
# DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

## CARRERA DE INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

**TRABAJO DE TITULACIÓN, PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES**

**TEMA: "IMPLEMENTACIÓN DE UN PROTOTIPO PARA CONTAR PERSONAS EN VIDEO CAPTURADO EN TIEMPO REAL UTILIZANDO TÉCNICAS DE APRENDIZAJE PROFUNDO."**

**AUTOR: PARREÑO ESCOBAR, JUAN FERNANDO**

**DIRECTOR: ING. SILVA TAPIA, RODRIGO**

**SANGOLQUÍ**

**2020**

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES**

**CARRERA DE INGENIERIA EN ELECTRÓNICA Y TELECOMUNICACIONES**

## CERTIFICACIÓN

Certifico que el trabajo de titulación, ***"IMPLEMENTACIÓN DE UN PROTOTIPO PARA CONTAR PERSONAS EN VIDEO CAPTURADO EN TIEMPO REAL UTILIZANDO TÉCNICAS DE APRENDIZAJE PROFUNDO"*** realizado por el señor ***Parreño Escobar Juan Fernando***, ha sido revisado en su totalidad y analizado por el software anti-plagio, el mismo cumple con los requisitos teóricos, científicos, técnicos, metodológicos y legales establecidos por la Universidad de Fuerzas Armadas ESPE, por lo tanto me permito acreditarlo y autorizar al señor ***Parreño Escobar Juan Fernando***, para que lo sustente públicamente.

**Sangolquí, 15 de Enero de 2020**

**Ing. Rodrigo Silva Tapia**
C.C.: 0602199523

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES**

**CARRERA DE INGENIERIA EN ELECTRÓNICA Y TELECOMUNICACIONES**

***AUTORÍA DE RESPONSABILIDAD***

*Yo,* ***Parreño Escobar Juan Fernando****, con cedula de identidad Nº 1722147434, declaro que este trabajo de titulación* ***"IMPLEMENTACIÓN DE UN PROTOTIPO PARA CONTAR PERSONAS EN VIDEO CAPTURADO EN TIEMPO REAL UTILIZANDO TÉCNICAS DE APRENDIZAJE PROFUNDO"*** *ha sido desarrollado considerando los métodos de investigación existentes, así como también se ha respetado los derechos intelectuales de terceros considerándose en las citas bibliográficas.*

*Consecuentemente declaro que este trabajo es de mi autoría, en virtud de ello me declaro responsable del contenido, veracidad y alcance de la investigación mencionada.*

**Sangolquí, 15 de Enero de 2020**

.....

**Juan Fernando Parreño**
C.C.: 1722147434

ESPE
UNIVERSIDAD DE LAS FUERZAS ARMADAS
INNOVACIÓN PARA LA EXCELENCIA

**DEPARTAMENTO DE ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES**

**CARRERA DE INGENIERIA EN ELECTRÓNICA Y TELECOMUNICACIONES**

**AUTORIZACIÓN**

*Yo,* ***Parreño Escobar Juan Fernando,*** *autorizo a la Universidad de las Fuerzas Armadas ESPE publicar en la biblioteca Virtual de la institución el presente trabajo de titulación* ***"IMPLEMENTACIÓN DE UN PROTOTIPO PARA CONTAR PERSONAS EN VIDEO CAPTURADO EN TIEMPO REAL UTILIZANDO TÉCNICAS DE APRENDIZAJE PROFUNDO"*** *cuyo contenido, ideas y criterios son de mi autoría y responsabilidad.*

**Sangolquí, 15 de Enero de 2020**

**Juan Fernando Parreño**
C.C.: 1722147434

## DEDICATORIA

Este trabajo está dedicado para mi madre Teresa Escobar y mi padre Antonio Parreño que físicamente ya no está aquí pero siempre está presente en mis pensamientos y deseo que desde el cielo se sienta muy orgulloso. Gracias madre por ser mi pilar y soporte en la vida para seguir adelante en todo este proceso educativo. De igual forma este trabajo va dedicado para mi Mujer y mi hijo Elias Fernando que al momento recién es un bebe, pero anhelo en cierto momento cuando te encuentres grande sepas que eres mi inspiración y mi fuerza.

Juan Fernando Parreño Escobar.

## AGRADECIMIENTO

Mi agradecimiento más profundo y sincero es a mi madre por su incondicional apoyo en todas las situaciones que se han presentado, gracias por esos sabios consejos y enseñanzas que me ha brindado con la intención de que sea un hombre de bien, gracias por la paciencia, por su amistad e incondicional amor.

Un agradecimiento especial para mis hermanos José Parreño y Pablo Parreño que a lo largo de este proceso educativo siempre me han brindado su apoyo y conocimientos para poder lograr estas metas propuestas en mi vida.

Un agradecimiento especial a mi mujer Jeniffer Nivelo por el apoyo incondicional en este proceso, por sus palabras de motivación que me ayudaron mucho para lograr cumplir esta meta.

Un agradecimiento especial a mi tutor Ing. Rodrigo Silva por saber guiarme en este proceso, por la paciencia y ayuda que me brindo.

Juan Fernando Parreño Escobar

**ÍNDICE DE CONTENIDO**

## ÍNDICE DE TABLAS

## ÍNDICE DE FIGURAS

# RESUMEN

El desarrollo de dispositivos de visión artificial para la detección y seguimiento de objetos en video requiere la utilización de algoritmos complejos implementados en sistemas computacionales de alto rendimiento. Aunque existen un sinnúmero de técnicas holísticas y de machine learning utilizadas para la detección de objetos, hoy en día, la tendencia es utilizar nuevas técnicas de procesamiento de imágenes y video basadas en redes neuronales convolucionales, con las cuales se han logrado mejores resultados en cuanto a la confiabilidad de la detección y el tiempo de procesamiento. En este trabajo se implementó un prototipo para realizar el conteo de personas entrelazando un algoritmo de detección de objetos pre entrenado Single Shot Detection (SSD) de MobilNet con un algoritmo de seguimiento de objetos basado en la estimación del centroide, operando en un hardware dedicado de Nvidia Jetson Tx1. La programación del prototipo se realiza utilizando librerías de Python y OpenCV bajo el entorno del sistema operativo linux Ubuntu instalados en la tarjeta Nvidia. En la evaluación del contador de personas se evidencia el equilibrio logrado entre la rapidez, la precisión y el costo computacional del prototipo funcionando en tiempo real bajo un escenario con ambiente de luz natural y luz artificial con resultados muy aceptables.

**PALABRAS CLAVE:**

- **VISION ARTIFICIAL**
- **CONTADOR DE PERSONAS**
- **JETSON TX1**
- **SINGLE SHOT DETECTION (SSD)**
- **CENTROIDE**

## ABSTRACT

The development of artificial vision devices for the detection and tracking of objects in video requires the use of complex algorithms implemented in high performance computing systems. Although there are a number of holistic and machine learning techniques used for object detection, today, the tendency is to use new image and video processing techniques based on convolutional neural networks, with which better results have been achieved in terms of to the reliability of detection and processing time. In this work, a prototype is implemented to count people by interlacing a pre-trained Single Shot Detection (SSD) object detection algorithm with an object tracking algorithm based on the centroid detection, operating on a hardware dedicated by Nvidia Jetson Tx1. Prototype programming is done using Python and OpenCV libraries under the Linux Linux operating system environment connected to the Nvidia card. The evaluation of the people counter shows the balance achieved between the speeds, precision and computational cost of the operating prototype in real time under a scenario with natural light and artificial light environment with very acceptable results.

**KEY WORDS:**

- **ARTIFICIAL VISION**
- **PEOPLE COUNTER**
- **JETSON TX1**
- **SINGLE SHOT DETECTION (SSD)**
- **CENTROID**

# CAPITULO 1

# 1. INTRODUCCIÓN

## 1.1. Motivación

El aprendizaje profundo o deep learning es un campo de la Inteligencia Artificial cuyo objetivo es hacer que las máquinas emulen ciertas tareas realizadas por una persona. Para ello, las máquinas deben pasar por un proceso de entrenamiento para configurar los parámetros que les permitan luego ejecutar tareas con resultados esperados dentro de un proceso de toma de decisiones.

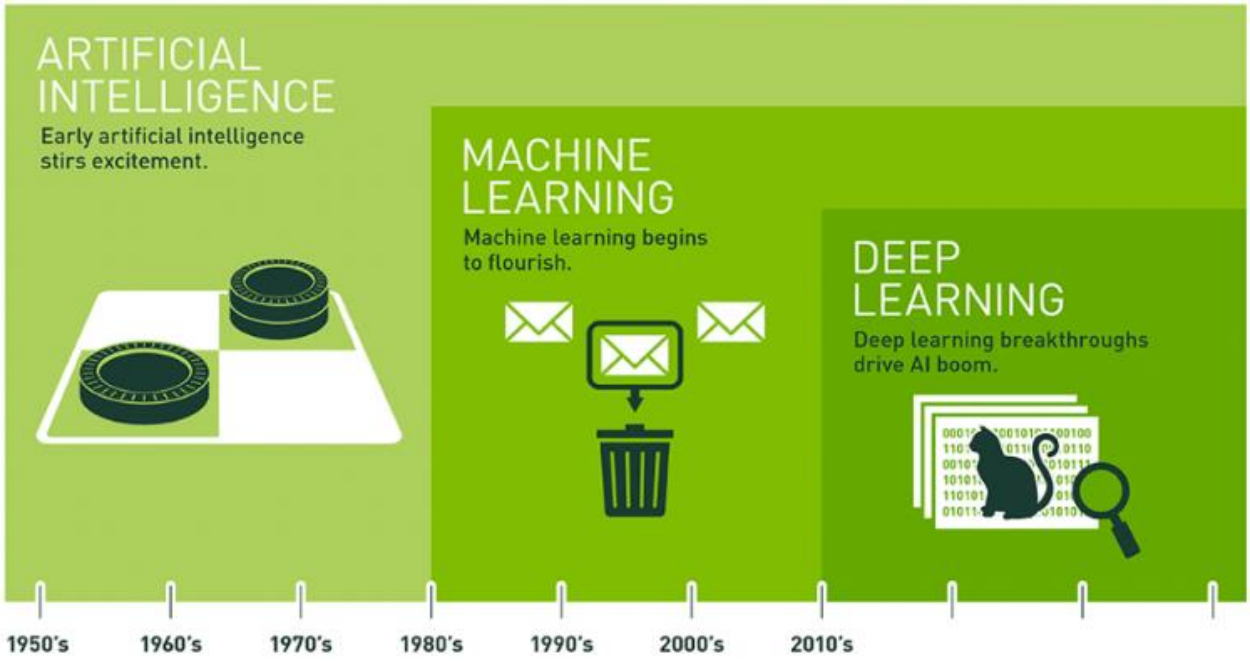



***Figura 1.*** Evolución de la Inteligencia artificial.

Fuente: (SmartPanel, 2019)

Los algoritmos de deep learning se enfocan en capas de unidades de proceso (redes neuronales convolucionales), son estructuras que intentan imitar la organización del sistema neuronal del

humano, gracias a esta configuración el algoritmo tiene la capacidad de aprender y lograr la detección de un objeto con una confiabilidad aceptable, teniendo en cuenta que entre más grande es la base de datos de imágenes y videos en diferentes ambientes mejor es el entrenamiento del detector.

