automático en paralelo.

# ● Anexos. MATERIALES

El tipo de estudio realizado evaluará la fiabilidad del algoritmo reconociendo objetos urbanos en Bogotá, como asistente para la conducción aportando a mejorar la movilidad y reduzca posibles siniestros viales.

El segundo esbozo presentará un marco conceptual junto a los antecedentes en los carros autónomos y la evolución del aprendizaje profundo. En la 3ra sección se evaluará el tipo de estudio, los sujetos, el método y las herramientas involucradas  El capítulo 4 mostrará los avances en el alcance del desarrollo del trabajo planteado el cual se dividirá en 3 etapas. Se profundizará en el estudio para el reconocimiento de semáforos verde y rojo, usando técnicas de aprendizajes de máquina y aprendizaje profundo, seguido se etiquetarán los seis agentes de tránsito con varios millares encontrados en las calles de Bogotá y se evaluará su funcionamiento con una arquitectura de redes convoluciones, en la tercera etapa se diseñará una interfaz gráfica que cargue videos y evalúe la efectividad del algoritmo y se evidencie el comportamiento en una simulación usando imágenes reales. Finalmente, en el 5to capitulo se presentarán conclusiones y proyectos futuros.

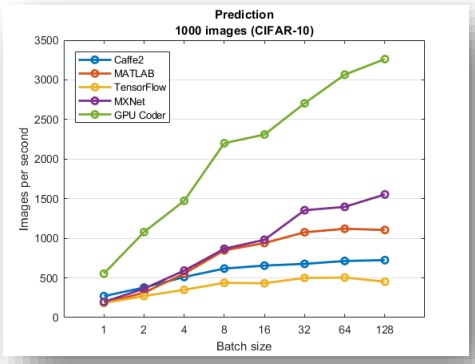
El presente trabajo obedece a la aplicación de un algoritmo de aprendizaje profundo en una simulación para el cual se necesitó la construcción de una base de datos con imágenes etiquetadas para simular los agentes a entrenar, y posteriormente validar con una arquitectura de Deep Learning, la cual evidencie un correcto funcionamiento en la simulación. Para el desarrollo del mismo se proponen 5 pasos metodológicos, los cuales se explican a continuación.

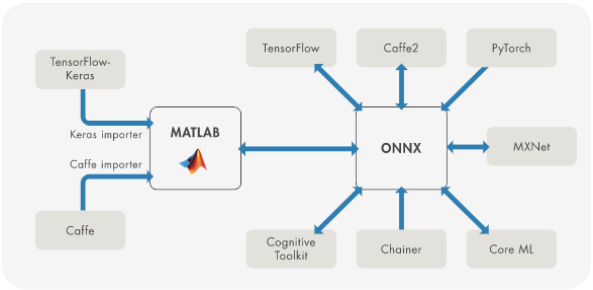
## ○ ¿Por qué MATLAB?

- El entrono a utilizar para entrenar etiquetar, entrenar la red conectar con el entorno virtual y hacer pruebas será Matlab
- MATLAB soporta la **interoperabilidad** con marcos de Deep Learning de código abierto a través de las capacidades de importación y exportación de ONNX, además

de presentar mejores resultados en la velocidad de procesamiento, frente a librerías como Tensor Flow o Caffe2 visualizar en el anexo 1

**Anexo 1 Velocidad de procesamiento e interoperabilidad con Matlab [53]**




## ○ Herramientas e Instrumentos

- Se requirió un computador con GPU que facilite el procesamiento de imágenes a tiempo real.
- Se necesitará un entorno virtual para controlar y simular las redes entrenadas en Matlab.
- Se utilizó un carro convertido a eléctrico categorizado como Hatch Back[5] de 3 puertas Micro car para desde allí hacer la toma de varias imágenes que se utilizaran como data set.

Una aproximación de los precios de los recursos a utilizar y del talento humano se encuentra en el anexo 2

[5] Hatch Back es un auto con un espacio de carga (maletero) integrado, al cual se tiene acceso mediante un portón trasero.