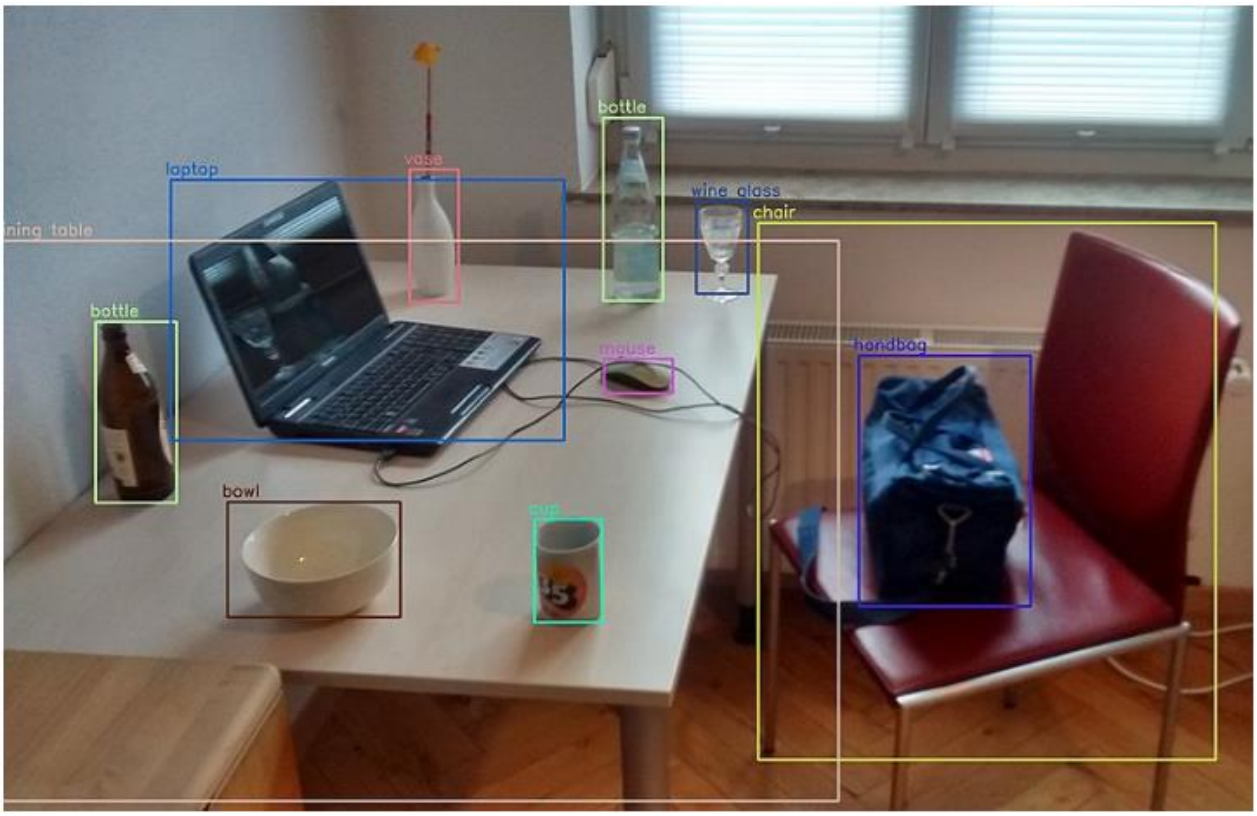



***Figura 2.*** Funcionamiento de Deep Learning, reconocimiento de objetos.
Fuente: (Atriainnovation, 2019)

Son varias las aplicaciones en donde se utilizan actualmente técnicas de aprendizaje profundo, la medicina es un campo en donde ayuda en un diagnóstico médico, otra aplicación es en el servicio de traducción automática de texto de Google, el reconocimiento de voz en los asistentes de Apple y Windows, la seguridad a través de sistemas de videovigilancia inteligentes, etc.

En particular, la videovigilancia es un campo en el que cada día aumenta el número de cámaras de video instaladas alrededor de todo el mundo. Por ejemplo, grandes ciudades como Londres o Chicago disponen de amplias redes de cámaras con las que pretenden minimizar la criminalidad con capacidad para la detección de presencia de personas, seguimiento, evaluó de conducta o detección facial. Otras aplicaciones de la videovigilancia son el reconocimiento de intrusos en zonas de seguridad, el guiado de vehículos, el control de tráfico en carreteras, aduanas, puertos y aeropuertos, vigilancia de procesos industriales, control de multitudes, etc. (Izquierdo, 2015)

La ejecución de este algoritmo para la detección de un objeto como se ha explicado su estructura requiere un gran costo computacional, en el presente trabajo se incluirá también el seguimiento del objeto, para aquello se requiere de un hardware dedicado que pueda soportar el procesamiento en tiempo real y se pueda obtener resultados aceptables, para ello se va a utilizar el kit de desarrollo Nvidia Jetson Tx1 que es especializado para estas aplicaciones.

La visión artificial es un campo tecnológico desafiante, pues en las características de las imágenes pueden incluir las deformaciones del objetivo, las variaciones de iluminación, los cambios de escala, el movimiento rápido y abrupto, las oclusiones parciales, el desenfoque de movimiento, la deformación del objeto y los ruidos de fondo siendo estos algunos de los principales elementos a tomarse en cuenta en los métodos para la detección y seguimiento de objetos. (Guanghan, Zhi, Chen, & Zhihai, 2016)

## 1.2. Justificación e Importancia

En ciertas actividades cotidianas domésticas o industriales, puede ser necesario disponer de aplicaciones de video con la capacidad de contar objetos de manera automática, más aún cuando se trata de realizar esta tarea en periodos de tiempo prolongados. Aunque el problema de contar objetos de manera automática puede resolverse de distintas maneras, una de las más modernas y efectivas puede implementarse utilizando algoritmos de visión artificial incorporados en sistemas con cámaras de video.

El presente trabajo pretende contribuir al desarrollo del campo seguridad en el programa SMART CAMPUS de la Universidad de las Fuerzas Armadas (ESPE), implementando un pequeño sistema autónomo de visión artificial que contabiliza en tiempo real el número de personas que entran y salen a través de la garita de la entrada principal.

## 1.3. Alcance del Proyecto

En la actualidad todo se está automatizando y estamos entrando a una etapa SMART, con lo cual este trabajo se enfoca principalmente en dar un primer paso en el programa SMART CAMPUS de la Universidad de las Fuerzas Armadas (ESPE), implementando un prototipo autónomo capaz de realizar el conteo de personas captadas por una cámara de video bajo en el entorno IP que entran y salen a través de la garita principal de la ESPE. El algoritmo pre entrenado a utilizar para la detección es el Single Shot Detection de MobileNet (SSD) el cual por sus características en comparación a otros nos beneficia por su rapidez y lo más importante su menor costo computacional ya que es necesario acoplar al algoritmo de seguimiento que se basa en el Centroide

que nos permite identificar y reconocer la acción de la persona dentro del marco de entrada de video.

Los resultados obtenidos en este trabajo nos favorecerán disponer de un prototipo autónomo independiente que se podrá utilizar para un posterior desarrollo o aplicación como ejemplo de seguridad, restringiendo el acceso a personas a un sitio determinado. Realizado diferentes pruebas se evaluará los parámetros óptimos de operación del prototipo considerando el nivel de iluminación del ambiente, la ubicación de la cámara, los errores de detección y cuenta obtenidos en las pruebas realizadas bajo condiciones de distintos ambientes controlados.

## 1.4. Objetivos

### 1.4.1. General

Implementar un prototipo que combine algoritmos de detección y seguimiento de objetos para contar personas identificadas en video capturado en tiempo real.

### 1.4.2. Objetivos Específicos.

- Investigar el fundamento teórico y práctico del aprendizaje profundo (Deep learning) aplicado para la detección y seguimiento de objetos presentes en un flujo de video.
- Implementar un contador de personas utilizando herramientas computacionales NVIDIA Jetson TX1, OpenCV y Python.
- Evaluar la efectividad del contador de personas en distintos escenarios controlados.

- Optimizar los parámetros de funcionamiento del prototipo operando en distintos escenarios controlados.

## 1.5. Estado del Arte

### 1.5.1. Aplicación basado en aprendizaje profundo para la detección automática de accidentes inesperados bajo malas condiciones de monitoreo de CCTV en túneles.

En la actualidad el campo de la seguridad vehicular y prevención de accidentes es muy importante, ya que anualmente existe cifras muy altas de tragedias que en una parte se podrían evitar, el algoritmo de aprendizaje profundo gracias a sus características se lo puede aplicar para la detección automática de accidentes, por ejemplo existe un prototipo que se encuentra implementado en un ambiente de condiciones malas de monitoreo (CCTV), es decir, dentro de túneles, el sistema de detección basado en una red neuronal convolucional Faster R-CNN y seguimiento de objetos (ODTS), la combinación de métodos es capaz de prevenir los siguientes casos: un manejo incorrecto, parar el vehículo, reconocer si una persona sale del vehículo dentro del túnel y por último aviso oportuno en caso de algún incendio, todo lo descrito se ejecuta en tiempo real. (Kyu Beom & Hyu Soung, 2019).

### 1.5.2. Algoritmo de seguimiento de objetivos basado en aprendizaje profundo y monitoreo de múltiples videos.

En los últimos años, la red de monitoreo urbano de China se ha desarrollado rápidamente y se ha vuelto cada vez más inteligente y de alta definición. Por lo cual, se necesitan tecnologías

informáticas para resolver algunos problemas existentes en el seguimiento de objetivos actualmente. Los algoritmos basados en aprendizaje profundo son la solución ya que se ha demostrado que se puede obtener muy buenos resultados bajo dicho método. Por lo tanto, se ha implementado el modelo de seguimiento de aprendizaje profundo MDNet (Red multidominio) que se combina con la red de detección de objetivos Faster R-CNN modificada con la finalidad de mejorar la robustez y precisión del multi método de reconocimiento de objetivos en el entorno dinámico. El algoritmo se compara con otro algoritmo de seguimiento CF2 que tiene como base el aprendizaje profundo en varios entornos, y los resultados indican que el algoritmo propuesto mejora la precisión del reconocimiento en tiempo real. (Yuncai, Pan, & Hongtao, 2018)

### 1.5.3. Sistema de conteo de personas basado en la detección de rostros

Los algoritmos de conteo no solo se pueden basar en la silueta o forma del objeto sino como es el siguiente caso como se observa en la Figura 3 se basa en el conteo de personas detectando su rostro, donde el número de personas que pasan por una puerta y son reconocidas mediante una cámara de video en tiempo real serán discriminadas si ingresan o no. La idea principal es utilizar primero la diferencia de cuadro para detectar los bordes ásperos de las personas que se encuentran en movimiento para posterior usar la función cromática para localizar la cara de las personas. Basado en el espacio de color NCC (Coordenadas de color normalizadas), la primera opción de cara inicial se obtiene detectando la región de color de piel y luego se reconoce la característica de cara del candidato para determinar si el candidato es cara real o no. Después de la detección de los rostros, se rastreará a una persona siguiendo el rostro detectado y luego contará si su rostro toca la línea de conteo. (Tsong-Yi, Chao-Ho, & Da-Jinn, 2010)


***Figura 3.*** Algoritmo de conteo de personas por detección de rostro.

Fuente: (Tsong-Yi, Chao-Ho, & Da-Jinn, 2010)

### 1.5.4. Tabla comparativa entre los diferentes algoritmos de detección y seguimiento frente a nuestro prototipo contador de personas.

**Tabla 1.**
*Tabla comparativa entre las diferentes aplicaciones basadas en Deep learning frente al prototipo contador de personas.*

| | **Deep Learning CCTV túneles.** (Kyu Beom & Hyu Soung, 2019) | **Deep Learning Múltiples videos.** (Yuncai, Pan, & Hongtao, 2018) | **Conteo bajo detección de rostros** (Tsong-Yi, Chao-Ho, & Da-Jinn, 2010) | **Prototipo del presente trabajo.** |
|---|---|---|---|---|
| ***Detección*** | ✓ | ✓ | ✓ | ✓ |
| ***Seguimiento*** | ✓ | ✓ | ✓ | ✓ |
| ***Iluminación*** | | ✓ | | ✓ |
| ***Múltiples entradas de videos*** | ✓ | ✓ | | |
| ***Detección de rostros*** | | | ✓ | ✓ |
| ***Detección de diferentes objetos.*** | | | | ✓ |

# CAPITULO 2

# 2. FUNDAMENTO TEÓRICO

## 2.1. Aprendizaje profundo (Deep Learning)

El aprendizaje tradicional de las maquinas su proceso se encuentra supervisado en la cual el programador tiene que especificar en decirle a la computadora que características debe tomar en cuenta para decidir que objeto se encuentra en una entrada de imagen. Este método es muy laborioso y de igual manera la tasa de éxito depende de las características del programador.

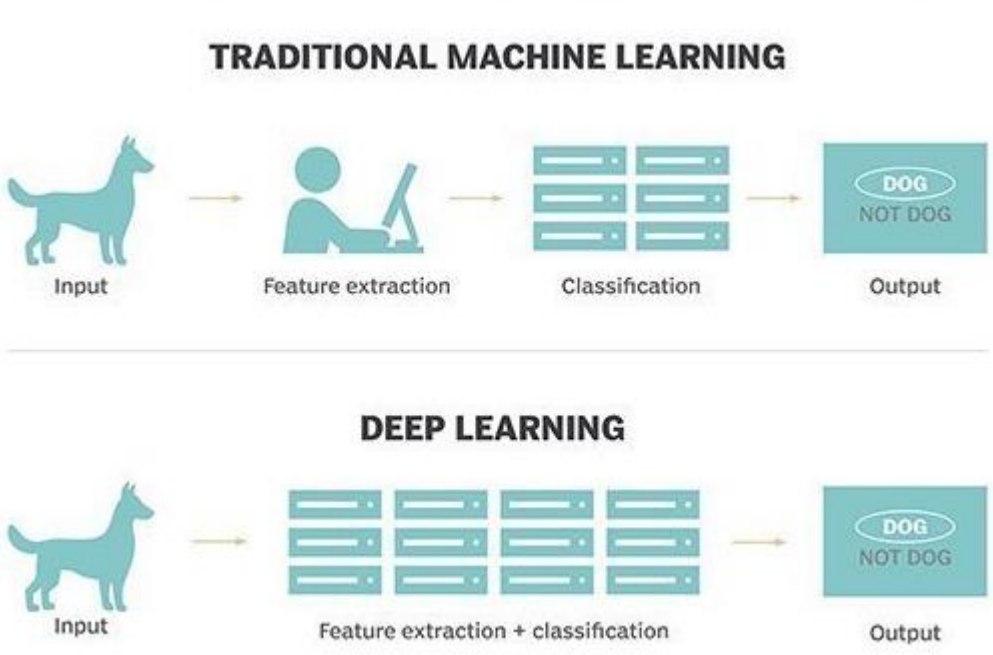



***Figura 4.*** Aprendizaje tradicional de máquinas vs Aprendizaje profundo.
Fuente: (Rouse, 2017)

El Deep Learning o aprendizaje profundo se le considera como un algoritmo automático bien estructurado que intenta emular el aprendizaje humano con el fin de obtener distintos conocimientos. Destaca porque no requiere de reglas programadas previamente, sino que el propio sistema es capaz de "aprender" por sí mismo para efectuar una tarea a través de una fase previa de entrenamiento. (SmartPanel, 2019).

Dicho algoritmo se compone de tres diferentes capas neuronales:

- Capa de entrada: Se compone por las neuronas que comprenden los datos de entrada que pueden ser imágenes o videos.
- Capa oculta: Se encarga de realizar el procesamiento y cálculos, en esta capa mientras más neuronas exista más complejo y costoso computacionalmente.
- Salida: Esta es la última red que se encarga de tomar la decisión del objeto a reconocer guiándose por su porcentaje de probabilidad obtenido.

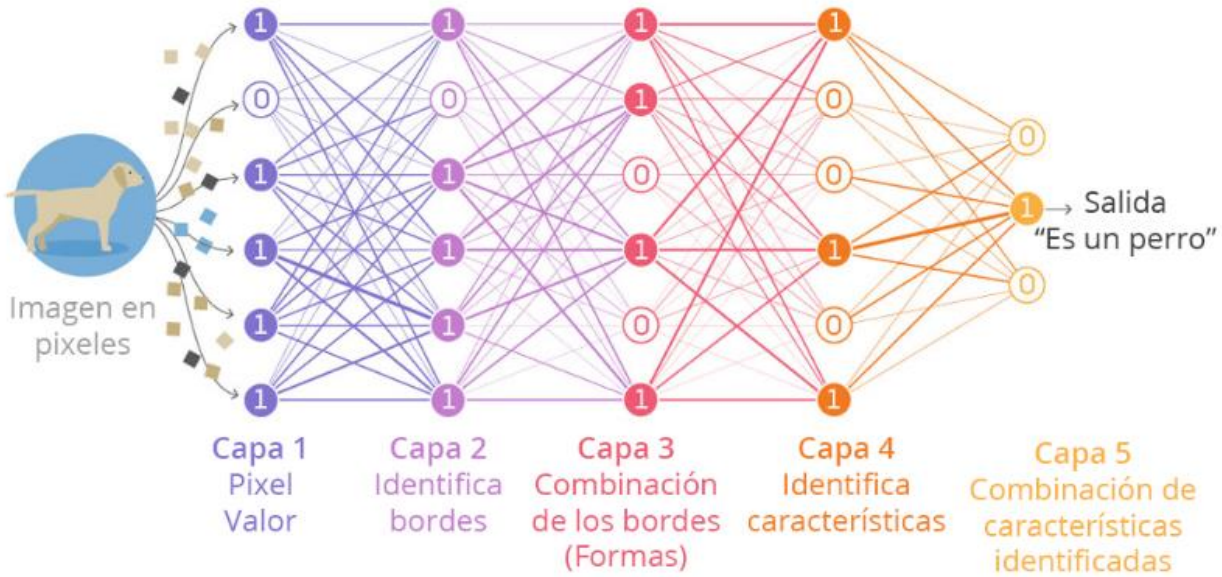



***Figura 5.*** Capas neuronales en el que se basa el algoritmo de aprendizaje profundo

Fuente: (Quantamagazine, 2019)

Hoy en la actualidad gracias a las diferentes fuentes de información donde cualquier persona la puede subir y se puede encontrar en el internet, dicha cantidad de datos beneficia en una manera muy positiva a este algoritmo ya que en eso se basa, en aprender de una base de datos muy amplia para así poder obtener las mejores probabilidades del objeto en diferentes ambientes por la razón de que no es lo mismo una entrada de video o una imagen cuando se encuentra con la mejor iluminación que es el día que comparar con la noche o condición de luz baja.

## 2.2. Algoritmos de detección de objetos

En la actualidad existen algoritmos de detección con distintas características que se usan en varias aplicaciones, a continuación, revisaremos los más conocidos y en los cuales se han hecho mejoras para un mejor desempeño.

### 2.2.1. R-CNN

Dicho algoritmo ayuda a mejorar el inconveniente que existía en un comienzo con los primeros programas de aprendizaje profundo, por la razón de tomar gran cantidad de regiones de interés en la imagen y con esto llevar a un gran costo computacional, para aquello R-CNN, se enfoca en seleccionar una gran cantidad de regiones, Ross Girshick propuso un método donde se utilizara en la búsqueda selectiva extraer la cantidad de 2000 regiones de la imagen de entrada y las nombro

propuestas de región. Por lo tanto, en lugar de tratar de clasificar una cantidad grande de regiones, ahora se limita a trabajar con 2000 regiones.

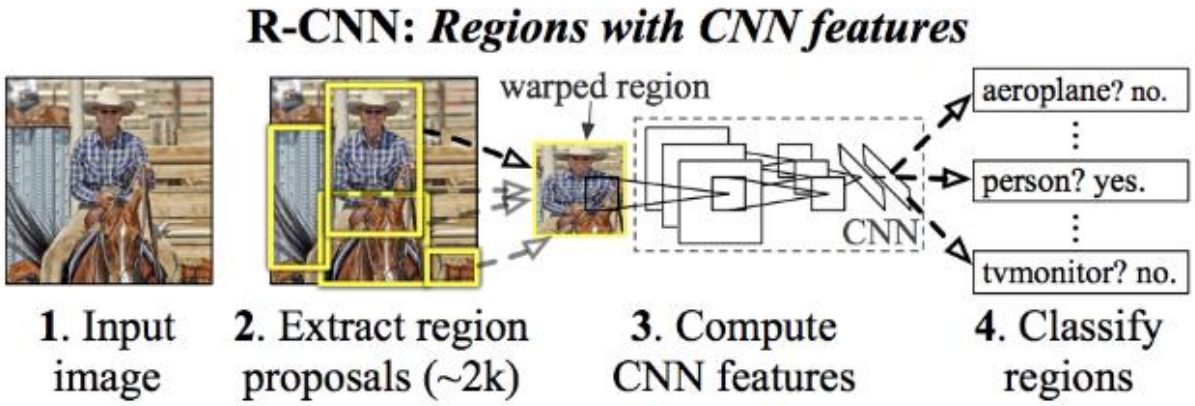



***Figura 6.*** Algoritmo de detección R-CNN

Fuente: (Khandelwal, 2019)

Esta solución de las 2000 regiones extraídas se combina en un cuadrado para posteriormente alimentar a una red neuronal convolucional que genera un vector de características de 4096 dimensiones de resultado. El CNN trabaja como un recolector de características y la capa densa de salida ya contiene de las características extraídas de la imagen de entrada, luego se introducen en un SVM para identificar el objeto dentro de la propuesta elegida.

Es importante mencionar que el problema que tiene este algoritmo es el tiempo que lleva realizar todo el proceso, a pesar de haber solucionado en cierta manera un problema de igual forma no se puede utilizar este programa para una aplicación en tiempo real ya que toma alrededor de 47 segundos el proceso de una sola imagen.

### 2.2.2. Algoritmo Fast R-CNN

Como en el caso anteriormente dicho el autor Ross Girshick dio la solución a algunos de los problemas de R-CNN, construyendo un algoritmo mejorado el cual lo llamo Fast R-CNN. Es similar a R-CNN, pero en lugar de generar el paso a las propuestas de la región a la CNN, alimentamos la imagen de entrada a la CNN con la finalidad de generar un mapa de características convolucional. A partir de dicho mapa, se puede lograr reconocer la región de las propuestas y se procede en deformar en cuadrados y, al usar una capa de agrupación RoI, las rediseñamos en un valor fijo para que se pueda proporcionar a una capa completamente conectada. Desde el vector de características RoI, se utiliza una capa softmax para identificar la clase de la región propuesta y adicional los valores de desplazamiento para el cuadro delimitador. (Gandhi, 2018)

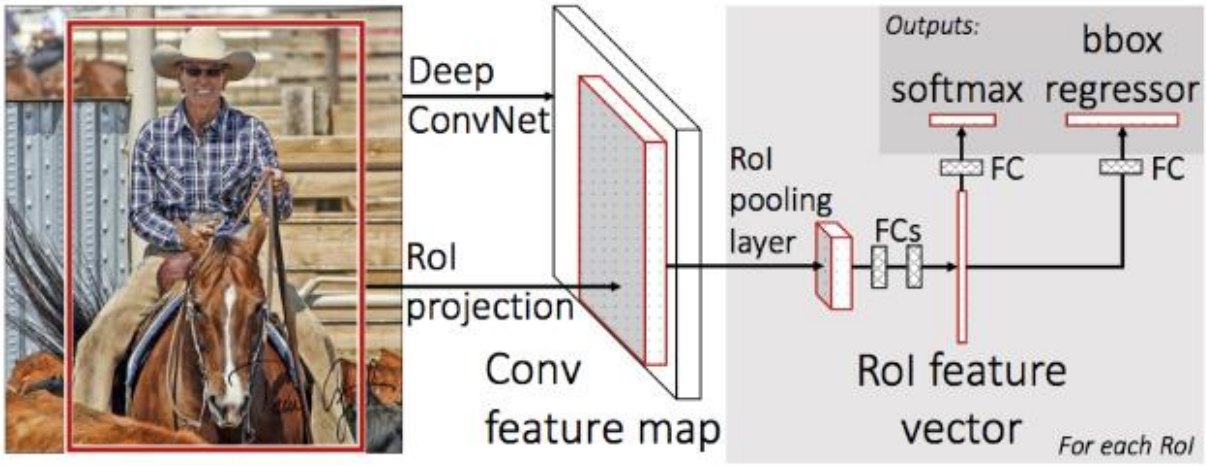



***Figura 7.*** Algoritmo de detección Fast R-CNN

Fuente: (Khandelwal, 2019)

Básicamente Fast R-CNN es más rápido que R-CNN por la razón de que ya no existe las 2000 regiones que alimentaban a las capas, en cambio el mapa de características se genera una vez amenorando el tiempo de ejecución por imagen como se puede observar en la Figura 8.

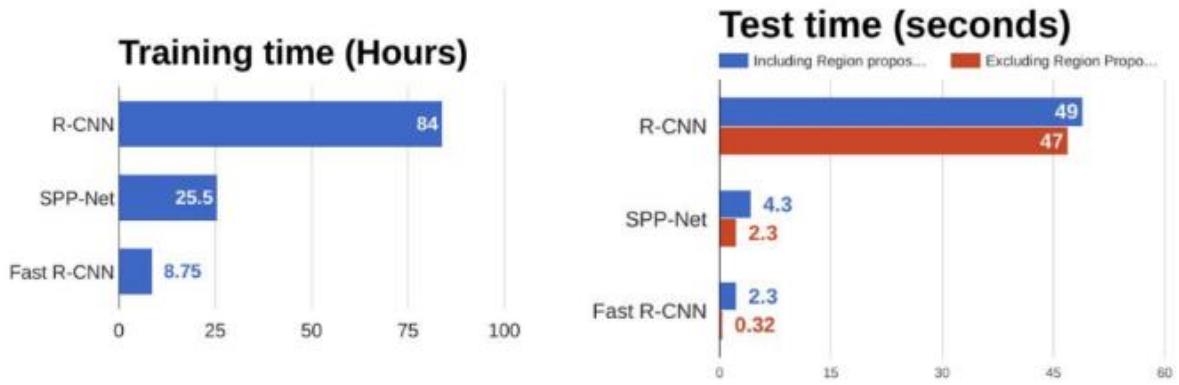



***Figura 8.*** Comparación en tiempos en la ejecución de los algoritmos de detección.
Fuente: (Khandelwal, 2019)

### 2.2.3. Algoritmo Faster R-CNN

En este caso el autor Shaoqing Ren mejoro el algoritmo con la finalidad de que sea mucho más rápido que los anteriores programas (R-CNN y Fast R-CNN), obteniendo como resultado un tiempo por imagen de 10 ms que a comparación de Fast R-CNN le toma 2 segundos. Dicho algoritmo cambia las propuestas de región de búsqueda selectiva a la idea de usar otra red independiente convolucional (RPN) la cual básicamente se encarga de predecir las propuestas de región o propuestas de objeto en la imagen, seleccionado el cuadro delimitador se vuelven a dar forma usando la capa de agrupación RoI, finalmente se clasifica y se obtiene el resultado del objeto entrenado.