**Anexo 2 Presupuesto de Materiales y Equipos**

| **Recursos** | **Cantidad** | **Precio Unitario (COP)** |
|---|---|---|
| Cámara digital | 1 | $ 300.000 |
| Computador con Tarjeta Gráfica NVIDIA 920mx | 1 | 2000000 |
| ***Subtotal*** | | ***$ 2.300.000*** |
| ***IVA 19%*** | | ***$ 437.000*** |
| ***TOTAL*** | | ***$ 2.737.000*** |

**Anexo 3 Presupuesto Talento Humano**

| **Personal** | **Cantidad** | **Horas al mes** | **Precio Hora-Hombre (COP)** | **Precio Total Mes (COP)** |
|---|---|---|---|---|
| Ingeniero computacional para el etiquetado de imágenes. | 1 | 60 | $ 15.000 | $ 750.000 |
| | | | ***TOTAL /mes*** | **$ 750.000** |

La financiación se realizará con recursos propios anexo 3

**Especificaciones del Hardware a utilizar**

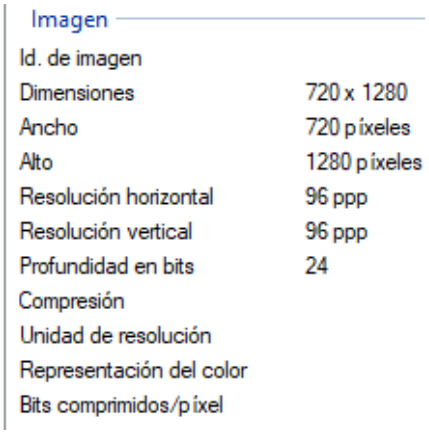
La especificación del hardware, es decir computador, cámara y formato de imágenes se dan en el anexo 4, 5, 6, 7

**Anexo 4 Especificaciones del Hardware a utilizar**

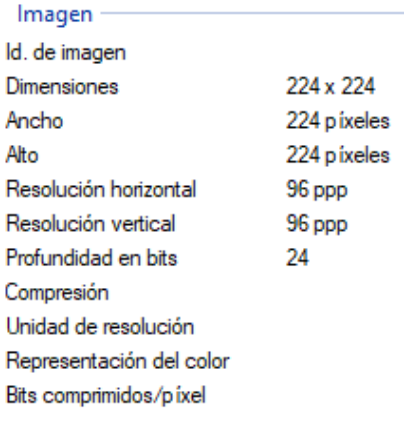
| |
|---|
| **Marca Computador:** ASUS |
| **Modelo:** X441U |
| **Procesador:** Intel i5 7200U @ **2.50GHz (4 CPUs)** |
| **T. Video:** NVidia Gforce GT920MX 2GBDDR3 |
| **Memoria RAM:** 8GB DDR4 |
| **Pantalla:** Full HD 14 Pulgadas |
| **S.0perativo:** Endless Linux |

**Anexo 5 Especificaciones cámara a utilizar**

| |
|---|
| **CÁMARA (PRINCIPAL)** |
| **Resolución modo foto** |
| 12 MP (4000 x 3000) |
| **Resolución modo vídeo** |
| Full HD1080p (1920 x 1080) |
| **FPS máximo de grabación** |
| **Apertura** |
| *f*. 2.2 |




**Anexo 6 Resolución por frame**




**7 Resolución por frame redimensionado para CNN RESNET50**

**Toma de videos desde auto eléctrico**

Para la creación de varios videos, para la elaboración de la segunda base de datos, se requirió un móvil. Desde este, se realizan algunas de las tomas fotográfica y de video. El móvil utilizado para la creación de videos se realiza desde un coche eléctrico convertido por un hombre Colombo lituano[6], el cual cumple con las especificaciones de automóvil categoría tipo micro car[7] marca Fiat Zastava 750 como se muestra en la **Figura 92**


**Figura 92 Auto eléctrico utilizado para tomar videos**

Este móvil se utiliza como medio para la toma fotográfica, debido a sus dimensiones y ser eléctrico, se pretende utilizar en proyectos futuros para la implementación del sistema de asistente para la conducción en una tarjeta embebida de este.

[6] El auto Fiat Zastava 750, es convertido de combustión interna a eléctrico por el ingeniero Jonas Vogulys en 2014

[7] Micro car, auto de dos puertas, Especificaciones Aprox altura 1,5m, ancho 1.5m, largo 3m, velocidad Max 90km/hr, este coche será utilizado para trabajos futuros.

## ○ Sujetos Interesados

Se requirió salida de campo donde se graba la panorámica de un conductor desde un automóvil real en Bogotá, en el que aparecerán personas, autos, bicicletas entre otros objetos en su estado natural para evitar sesgar el análisis y eventualmente poder utilizar estas imágenes en el entrenamiento y validación. (A todos los agentes se les protegerá su identidad).

## ○ Consideraciones éticas

No se considera como riesgo la vida de algún individuo ya que se realizará una simulación. Por otro lado, la propuesta si traerá efectos positivos en la reducción de siniestros viales ya que se propone la utilización de un sistema con este algoritmo que pueda controlar y almacenar las infracciones de cualquier automotor para después ser castigado con multados según el código de tránsito y corrija de una manera conductista a los conductores que sean reiterativos no respetando las normas de tránsito.