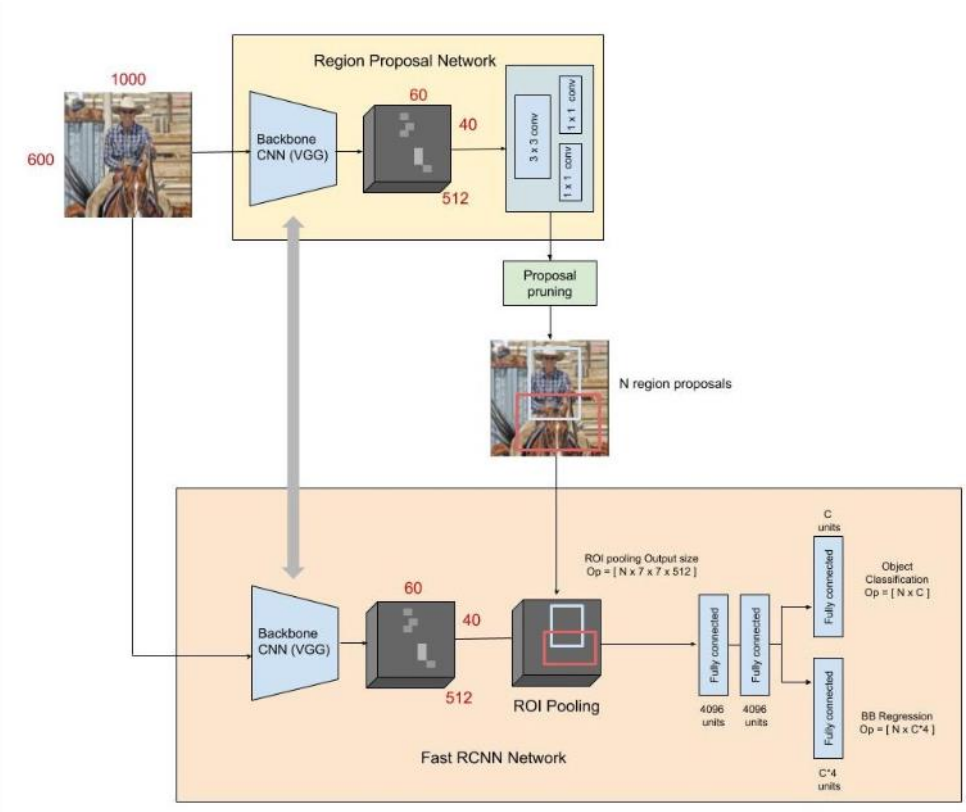



***Figura 9.*** Arquitectura del algoritmo Faster R-CNN

Fuente: (Khandelwal, 2019)

Tiempo de respuesta en comparación con los otros algoritmos.

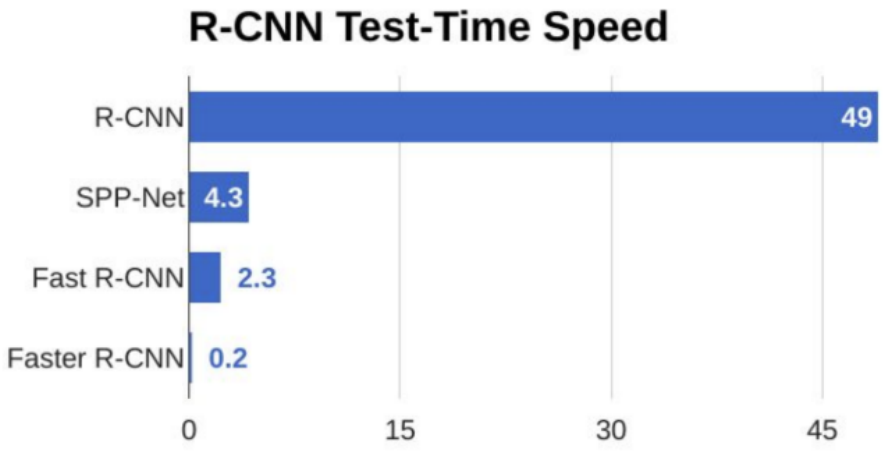



***Figura 10.*** Comparación de velocidad de los algoritmos de detección.

Fuente: (Khandelwal, 2019)

### 2.2.4. Algoritmo de detección YOLO

Los algoritmos de detección mencionados anteriormente se basan en usar regiones para detectar el objeto en la imagen de entrada, el problema que se genera es que la red que se esté utilizando no observa la imagen completa, el algoritmo de YOLO es bastante diferente ya que se utiliza una sola red convolucional que se encarga de predecir los cuadros delimitadores y las probabilidades de objeto para cada cuadro.

El funcionamiento de YOLO se basa en tomar la imagen de entrada y dividirla completamente en una cuadricula de SxS, en cada una de las cuadriculas se toma m cuadros delimitadores. Para cada cuadro elegido la red realiza una probabilidad de valores, si los cuadros pasan por encima de un valor de umbral definido se le selecciona y se utiliza para ubicar el objeto que se encuentra en la imagen.

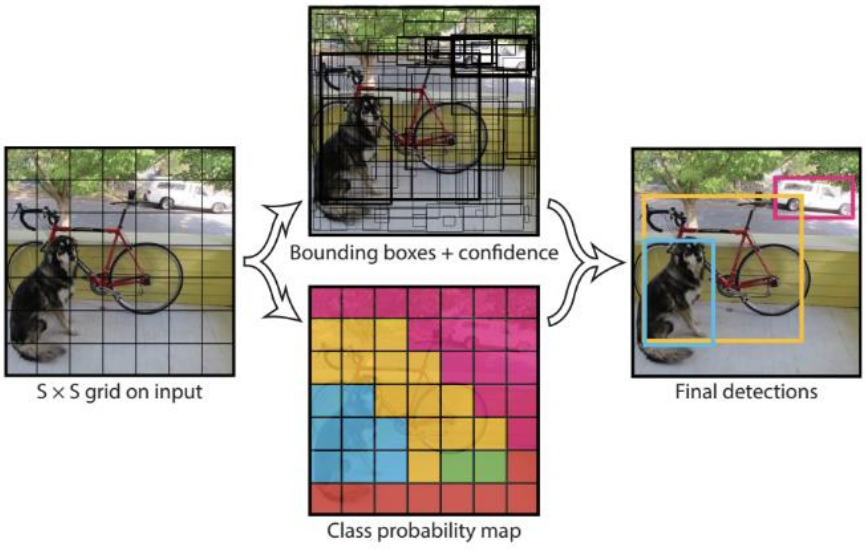



***Figura 11.*** Arquitectura del algoritmo de detección YOLO

Fuente: (Khandelwal, 2019)

Con este método que trabaja YOLO se obtiene como resultado que es súper rápido pero en cuestión de precisión baja un poco como se puede observar en la Figura 11, con la rapidez que obtiene el algoritmo el contra que ocurre es que en una imagen de entrada objetos muy pequeños no puedan ser detectados.


***Figura 12.*** Comparación de los algoritmos de detección en Precisión y velocidad.
Fuente: (CV-TRICKS, 2017)

### 2.2.5. Detector de disparo único (SSD)

Este detector su principal característica es su buen equilibrio entre velocidad y precisión, SSD funciona ejecutando una red convolucional en la imagen de entrada una sola vez en la cual calcula el mapa de características, posterior se ejecuta un pequeño núcleo convolucional de 3x3 en dicho mapa para lograr predecir los cuadros delimitadores y la probabilidad de clasificación. SSD obtiene

predicción después de múltiples capas convolucionales con cada una de ellas tiene la ventaja de poder detectar objetos en diferentes escalas.

Para el desarrollo de este trabajo se utiliza el algoritmo SSD por las ventajas que posee y la aplicación donde se le está ejecutando, nos permite juntar otro algoritmo el cual se trata del tracking del objeto, de igual manera nos permite trabajar en tiempo real a través de un flujo de video bajo un costo computacional aceptable.

## 2.3. Algoritmos de seguimiento de objetos

El seguimiento o tracking se basa principalmente en ubicar un objeto específico dentro de una cadena de imágenes, a continuación, se detallará los diferentes algoritmos que cubren este objetivo:

- Densidad de Flujo Óptico: Estos algoritmos determinan el vector de dirección de un pixel seleccionado dentro de la imagen de entrada.
- Caudal de Flujo Óptico: Estos algoritmos ayudan a realizar el seguimiento de los puntos característicos de una imagen.
- Filtros de Kalman: Este es un algoritmo de visión muy común que nos beneficia en determinar la posición de un objeto recogiendo la información de movimiento previa.
- Localizadores de único objeto: Dichos algoritmos encierran al objeto de interés dentro de un cuadro para posterior seguimiento en las imágenes de secuencia que aparecen.
- Localizadores de varios objetos: Este algoritmo funciona en secuencia del detector de objetos, el cual se requiere que sea rápido, ya que al reconocer un objeto el algoritmo

encierra en un cuadro a cada uno identificándolos con distintas variables y posterior se genera el seguimiento en la secuencia de imágenes.

### 2.3.1. TLD Tracker

Seguimiento, Aprendizaje y Detección (Tracking, Learning and Detection). En este algoritmo, el proceso de seguimiento se separa en tres etapas principales: seguimiento del objeto a corto plazo, aprendizaje y detección. La salida se muestra a saltar entre objetos de las mismas características. Por ejemplo, en el seguimiento de un objeto (persona) en una imagen de entrada, si llega a aparecer una segunda, se ubicará de forma temporal a la última. Esto en el algoritmo genera un alto valor de falsos positivos lo cual es una de las desventajas considerables. (Jovann, 2019)

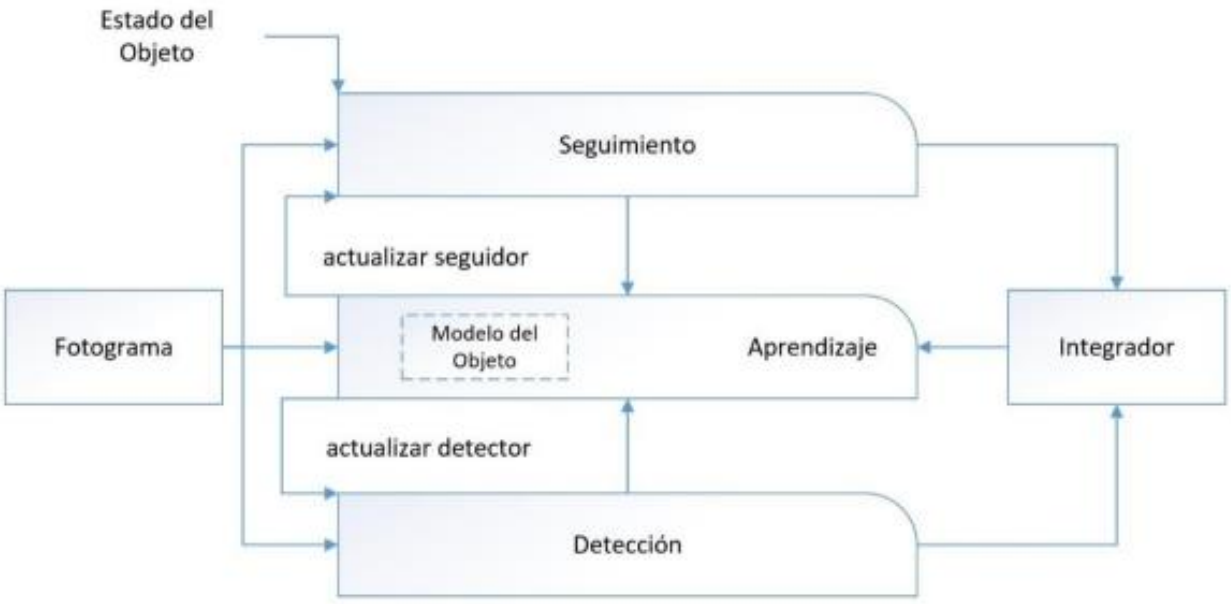



***Figura 13.*** Esquema de operación de TLD

Fuente: (Wen, Wu, & Li, 2015)

### 2.3.2. Algoritmo de seguimiento de Centroide

Este algoritmo trabaja en conjunto con la detección de objetos tomando en cuenta una o más entradas de objetos detectados, al aparecer un objeto en la imagen se crea una identificación única "ID" que posteriormente nos ayudara a rastrear a medida que se mueven alrededor de los cuadros en un video. Dicho algoritmo de seguimiento se basa en el Centroide por la razón de que encierra al objeto detectado en un cuadro delimitador y define su centroide, luego se ayuda del cálculo de la distancia euclidiana entre los objetos detectados con la finalidad de poder reconocer y diferenciar si se trata del mismo objeto o es uno nuevo, finalmente comienza con el seguimiento en el marco del video.