# ● BIBLIOGRAFIA


[1] “Cuantos autos hay en el mundo”,2016, [en line], disponible: https://www.hoylosangeles.com/vidayestilo/autos/hoyla-aut-cuantos-autos-hay-en-el-mundo-y-cuantos-se-fabrican-anualmente-20160923-story.html

[2] Forrest, & M. Konca, “Autonomous cars and society,” Worcester Polytechnic Institute, 2007.

[3] M. Aly, "Real time detection of lane markers in urban streets," 2008 IEEE Intelligent Vehicles Symposium, Eindhoven, 2008, pp. 7-12, doi: 10.1109/IVS.2008.4621152 8.

[4] A. Gonçalves, A. Godinho, & J. Sequeira, “Low cost sensing for autonomous car driving in highways,” ICINCO-RA (2), pp. 370-377,2007.

[5] C. Laugier, S. Sekhavat, L. Large, J. Hermosillo, & Z. Shiller, “Some steps towards autonomous cars,” Proceedings of the IFAC Symposium on Intelligent Autonomous Vehicles, pp. 10-18, 2001.

[6] J. M. P. Rastelli, «Agentes de control de vehículos autónomos en entornos urbanos y autovías», p. 197.2016

[7] J. C. Gomez Santofimio, L. D. Diaz Chica, y C. G. Quintero M., «Intelligent Autonomous Driving in a Virtual Environment», en 2019 2nd Latin American Conference on Intelligent Transportation Systems (ITS LATAM), Bogota, Colombia, 2019, pp. 1-7.

[8] I. Thammachantuek, S. Kosolsombat, y M. Ketcham, «Support Vector Machines for Road Images Recognition in Autonomous Car», en 2018 18th International Symposium on Communications and Information Technologies (ISCIT), Bangkok, 2018, pp. 291 293.

[9] Defense Advanced Research Projects Agency (DARPA).2007See

http://www.darpa.mil/grandchallenge. Google Scholar

[10] Schmidhuber, J., 2015. Deep learning in neural networks: An overview. Neural networks, 61, pp.85-117.

[11] N. Eric, et.al, "Kinect depth sensor for computer vision applications in autonomous vehicles," in Ubiquitous and Future Networks (ICUFN), 2017 Ninth International Conference on , Milan, Italy, 2017.

[12] A. Krizhevsky, I. Sutskever, & G. E. Hinton, (2012). ImageNet Classification with Deep Convolutional Neural Networks. Advances in Neural Information Processing Systems 25 (NIPS 2012). pp. 1106– 1114.

[13] Level of driving automation (SAE), online https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles

[14] Paquete de asistencia a la conducción 2019 https://www.mercedes-benz.com.co/passengercars/mercedes-benz-cars/models/gls/suv-x167/safety/assistancesystems/driving-assistance.html

[15] Lidar Sensor, 2019, https://techcrunch.com/2019/03/06/waymo-to-start-selling-standalone-lidar-sensors/

[16] Teichmann, M., Weber, M., Zoellner, M., Cipolla, R., and Urtasun, R. (2018, June). Multinet: Real-time joint semantic reasoning for autonomous driving. In 2018 IEEE Intelligent Vehicles Symposium (IV) (pp. 1013-1020). IEEE.

[17] “Cuando llegaran los carros autónomos a Colombia?, El tiempo”,2017, [en línea], Disponible: https://www.eltiempo.com/tecnosfera/novedades-tecnologia/futuro-de-los-carros-autonomos-en-colombia-93398

[18] «Google saca a la calle a su prototipo de vehículo sin chofer». La Nación. 15 de mayo de 2015. Consultado el 5 de julio de 2015.

[19]Juan Martínez, “El absurdo modelo económico”, EL TIEMPO, http://blogs.eltiempo.com/digital-jumper/2019/02/01/absurdo-modelo-economico-colombiano/ 2019

[20] S. R. Narla, “The evolution of connected vehicle technology: from smart drivers to smart cars to... self-driving cars,” Institute of Transportation Engineers. ITE Journal, vol. 83, no. 7, pp. 22-26, 2013.

[21] S. Duffy, & J. P. Hopkins, “Sit, stay, drive: The future of autonomous car liability,” 2013.

[22]Md. A. Islam y S. I. Rashid, «Algorithm for Ethical Decision Making at Times of Accidents for Autonomous Vehicles», en 2018 4th International Conference on Electrical Engineering and Information & Communication Technology (iCEEiCT), Dhaka, Bangladesh, 2018, pp. 438-442.

[23] Primer atropello mortal de un coche sin conductor». El País. 19 de marzo de 2018. ISSN 1134-6582. Consultado el 19 de marzo de 2018.

[24] T. Sun, S. Tang, J. Wang, & W. Zhang, “A robust lane detection method for autonomous car-like robot,” Fourth International Conference on Intelligent Control and Information Processing (ICICIP), 2013, pp. 373-378, 2013. **

[25] “Las 10 principales causas de Siniestros en Colombia, ELTIEMPO, 2016”, [en línea]Disponible en: https://www.eltiempo.com/multimedia/especiales/principales-causas-de-accidentalidad-en-colombia/16454197/1/index.html)

[26] Error humano en la conducción 2004, en https://www.uv.es/metras/docs/2004_CIT_DBQponencia.pdf

[27] “En Colombia fallecen 15 personas al día, RCNRADIO”,2019, [en línea], Disponible en:https://www.rcnradio.com/colombia/en-colombia-fallecen-15-personas-al-dia-por-accidentes-de-transito.

[28] “Fallecidos, lesionados y accidentes en hechos de tránsito - Colombia, ONSV, 2016-2017P”, [en línea], Disponible en: http://ansv.gov.co/observatorio/public/documentos/boletin.pdf

[29] Movilidad en cifras 2015, secretaria de movilidad en Bogotá https://www.movilidadbogota.gov.co/web/SIMUR/ARCHIVOS/Movilidad_Cifras_2015_V4_marzo2017.pdf

[30] Unknown, "Top 25 Causes of Car Accidents," Law Offices of Michael Pines, APC, [Online]. Available: https://seriousaccidents.com/legaladvice/top-causes-of-car-accidents/. [Accessed 23 9 2017].

[31] Unknown, "Global Status Report on Road Safety 2015," World Health Organization, 2015. [Online]. Available: http://www.who.int/violence_injury_prevention/road_safety_status/2015/GSRRS2015_Summary_EN_final2.pdf?ua=1. [Accessed 23 9 2017].

[32] Zhu, Y., Mottaghi, R., Kolve, E., Lim, J. J., Gupta, A., Fei-Fei, L., and Farhadi, A. (2017, May). Target-driven visual navigation in indoor scenes using deep reinforcement learning. In Robotics and Automation (ICRA), 2017 IEEE International Conference on (pp. 3357-3364). IEEE.

[33] H.-s. Lee, et.al, "A Real-Time System for Detecting Illegal Changes-of-Lane Based on Tracking of Feature Points," in Vehicular Technology Conference, Taipei, Taiwan, 2010.

[34] E. U. Haq, Y.lui, "Image processing and vision techniques for smart vehicles," in Circuits and Systems (ISCAS), Seoul, South Korea , 2012.

[35] J. Guivant, et.al, "A light-weight yet accurate localization system for autonomous cars

in large-scale and complex environments," in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, 2016.

[36] M. Mancini, et.al, "Fast robust monocular depth estimation for Obstacle Detection with fully convolutional networks," in Intelligent Robots and Systems (IROS), Daejeon, South Korea, 2016.

[37] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In CVPR09, 2009.

[38] P. Pasquel y C. Pamela, «Diseño de un sistema de reconocimiento automático de vehículos mediante el uso de redes neuronales profundas (DNN)», may 2019.

[39] Vega, H., Cortez, A., Alarcón, L., & Romero, P. (2009). Vega H., Cortez A, Alarcón Reconocimiento de patrones mediante redes neuronales artificiales. Perú: Universidad Nacional Mayor de San Marcos.

[40] Caballero, Modelado y seguimiento de objetos por medio de distribución del color. Mexico. (2001).

[41] Salomón, N., Misller, V., & Miranda, R, Análisis digital de descriptores de color en trigo. Revista Internacional de Botánica Experimental, 306-311. (2015).

[42] Martínez, F., Gómez, F., & Romero, E. Análisis de vídeo para estimación del movimiento humano. Colombia: Universidad Nacional de Colombia. (2009).

[43] Reconocimiento de objetos por descriptores de forma. España: Universidad de Barcelona.2008

[44] Ciriza, R., Albizua, L., & Gonzales, M. Análisis de la utilidad de los descriptores texturales de haralick para la localización arranques de frutal en ortofoto. XIII Congreso de la Asociación Española de Teledetección, (págs. 597-600). España (2009).