***Figura 14.*** Funcionamiento algoritmo de seguimiento de centroide.
Fuente: (Rosebrock, 2018)

## 2.4. Algoritmos de Visión para conteo de objetos

Estos algoritmos enfocados a la aplicación de conteo de objetos se basan en la sustracción de fondo, esto se debe por los inconvenientes de condiciones de iluminación o condición propia del

ambiente a evaluar, utilizar este procedimiento conlleva establecer una normalización de la captura de imágenes en el entorno operativo. (Jovann, 2019)

***Figura 15.*** Proceso de ejecución en el conteo de objetos.
Fuente: (Da, Bodong, & Weigang, 2014)

### 2.4.1. Algoritmo de conteo de personas basado en el reconocimiento de rostros.

Los algoritmos de conteo de personas también se pueden basar en el reconocimiento de rostros junto con el tracking, la finalidad consiste en reconocer y contabilizar las personas que ingresan a cierto escenario controlado. Su principal aporte es la revisión matemática que se realiza para poder realizar la extracción de características en las imágenes, existe diferentes técnicas en la detección de rostros como se puede mencionar: Eigenfaces o discriminantes lineales FLD, dichos algoritmos pre entrenados ayudan en la detección e identificación de rostros.

### 2.4.2. Contador de alta demanda de personas usando herramientas morfológicas.

Este algoritmo contador de personas aborda el problema de determinar la cantidad de gente que entra y sale en un vagón de tren en hora pico, de igual manera se evalúa en diferentes

cambios de fondo y/o la iluminación. Este método utiliza una técnica de marcado de pixeles para marcar sobre unos vectores entre imágenes que se encarga de decidir hacía que dirección va el objeto. El cruce de la línea de frontera para la zona de conteo nos ayuda en la entrada y salida de las personas en este caso realiza el conteo. (Albiol, Mora, & Naranjo, 2001)

# CAPITULO 3

# 3. METODOLOGÍA

## 3.1. Descripción del modelo

El prototipo contador de personas se puede apreciar su estructura general en la Figura 16, en la cual interviene la entrada de video que es la Cámara, el entrelazo entre los algoritmos tanto de detección como de seguimiento, como salida para el operario el resultado en un monitor que indica los resultados obtenidos.

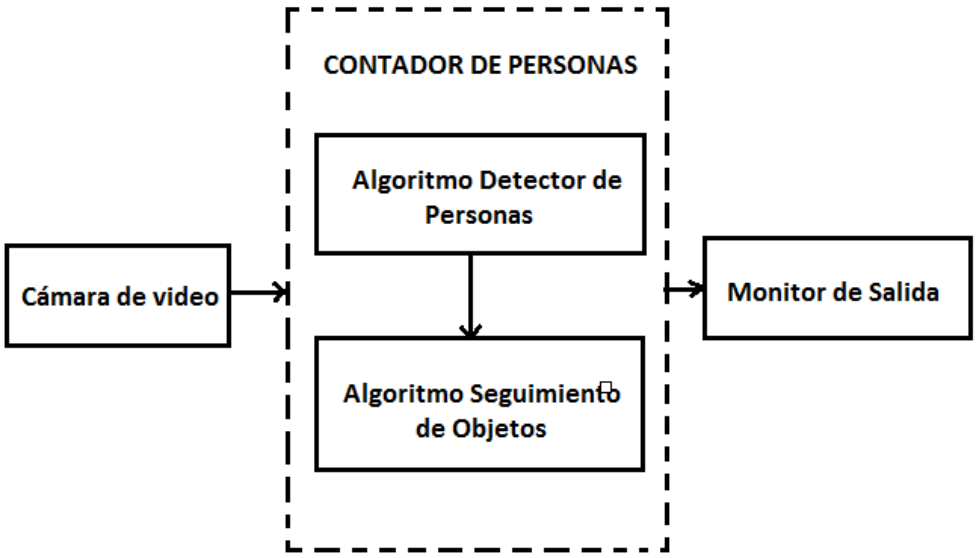
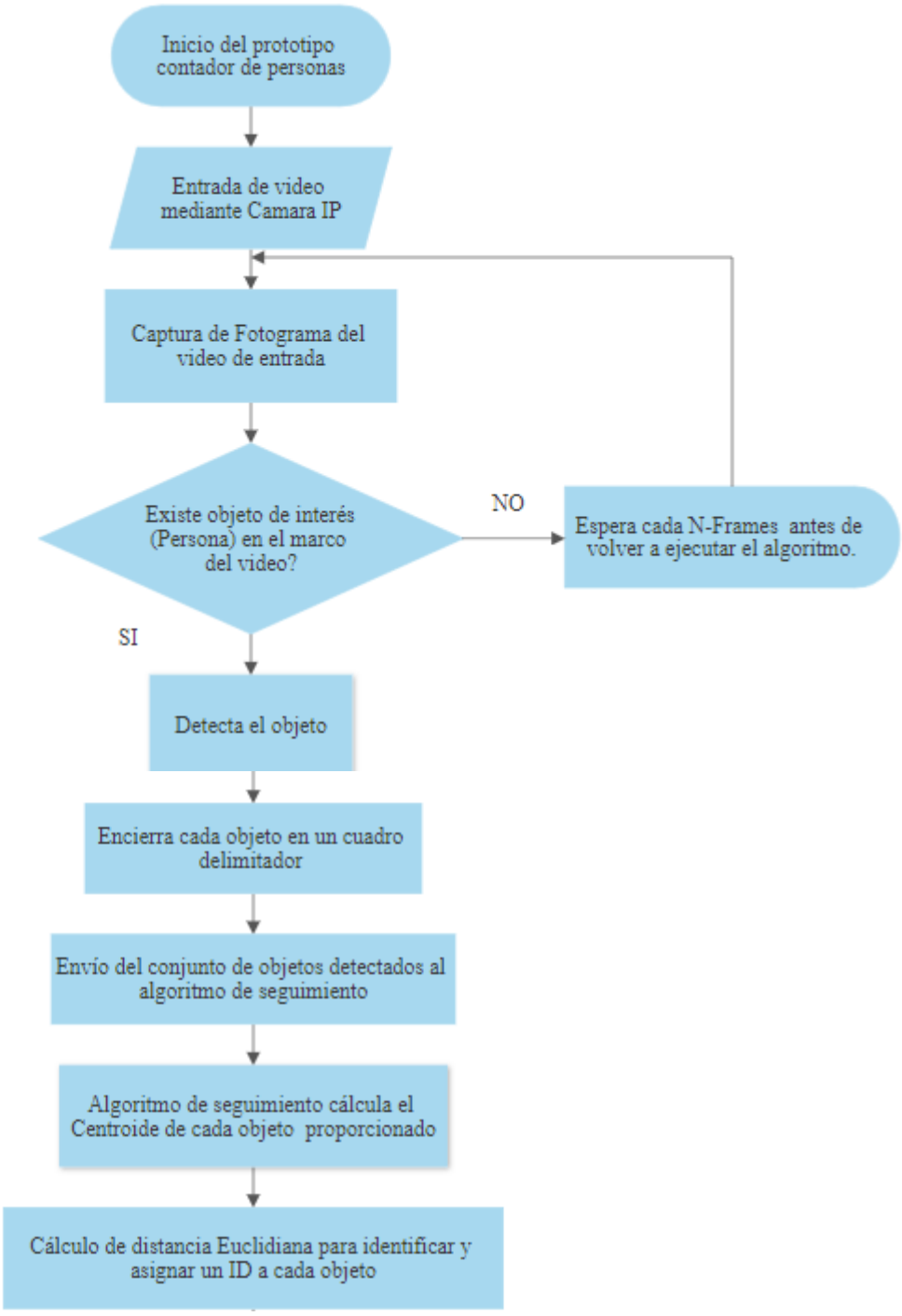



***Figura 16.*** Estructura del prototipo contador de personas.

En la Figura 17 se muestra un diagrama de flujo con el funcionamiento del prototipo contador de personas que se implementa en este trabajo.


Inicio del prototipo contador de personas
Entrada de video mediante Camara IP
Captura de Fotograma del video de entrada
Existe objeto de interés (Persona) en el marco del video?
NO
Espera cada N-Frames antes de volver a ejecutar el algoritmo.
SI
Detecta el objeto
Encierra cada objeto en un cuadro delimitador
Envío del conjunto de objetos detectados al algoritmo de seguimiento
Algoritmo de seguimiento cálcula el Centroide de cada objeto proporcionado
Cálculo de distancia Euclidiana para identificar y asignar un ID a cada objeto

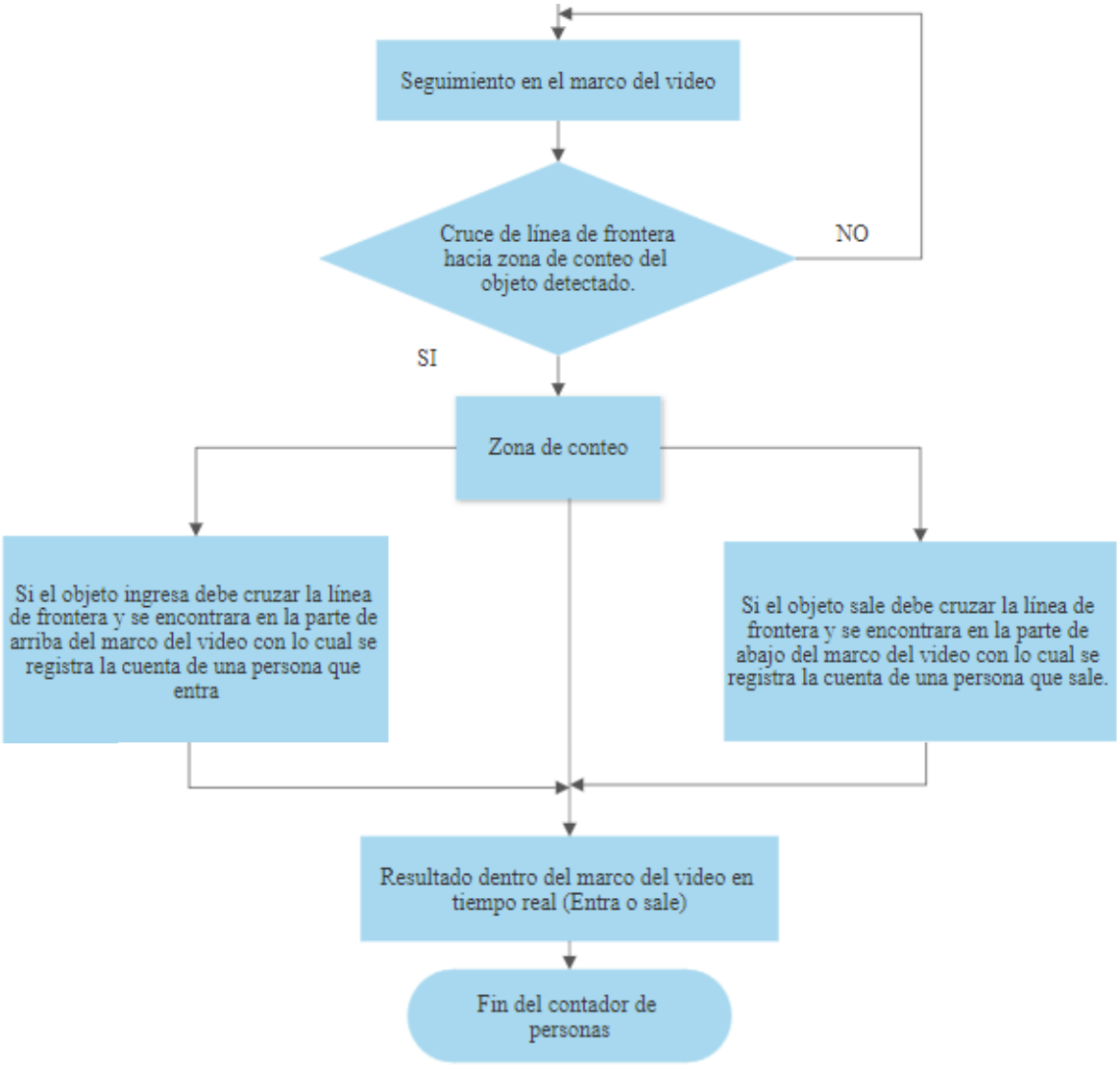



***Figura 17.*** Diagrama de flujo del prototipo contador de personas

El prototipo se ejecuta en un dispositivo especializado de Nvidia para la ejecución de algoritmos bajo redes neuronales convolucionales, como entrada de video se utiliza una cámara independiente IP.