[45] Malavé, J., Márquez, E., & Lezama, J. Cálculo de descriptores moleculares topológicos en flavonoides con actividad ANTI-VIH-1. SABER. Revista Multidi (2015).

[46] Bergamini, M., & Kamlofsky, J. Estudio de descriptores geométricos para el diseño y optimización de algoritmos de reconocimiento de objetos digitales. Argentina: Universidad abierta Interamericana. (2015).

[47] Solera, P., Moreira, I., & Hernández, J. Descriptores botánicos para caracterizar germoplasmas de Ricinus communis de diferentes zonas de Costa Rica. Revista Tecnología en Marcha, 37-46. (2014).

[48] H., & Martínez, A. Inteligencia artificial: Redes neuronales y aplicaciones. España: Universidad Carlos III de Madrid. (2015).

[49] Sistemas Expertos e Inteligencia artificial. Obtenido de Universidad Don Bosco: (2016). http://www.udb.edu.sv/udb/archivo/guia/informatica-ingenieria/sistemas-expertos-einteligencia-artificial/2016/i/guia-9.pdf

[50] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 15(1), pp. 1929-1958.

[51] Goodfellow, I., Bengio, Y., Courville, A., and Bengio, Y. (2016). Deep learning (Vol. 1). Cambridge: MIT press.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, & A. Rabinovich. (2015). Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, 2015, pp. 1-9.

[53] Matlab https://la.mathworks.com/discovery/deep-learning.html#whyitmatters

[54] PALOMARES, Fernando Giménez; SERRÁ, Juan Antonio Monsoriu; MARTÍNEZ, Elena Alemany. Aplicación de la convolución de matrices al filtrado de imágenes. Modelling in Science Education and Learning, 2016, vol. 9, no 1, p. 97-108.

[55] Y. LeCun, Y. Bengio, and G. Hinton. “Deep learning,” Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[56] WOLF, Fabian. Densely Connected Convolutional Networks for Word Spotting in Handwritten Documents. Tesis de Maestría, Universidad Técnica de Dortmund, 2018

[57]Günel, M. GoogleNet. (2016). Obtenido de https://pdfs.semanticscholar.org/0b99/d677883883584d9a328f6f2d54738363997a.pdf

[58] Zeiler, M. D., and Fergus, R. (2014, September). Visualizing and understanding convolutional networks. In European conference on computer vision (pp. 818-833). Springer, Cham.

[59] Ioffe, S., and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[60] Wu, Y., and He, K. (2018). Group normalization. arXiv preprint arXiv:1803.08494.

[61] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779–788).

[62] «Optimizing CNN-based Object Detection Algorithms on Embedded FPGA Platforms

[63]B. C. Zanchin, R. Adamshuk, M. M. Santos, y K. S. Collazos, «On the instrumentation and classification of autonomous cars», en 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Banff, AB, 2017, pp. 2631-2636.

[64] Gomes, L. “Hidden Obstacles For Google’s Self-Driving Cars: Impressive Progress Hides Major Limitations Of Google’s Quest For Automated Driving,” MIT Technological Review, 28 August 2014; Access in March, 2017, at https://www.technologyreview.com/s/530276/hidden-obstacles-forgoogles-self-driving-cars/

[65] ITIF, “The Road Ahead: The Emerging Policy Debates for IT in Vehicles,” Information Technology & Innovation Foundation, Access in March, 2017, at http://www2.itif.org/2013-road-ahead.pdf

[66]APA, “Planning for the Autonomous Vehicle Revolution”, American Planning Association Blog; Access in March, 2017, at https://www.planning.org/blog/blogpost/9105024/

[67] S. Chakraborty, H. Laware, D. Castanon, e S. R. Zekavat, “High precision localization for autonomous vehicles via multiple sensors, data fusion and novel wireless technologies”, in Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), IEEE Annual, 2016, p. 1–9

[68] Autopilot Tesla https://www.tesla.com/es_MX/autopilot?redirect=no

[69] Aish, D. “Stereo vision - Facing the challenge and seeing the opportunities for ADAS applications,” Texas Instruments, 2016, Acces in March, 2017, at http://www.ti.com/lit/wp/spry300/spry300.pdf

[70] LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W. and Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural computation, 1(4), pp.541-551.

[71] Y. Jia, E. Shelhamer, J. Donahue, et al., “Caffe: Convolutional architecture for fast feature embedding,” arXiv preprint arXiv:1408.5093, 2014.

[72] Lampert, C. AlexNet. Obtenido de Institute of Science and Technology Austria: http://cvml.ist.ac.at/courses/DLWT_W17/material/AlexNet.pdf (2017).