Al ejecutar el algoritmo en la tarjeta Jetson Tx1 de Nvidia, comienza con el funcionamiento de la cámara como entrada de video, posterior inicia la detección de objetos en este caso personas

cada N-frames, es decir, cada cierto número de fotogramas del video el algoritmo SSD realiza la detección en caso que exista una persona en el marco del video, identificado el objeto el detector encierra a cada uno en un cuadro delimitador para que pueda entrar en funcionamiento el algoritmo de seguimiento, recibido el conjunto de objetos proporcionados por la detección, el seguimiento realiza su primer paso que consiste en calcular el centroide en cada cuadro delimitador, con la coordenada en el marco del video que muestra el centroide de cada objeto se procede a realizar el cálculo de distancia euclidiana que básicamente se encarga de asignar un ID a cada objeto o poder reconocer si se trata de uno nuevo, con la asignación e identificación de cada objeto se inicia propiamente el tracking en la secuencia de fotogramas, para la discriminación en la que se reconoce si una persona ingresa o sale del lugar el algoritmo establece una línea de frontera en la cual cada persona según su acción entra a la zona de conteo que puede ser en la parte de arriba o abajo reconociendo y contabilizando la entrada o salida de la persona.

## 3.2. Diagrama de componentes del prototipo

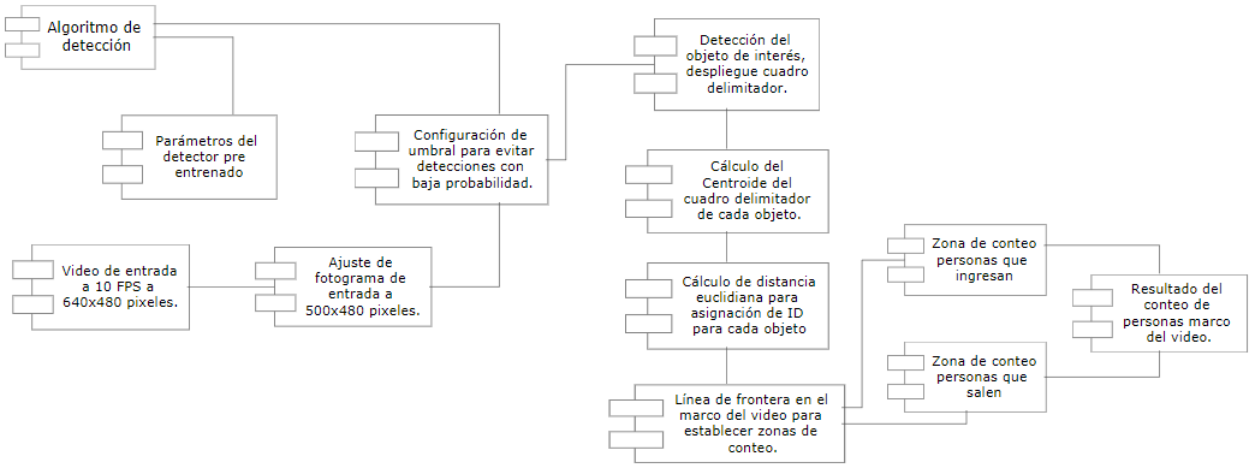



***Figura 18.*** Diagrama de componentes del prototipo contador de personas

- **Algoritmo de detección:** El algoritmo ocupado en el presente trabajo es el detector pre entrenado de disparo único SSD.
- **Video de entrada:** Para el video de entrada se ocupará una cámara independiente IP en la cual se necesita configurar diferentes parámetros como la resolución, flujo de datos y fotogramas por segundo.
- **Parámetros del detector:** Para hacer uso del detector SSD se tiene que establecer diferentes argumentos que hacen referencia al direccionamiento al modelo pre entrenado de Caffe, cantidad de frames para ejecutar el algoritmo, dirección del archivo de entrada y salida del video.
- **Configuración de probabilidad:** Valor de umbral mínimo establecido en el algoritmo de detección para discriminar detecciones débiles.
- **Cuadro delimitador:** Una vez que el detector entra en funcionamiento cada *N*-frames realiza un proceso en el cual encierra al objeto de interés detectado en un cuadro delimitador dentro del marco del video.
- **Cálculo del Centroide:** Este cálculo ya corresponde al algoritmo de seguimiento, consiste en recibir el conjunto de cuadros delimitadores de cada objeto proporcionado por el detector para determinar su Centroide y establecer las coordenadas dentro del marco de video.
- **Cálculo de distancia euclidiana:** Este cálculo se realiza para la identificación de nuevos o existentes objetos y de igual manera designa un ID a cada uno.
- **Línea de frontera:** La línea de frontera se encuentra en la mitad del marco del video, dicha línea ayuda al algoritmo de seguimiento para poder establecer zonas de conteo en

las cuales depende la dirección que tome el objeto será registrado si entra o sale del lugar evaluado.

- **Resultado:** Cada cuenta que registra el algoritmo contador de personas es representado en el marco del video diferenciando las personas que entraron y las que salieron.

### 3.3. Algoritmo de detección de Disparo único (SSD) de MobilNet

Este método de detección utiliza una sola red neuronal, es semejante al algoritmo YOLO ya que como su nombre lo dice toma un solo disparo para lograr detectar diferentes objetos en la imagen de entrada. Debido a dicha característica el método es rápido y tiene un equilibrio con la precisión en la detección, a continuación, se indicará la velocidad y precisión de los algoritmos más utilizados.

- SSD 300: Resulta con 59 FPS con mAP 74.3%
- SSD 500: Resulta con 22 FPS con mAP 76.9%
- Faster R-CNN: Resulta con 7 FPS con mAP 73.2%
- YOLO: Resulta con 45 FPS con mAP 63.4%

Dichos resultados obtenidos con este método se deben principalmente a eliminar las “propuestas de región” como utiliza R-CNN, adicional cuenta con un filtro convolucional progresivamente decreciente aplicado a varios mapas de características con la finalidad de predecir categorías de objetos y compensaciones en ubicaciones de cuadro delimitador, es decir a múltiples escalas.

### 3.3.1. Arquitectura del método de detección SSD

Este algoritmo cuenta con su principal red que es VGG-16 junto con capas de convolución multibox. La red principal se encarga de extraer las características de la imagen de entrada, las demás capas que le siguen se encargan de la detección y lograr reconocer en diferentes escalas.

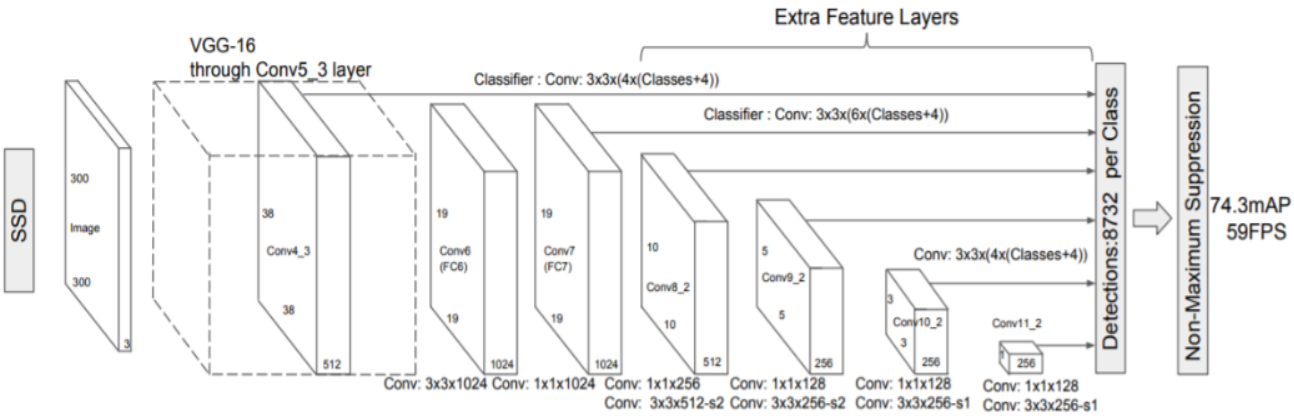



***Figura 19.*** Arquitectura algoritmo de detección SSD.
Fuente: (Liu, y otros, 2016)

La reducción progresiva de las capas convolucionales beneficia en la profundidad de análisis de la imagen cubriendo objetos de diferentes tamaños.

### 3.3.2. Detector MultiBox

En el proceso de entrenamiento se definen varios cuadros predeterminados que se comparan entre sí con la finalidad de coincidir valores de ubicación, escala y aspectos, se selecciona los cuadros con mayor superposición ya que indica que en ese marco se encuentra el objeto detectado.

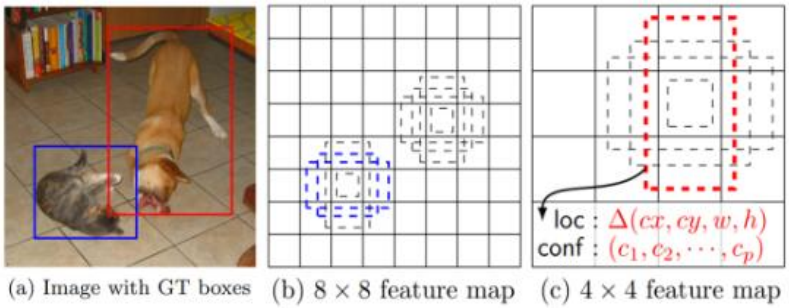



***Figura 20.*** Selección de cuadros a diferentes escalas, ubicaciones en varios mapas de características.

Fuente: (Liu, et al., 2016)

En la Figura 20 se puede observar dos cuadros delimitadores a diferentes escalas. En el cuadro de la izquierda se indica el reconocimiento de un gato mientras que el de la derecha indica a un perro. Después de definir diferentes cuadros delimitadores el cuadro con mayor definición de características del objeto se lo denomina cuadro delimitador positivo y el resto se tratan como negativos. Las predicciones para los cuadros delimitadores se conforman de:

- Valor de confianza para todas las categorías o clases de objetos detectados, la clase 0 se encuentra reservada para indicar la ausencia del objeto.
- Cada cuadro delimitador contiene, desplazamientos desde el centro del cuadro predeterminado, altura y ancho.

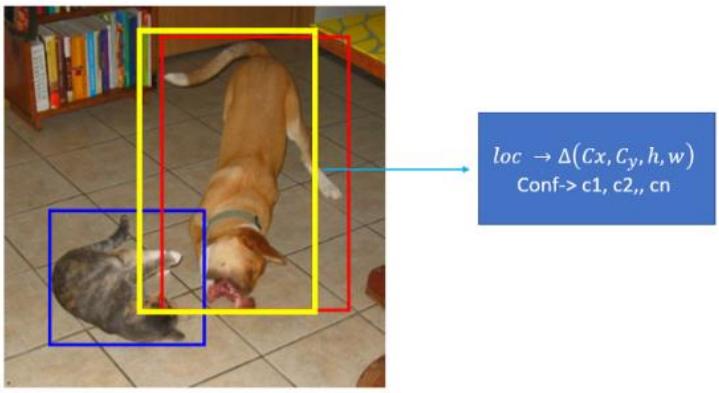



***Figura 21.*** Cuadro delimitador y confianza que contiene cada objeto detectado.

Fuente: (Khandelwal, 2019)

### 3.3.3. Entrenamiento del método de detección SSD.

La capa principal VGG-16 es la red base que realiza la extracción de características. Las capas que le siguen evalúan cajas de diferentes relaciones de aspecto en cada ubicación en varios mapas de características con diferentes escalas. Las Multibox son como anclas similares al algoritmo Fast R-CNN. Se obtiene múltiples cuadros predeterminados de diferentes tamaños, relación de aspecto en toda la imagen como se indica en la Figura 22. SSD utiliza 8732 cajas, esto beneficia en encontrar el cuadro predeterminado que más se superpone con el cuadro delimitador que contiene el objeto. (Khandelwal, 2019)

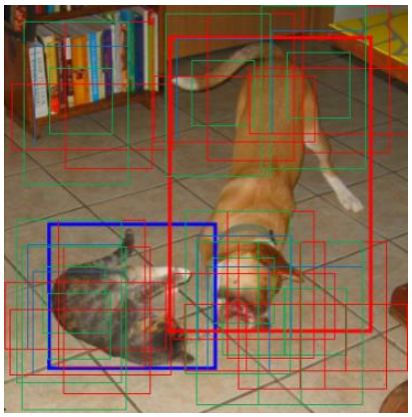


***Figura 22.*** Proceso de entrenamiento SSD.
Fuente: (Khandelwal, 2019)

## 3.4. Algoritmo de seguimiento de Centroide

Dicho algoritmo debe trabajar en conjunto con la detección, para cada objeto detectado se genere el rastreador y se pueda identificar la acción que realiza a medida que se mueve alrededor del marco del video.

El primer paso que realiza este algoritmo después de que se detectó algún objeto consiste en recibir un conjunto de cuadros delimitadores (objetos detectados), luego en cada uno se calcula su centro o centroide, esto se lo puede observar en la Figura 23 que existen 2 objetos con sus cuadros delimitadores.

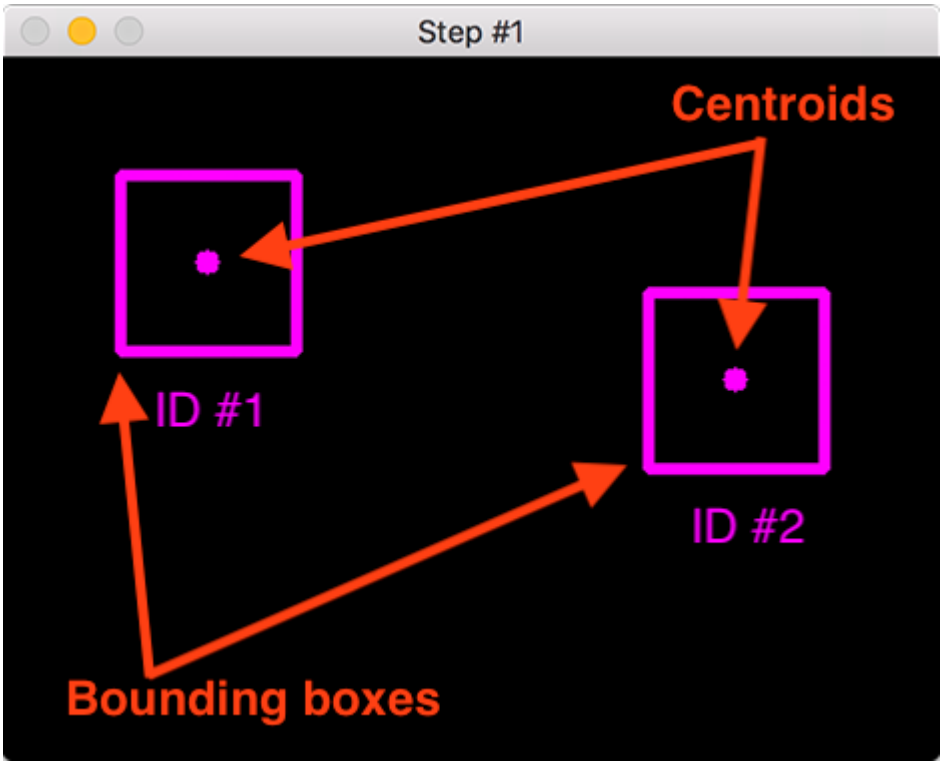



***Figura 23.*** Cálculo del centroide en cada cuadro delimitador del objeto detectado.
Fuente: (Rosebrock, 2018)

Como segundo paso se realiza el cálculo de distancia euclidiana entre los centroides nuevos (amarillo) y centroide existente (purpura) que se indica en la Figura 24, dicho calculo ayuda a identificar cuando existe un nuevo objeto o por otro lado que no se trate del mismo, mientras no sobrepase una distancia establecida se reconoce como el mismo objeto, gracias a esto evitamos falsos positivos y reduce el error que se puede generar en el conteo de personas.

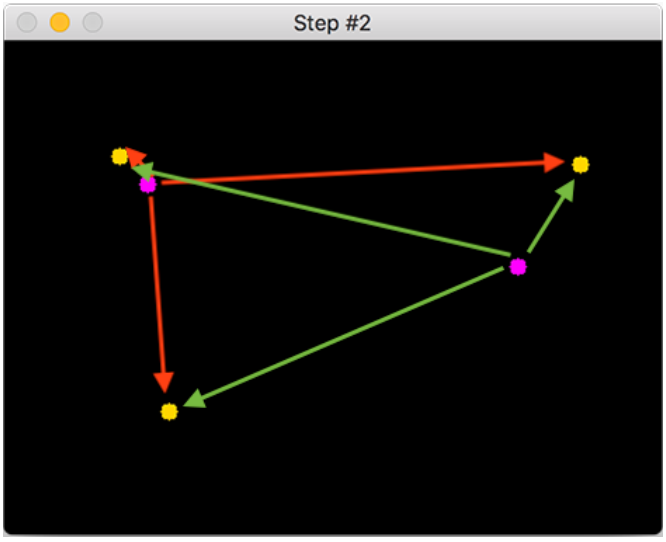



***Figura 24.*** Cálculo de distancia euclidiana entre centroides nuevos y existentes.

Fuente: (Rosebrock, 2018)

Realizado el cálculo de distancia se procede a asociar los ID de cada objeto, esto se puede observar en la Figura 25 que el punto de amarrillo que se encuentra con incógnita hace referencia a un nuevo objeto, mientras que los otros dos como no superan la distancia establecida vienen a ser los mismos objetos existentes que se movieron un poco en el marco.

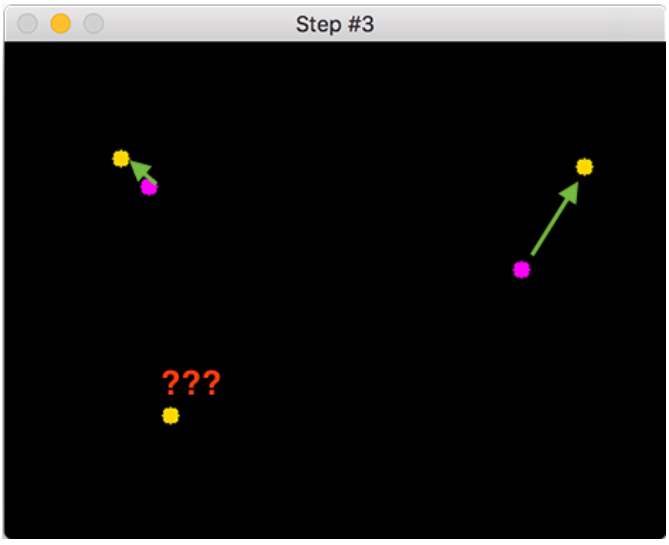



***Figura 25.*** Asociación de ID a nuevos o existentes objetos.

Fuente: (Rosebrock, 2018)

En la Figura 26 se observa la asignación de ID a cada objeto siendo así 2 objetos existentes y uno nuevo que apareció en el marco.

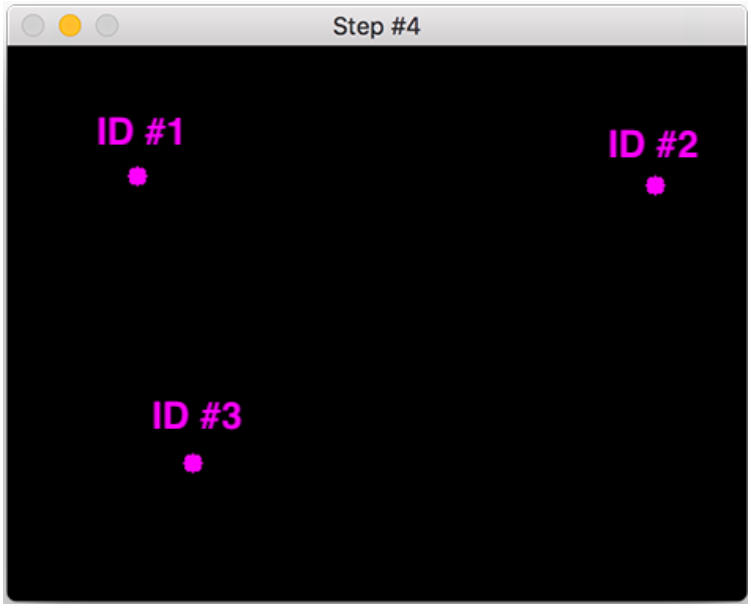



***Figura 26.*** Asignación de ID a cada objeto.
Fuente: (Rosebrock, 2018)

### 3.5. Hardware Kit de desarrollo Nvidia Jetson Tx1

Las aplicaciones que conllevan la utilización de los algoritmos basados en redes neuronales profundas requieren de un hardware que soporte dicho procesamiento, para aquello Nvidia ha desarrollado módulos que permiten procesar los complejos cálculos con una potencia de 1 Teraflop como es el caso del módulo Jetson Tx1 y el cual en este trabajo va a ser utilizado.

Este módulo emplea la plataforma de desarrollo CUDA de Nvidia para computación heterogénea optimizada para la programación de aplicaciones que se beneficien de la paralelización de los cálculos mediante el uso de la GPU. (Arenas, 2015)

El módulo Jetson Tx1 está compuesto por un procesador ARM A57 con capacidad para 64 bits y un procesador gráfico Maxwell con 256 núcleos que se unen a 4GB de memoria RAM LPDDR y 16 GB de almacenamiento interno usando memorita eMMC, tarjeta WiFi, tarjeta de red Gigabit Ethernet y Bluetooth 4.1.


***Figura 27.*** Kit de desarrollo Nvidia Jetson Tx1
Fuente: (Amazon, 2016)

### 3.5.1. Instalador JetPack de Nvidia

Nvidia JetPack SDK es un instalador que nos permite actualizar a nuestro kit de desarrollo Jetson Tx1 con la última versión del sistema operativo (Ubuntu 18.04), adicional nos ayuda a instalar herramientas de desarrollador, bibliotecas y API.

### 3.5.2. Instalador Nvidia SDK Manager

Este instalador nos ayuda inmensamente en la configuración que requieren los diferentes módulos (SDK DRIVE y Jetson de Nvidia), proporciona una solución en el entorno de desarrollo

tanto para la máquina que está trabajando como host y el dispositivo de destino que en este caso es el módulo Jetson Tx1.

Es importante destacar las siguientes características del instalador:

- Instalador mucho más rápido, permite el acceso a las descargas rápidamente y optimiza la configuración del entorno de desarrollo.
- Permite una administración de paquetes y dependencias con la finalidad de incompatibilidades de versión en el módulo a instalar.
- Permite instalar los paquetes en la maquina host y flashea el módulo de Nvidia conectado.
- En este entorno engloba los múltiples SDK y herramientas que se requiere para el desarrollo.