[73] Y. Bengio, Learning deep architectures for AI. Foundations and Trends in Machine Learning 2, 1–127 (2009).

[74] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M.Blau,and S. Thrun, “Dermatologist-level classification of skin cancer with deep neural networks,” Nature, vol. 542, no. 7639, pp. 115–118, 2017.

[75] X. Zhang et al., "Classification of mammographic masses by deep learning," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 2017, pp. 793-796, doi: 10.23919/SICE.2017.8105545205

[76] C. Spille, S.D. E Wert, B. Kollmeier, B.Meyer, Predicting speech intelligibility with deep neural networks, Computer Speech & Language 48(2018) 51-66.

[77] M. Y. Day and Y. D. Lin, "Deep Learning for Sentiment Analysis on Google Play Consumer Review," 2017 IEEE International Conference on Information Reuse and Integration (IRI), San Diego, CA, USA, 2017, pp. 382-388, doi: 10.1109/IRI.2017.

[78] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre et al., “Mastering the game of go with deep neural networks and treesearch,” Nature, vol.529, no. 7587, pp. 484–489, 2016.

[79] K. M. I. Khalilullah, S. Ota, T. Yasuda and M. Jindai, "Development of robot navigation method based on single camera vision using deep learning," 2017 56th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), Kanazawa, Japan, 2017, pp. 939-942, doi: 10.23919/SICE.2017.8105675

[80] ABDEL-ZAHER, Ahmed M.; ELDEIB, Ayman M. Breast cancer classification using deep belief networks. Expert Systems with Applications, 2016, vol. 46, p. 139-144.

[81] PARKHI, Omkar M., et al. Deep Face Recognition. En BMVC. 2015. p. 6.

[82] TAJBAKHSH, Nima, et al. Convolutional neural networks for medical image analysis: Full training or fine tuning?. IEEE transactions on medical imaging, 2016, vol. 35, no 5, p. 1299-1312.

[83] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, & R. R. Salakhutdinov, (2012). Improving neural networks by preventing coadaptation of feature detectors. (https://arxiv.org/pdf/1207.0580.pdf accessed on Aug. 20, 2017).

[84] Milioto, A., Lottes, P., and Stachniss, C. (2018, May). Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in CNNs. In 2018 IEEE International Conference on Robotics and Automation (ICRA) (pp. 2229-2235). IEEE

[85] Ross Girshick, Jeff Donahue, Trevor Darrell, et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, IEEE Conference on Compute Vision and Pattern Recognition, arXiv:1311.2524v5 [cs.CV] (2014).

[86] Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In Advances in neural information processing systems (pp. 91-99).

[87] ARENAS, Javier O. Pinzón; MORENO, Robinson Jiménez; MURILLO, Paula C. Useche. Hand Gesture Recognition by Means of Region-Based Convolutional Neural Networks. Contemporary Engineering Sciences, 2017, vol. 10, no 27, p. 1329-1342.

[88] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection." Coference on Computer Vision and Pattern Recognition (CVPR), 2005.

[89] Yang, S., Wang, J., Wang, G., Hu, X., Zhou, M. and Liao, Q., 2017, December. Robust RGB-D SLAM in dynamic environment using faster R-CNN. In Computer and Communications (ICCC), 2017 3rd IEEE International Conference on (pp. 2398-2402). IEEE.

[90] P. Viola and M. Jones. "Robust real-time face detection." International Journal of CompuTer Vision, vol.57,no.2, pp.137-154, 2004.

[91] P. Felzenszwalb, R.Girshick, D. McAllester, and et al, "Object detection with discriminatively trained partbased models." in PAMI, vol.32, pp. 1627-1645, Jul. 2010.

[92] K. He, X. Zhang, S. Ren, and et. al, "Spatial pyramid pooling in deep convolutional networks for visual recognition." in Proc. Eur. Conf. Comput. Vis., pp. 346-361, 2014.

[93] J. Dai, Y. Li, K. He, and et. al, "R-FCN: Object detection via regionbased fully convolutional networks." in Proc. Conf. Neural Inf. Process. Syst. (NIPS), pp. 379-387, 2016.

[94] P. Dollar, C. Wojek, B. Schiele, and et. al, "Pedestrian detection: An evaluation of the state of the art. " in PAMI, vol. 99, 2011. 3. http://pascal.inrialpes.fr/data/human/

[95]J. Dai, Y. Li, K. He, and et. al, "R-FCN: Object detection via regionbased fully convolutional networks." in Proc. Conf. Neural Inf. Process. Syst. (NIPS), pp. 379-387, 2016.

[96] W. Liu, D. Anguelov, D. Erhan, and et. al, "SSD: Single shot multibox detector" in Proc. Eur. Conf. Comput. Vis., pp. 21-37, 2016.

[97]Z. Yang, J. Li, y H. Li, «Real-time Pedestrian and Vehicle Detection for Autonomous Driving», en 2018 IEEE Intelligent Vehicles Symposium (IV), Changshu, 2018, pp. 179-184.

[98] Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. ArXiv Preprint.

[99] Chang, Y. H., Chung, P. L., and Lin, H. W. (2018, April). Deep learning for objectidentification in ROS-based mobile robots. In 2018 IEEE International Conferenceon Applied System Invention (ICASI) (pp. 66-69). IEEE.

[100] «De costa a costa sin conductor». El Mundo. 27 de marzo de 2015. Consultado el 5 de julio de 2015.

[101] Los retos del coche autónomo». ABC. 3 de julio de 2015. Consultado el 5 de julio de

2015.

[102] «Así es el auto sin conductor que corre a 240 km por hora». La Nación. 22 de octubre de 2014. Consultado el 5 de julio de 2015.

[103] «Un auto sin chofer atravesó Estados Unidos casi sin ayuda humana». La Nación. 7 de abril de 2015. Consultado el 5 de julio de 2015.

[104] Uber estrena su coche sin conductor en San Francisco». 2016. Consultado el 2016.

[105] Ruiz, Jesús (26 de enero de 2014). «“Los coches robot no beben”». El País. Consultado el 5 de julio de 2015.

[106] Acosta, A. (6 de julio de 2015). «MCity, un pueblo de 'atrezzo' para probar coches sin conductor». La Vanguardia. Consultado el 24 de julio de 2015.

[107] «Estudio del Laboratorio Berkeley concluye que esta será la opción más verde y económica a partir de 2030». El Mundo. 24 de julio de 2015. Consultado el 24 de julio de 2015.

[108] Kogan, Enrique (17 de julio de 20015). «Audi avanza hacia el autoconducción». La Opinión. Consultado el 24 de julio de 2015.

[109] Bosredon, Mickaël (10 de junio de 2015). «Bordeaux: Des véhicules autonomes seront intégrés à la circulation en octobre». 20 minutos (en francés). Consultado el 5 de julio de 2015.

[110] Mary Slosson (8 de mayo de 2012). «Google gets first self-driven car license in Nevada». Reuters. Consultado el 9 de mayo de 2012.

[111] Kessler, Aaron (17 de mayo de 2015). «El vacío legal detrás de los vehículos autónomos». The New York Times. Consultado el 5 de julio de 2015.

[112] H. Drezet, S. Colombel, y M.-L. Avenel, «Human-Man Interface Concept For Autonomous Car», en 2019 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 2019, pp. 1-5.

[113] Fan, Q., Brown, L. and Smith, J., 2016, June. A closer look at Faster R-CNN for vehicle detection. In Intelligent Vehicles Symposium (IV), 2016 IEEE (pp. 124-129). IEEE.

[114] N. Yang, M.Zhou, "Autonomous Overtaking Behavior Simulation for Autonomous Virtual Vehicle in Virtual Traffic Environment," in Computer Science and Software Engineering, China, 2008. **

[115] Krishnan, P. (2018). Design of Collision Detection System for Smart Car Using Li-Fi and Ultrasonic Sensor. IEEE Transactions on Vehicular Technology.

[116] S. J.Mankar, et.al, "Design of computer vision intelligent system for lane detection," in Green Engineering and Technologies (IC-GET), Coimbatore, India, 2016.

[117] NURHADIYATNA, Adi; LONČARIĆ, Sven. Semantic image segmentation for pedestrian detection. En Image and Signal Processing and Analysis (ISPA), 2017 10th International Symposium on. IEEE, 2017. p. 153-158.

[118] Wei, W., He, S., Wang, D. and Yeboah, Y., 2018, June. Multi-Objective Deep CNN for Outdoor Auto-Navigation. In Proceedings of the 2018 2nd International Conference on Deep Learning Technologies (pp. 81-85). ACM.

[119] Koutník, J. Schmidhuber and F. Gomez, "Online Evolution of Deep Convolutional Network for Vision-Based Reinforcement Learning", From Animals to Animats 13, vol.

8575, pp. 260-269, 2014.

[120] Sourceforge. “TORCS - The Open Racing Car Simulator”. Available: https://sourceforge.net/projects/torcs

[121]ROS Web page, http://www.ros.org/

[122] VanDung Nguyen, Oanh Tran Thi Kim, Tri Nguyen Dang, Seung Il Moon, y C. S. Hong, «An efficient and reliable Green Light Optimal Speed Advisory system for autonomous cars», en 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, Japan, 2016, pp. 1-4.

[123] T. Okuyama, T. Gonsalves, y J. Upadhay, «Autonomous Driving System based on Deep Q Learnig», en 2018 International Conference on Intelligent Autonomous Systems (ICoIAS), Singapore, 2018, pp. 201-205.

[124] K. Arulkumaran, N. Dilokthanakul, M. Shanahan, and A. A. Bharath, “Classifying options for deep reinforcement learning,” in Proc. IJCAI Workshop Deep Reinforcement Learning: Frontiers and Challenges, 2016.

[125] C. Otero et al., «SIMULACIÓN DE VEHÍCULOS AUTÓNOMOS USANDO V-REP BAJO ROS», p. 8. 2017.

[126] Coulter, R., (1992) Implementation of the pure pursuit path-tracking algorithm, Carnegie Mellon University, the Robotics Institute, Pittsburgh, Pa.

[127] Jazar, R., (2014) Vehicle dynamics (2nd ed.), New York, Springer New York.

[128] Romanov, N., Johnson, C.E., Case, J.P., Goel, D., Gutmann, S. and Dooley, M., iRobot Corp, 2015. Mobile robot for cleaning. U.S. Patent 8,961,695

[129] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G.Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” Nature, vol. 518, no. 7540, pp. 529–533, 2015.

[130] “Global Traffic Scorecard, Irix”,2017, [en line], disponible: http://inrix.com/scorecard/

[131] BADRINARAYANAN, Vijay; KENDALL, Alex; CIPOLLA, Roberto. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, vol. 39, no 12, p. 2481-2495.

[132] A. Geiger, P. Lenz, C. Stiller, and et. al, "Vision meets robotics: The kitti dataset." International Journal of Robotics Research (IJRR),vol.32,no.11,pp.1231-1237, 2013. http://www.cvlibs.net/datasets/kitti/

[133] K. He, X. Zhang, S. Ren, and et. al, "Deep residual learning for image recognition." in Proc. IEEE Conf. Comput. Vis. Pattern Recognit. pp.770-778, 2016.

[134] “Tesla reconoce otro accidente mientras utilizaba el piloto automatico”, EL PAIS, https://elpais.com/tecnologia/2019/05/17/actualidad/1558075375_210626.html

[135] Matlab https://la.mathworks.com/help/vision/ug/getting-started-with-yolo-v2.html

[136] Hui, JReal-time Object Detection with YOLO, YOLOv2 and now YOLOv3. Obtenido de Medium Corporation: https://medium.com/@jonathan_hui/realtime-object-detection-with-yolo-yolov2-28b1b93e2088. (17 de Marzo de 2018).

[137] Sourceforge. “TORCS - The Open Racing Car Simulator”. Available: https://sourceforge.net/projects/torcs

138] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. International Journal of Computer Vision, 88(2), 303–338.

[139] T. Lillicrap, J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver and D. Wierstra, "Continuous control with deep reinforcement learning", International Conference on Learning Representations, San Juan, Puerto Rico, May 2016.

[140] D. Loiacono, L. Cardamone, P. Lanzi, “Simulated Car Racing Competition Manual”,

Politecnico di Milano, 2013. [Online]. Available: https://arxiv.org/pdf/1304.1672.pdf.

[141] V-REP simulator Web page, http://www.coppeliarobotics.com/

[142] Nogueira, L “Comparative Analysis between Gazebo and V-REP Robotic Simulators”, Seminario Interno de Cognicao Artificial - SICA 2014, pp. 5

[143] Cebrián, L. Reconocimiento de emociones mediante técnicas de aprendizaje profundo. Valencia: Universidad Politécnica de Valencia (2015-2016).

[144] A. Ćorović, V. Ilić, S. Đurić, M. Marijan, B. Pavković (2018). The Real-Time Detection of Traffic Participants Using YOLO Algorithm.

[145] I. Mohamed, M. Ahmed, M. Mohamed, R. Mahmoud, H Abd, M. Ghoneima, M. Saeed H. Mostafa (2018) Real-Time Car Detection-Based Depth Estimation Using Mono Camera

**Para ver videos en youtube - Pruebas TESIS**

https://www.youtube.com/channel/UC7FJip3-QC309JiQpOKqZpg?view_as=subscriber