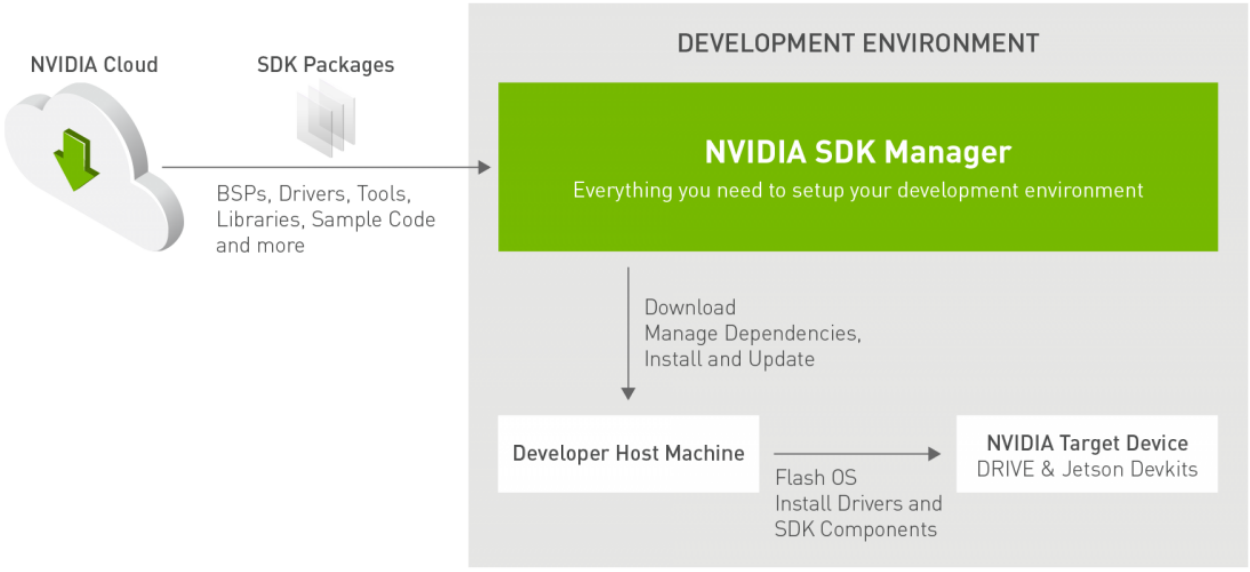



***Figura 28.*** Esquema de ejecución del Instalador SDK Manager de Nvidia.

Fuente: (Nvidia, 2019)

En el presente trabajo se utilizó este instalador beneficiando en la descarga e instalación de las siguientes herramientas:

- Sistema Operativo Ubuntu versión 18.04
- JetPack v4.2.3
- OpenCV v3.3.1 (Incluye los codecs de video)
- CUDA v10
- Python versión 3

## 3.6. Herramienta Python

Python en un lenguaje de programación bastante versátil multiplataforma y multiparadigma, permite crear sitios o aplicaciones en diferentes sistemas operativos como iOS, Android, Windows o Mac. El código se destaca por ser legible y tener la licencia de código abierto, ayuda en la automatización de procesos logrando así poder trabajar con grandes volúmenes de datos por su extracción y procesamiento. Por las razones anteriormente mencionadas este lenguaje es el ideal para el desarrollo de algoritmos basados en aprendizaje profundo.

### 3.6.1. Biblioteca dlib

Esta biblioteca es multipropósito que puede ser utilizada en C++ o Python siendo de código abierto, dispone de varios algoritmos de aprendizaje de máquinas, análisis de imágenes, métodos de rango para clasificación y regresión, SVM multiclase, etc. En este trabajo dicha biblioteca se utiliza para la implementación de filtros de correlación en el seguimiento del objeto.

### 3.6.2. Biblioteca Numpy

Numpy pertenece a una extensión de Python, constituye básicamente una biblioteca para funciones matemáticas de alto nivel agregando soporte para vectores y matrices.

En este trabajo se requiere instalar de esta biblioteca, en el proceso se asume que se instala una biblioteca llamada SciPy pero resulta que siempre da el error que no aparece por lo cual dicha biblioteca se la tiene que instalar independientemente. SciPy es muy importante en la ejecución del algoritmo ya que contiene algoritmos para optimización, algebra lineal, integración, interpolación, funciones especiales, procesamiento de señales e imágenes.

### 3.6.3. Biblioteca imutils

Esta biblioteca se acopla de mejor manera cuando se utiliza Python y OpenCV, sus funciones principales tienen que ver con el procesamiento de imágenes, rotación, cambio en el tamaño, esqueletización, clasificación de contornos y detección de bordes. De igual manera esta biblioteca es ocupada en el trabajo y se la debe instalar.

## 3.7. Herramienta OpenCV

OpenCV es una biblioteca multiplataforma, pertenece a código abierto principalmente utilizada en la visión artificial, esta biblioteca se ha utilizado en diferentes aplicaciones como reconocimiento de objetos, detección de movimiento, seguridad, control de procesos, etc. Dispone

de una gran variedad de funciones que justamente ayudan y benefician en las aplicaciones de detección y reconocimiento. En este trabajo esta biblioteca se encarga del procesamiento de imágenes.

### 3.8. Nvidia CUDA

El modelo CUDA (Arquitectura Unificada de Dispositivos de Cómputo) creado por Nvidia, su función principal es aprovechar el paralelismo, y el alto ancho de banda de la memoria en las GPU en aplicaciones con un gran coste aritmético frente a realizar numerosos accesos a memoria principal, lo que podría actuar de cuello de botella. El modelo de programación de CUDA está diseñado para que se generen aplicaciones que de forma transparente escalen su paralelismo para poder incrementar el número de núcleos computacionales. Este diseño contiene tres puntos claves, que son la jerarquía de grupos de hilos, las memorias compartidas y las barreras de sincronización. (Wikipedia, 2019)

Este modelo debe ser instalado al principio de la actualización o flasheo del kit de desarrollo, esto se debe por la razón que ayuda a aprovechar el GPU y procesamiento en las aplicaciones que en este caso se basa en el aprendizaje profundo, si uno no lo hace o se salta este paso puede llevar a desaprovechar el kit de desarrollo que se esté ocupando y de igual manera resultara una mala efectividad del algoritmo a ejecutar.

### 3.9. Aplicación IP WebCam

Esta aplicación de IP WebCam como se muestra en la Figura 28, nos ayuda en el presente trabajo ya que el video de entrada para la ejecución del algoritmo contador de personas puede ser bajo el

entorno IP, utilizando un teléfono celular con sistema operativo Android. Funciona cuando el teléfono está conectado a una red WiFi en la cual le muestra la dirección IP y el puerto asignado por la red, posterior tomamos dicho valor y ya podemos acceder desde otro dispositivo y poder observar el video.

Como se aprecia en la Figura 30 nos muestra la salida de video donde aparece la dirección IP y puerto asignado, proporcionado por la aplicación.

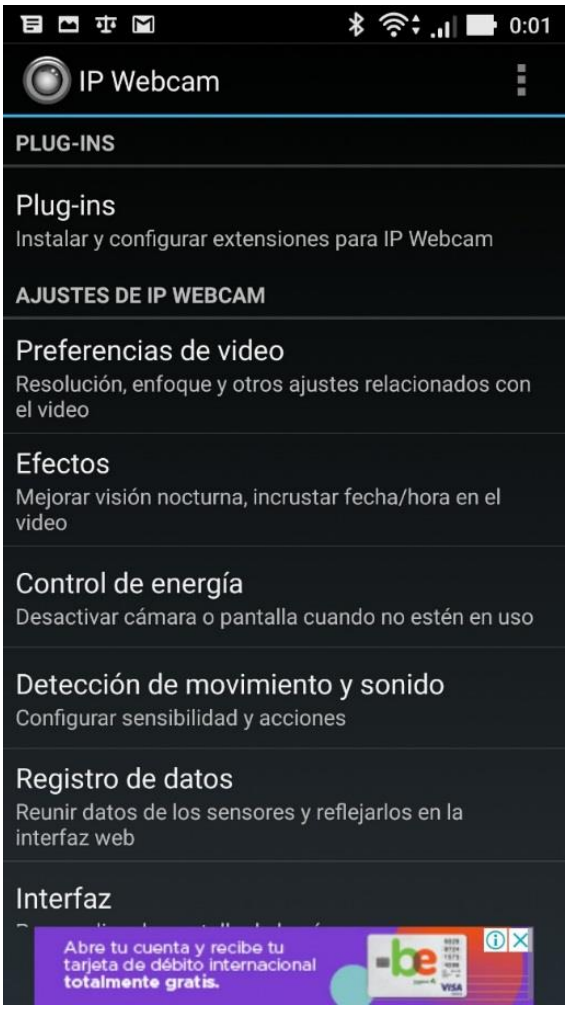



***Figura 29.*** Aplicación Ip WebCam


***Figura 30.*** Salida de video aplicación Ip WebCam

Hay que tomar en cuenta que las configuraciones de video establecidas por defecto son muy robustas para poder utilizar como entrada en el algoritmo, la aplicación si nos permite cambiar diferentes parámetros de video los cuales se puede observar en la Figura 31, estos son los necesarios a cambiar para un funcionamiento aceptable del algoritmo.

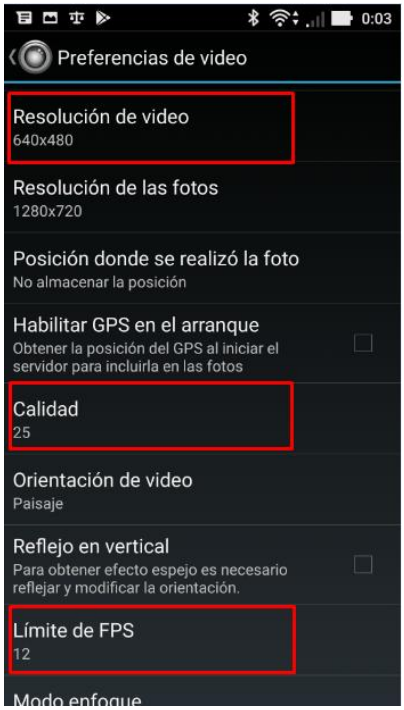



***Figura 31.*** Configuración de parámetros de video

## 3.10. Cámara de video IP.

En el presente trabajo se utilizó una cámara IP independiente gracias a las configuraciones y calidad de video que se obtiene para utilizar en la aplicación del contador de personas.

La cámara IP usada posee las siguientes características:

- Marca 3S Visión
- Tipo Domo de 2 Megapíxeles / 720p
- Códec dual H.264
- Infrarrojo con alcance de 15 metros
- Reducción de ruido 3D
- Análisis de video inteligente.

***Figura 32.*** Cámara IP 3S Visión
Fuente: (3SVision, 2017)

### 3.10.1. Protocolo RTSP (Protocolo de transmisión en tiempo real)

RTSP es un protocolo perteneciente a la capa de aplicación, no orientado a la conexión. En lugar de esto el servidor RTSP mantiene una sesión asociada a un identificador (Session ID). En la

mayoría de los casos RTSP usa TCP para el envío de datos de control del reproductor (mensajes "out of band") y UDP para los datos de audio y vídeo (mensajes "in band"). De forma intencionada, el protocolo es similar en sintaxis y operación a HTTP, por lo que los mecanismos de expansión añadidos a HTTP pueden, en muchos casos, añadirse a RTSP. (Apablaza, 2018)

Para el uso de una cámara independiente IP se necesita que soporte este protocolo de red, ya que es la manera como se le puede visualizar desde otro dispositivo. Para poder utilizar la cámara en la aplicación de este trabajo debemos ocupar la interfaz “cv - VideoCapture” junto con la URL de conexión de la cámara que se esté ocupando.

```
rtsp : // <user_name>: <user_passwd> @ <cameraIP>: <port> / <stream_path>! decodebin! videoconvert! xvimagesink
```

***Figura 33.*** Declaración adecuada de URL para conexión de cámara IP.
Fuente: (NvidiaDeveloper, 2019)

# CAPITULO 4

# 4. IMPLEMENTACIÓN DEL ALGORITMO CONTADOR DE PERSONAS

## 4.1. Introducción

En el presente trabajo se implementa un algoritmo utilizando las herramientas de OpenCV y Python bajo el entorno de Linux (Ubuntu), el cual se basa en el conteo de personas que ingresan o salen, evaluado en la garita principal de la Universidad de las Fuerzas Armadas ESPE. En este capítulo se detallará cómo funciona el algoritmo tanto en la detección como el seguimiento.

## 4.2. Explicación en la detección y seguimiento de objetos.

La Detección: El algoritmo que usaremos ya se encuentra pre entrenado el cual es el SSD de MobilNet, durante esta fase el detector se ejecuta cada N cuadros, consiste en actualizar cada objeto nuevo que aparezca en el marco de la imagen generando nuevas coordenadas en cada cuadro delimitador.

El seguimiento: El algoritmo a utilizar se basa en el seguimiento del Centroide, el cual consiste en receptar el conjunto de cuadros delimitadores proporcionado en la fase de la detección, calculado su centro o centroide, se procede al rastreo dentro del marco del video, con el movimiento que se genere en cada centroide se discriminara para que dirección se mueve, es decir, si entra o sale del lugar evaluado.

## 4.3. Implementación del algoritmo contador de personas

Iniciamos importando todos los paquetes necesarios para la implementación. Dentro de la carpeta (tracking) se encuentran los algoritmos de seguimiento a importar, de igual forma necesitamos la biblioteca (imutils) para funciones de conveniencia que requiera OpenCV, la biblioteca (dlib) se usará para la implementación del rastreador de correlación y OpenCV se hará uso en lo que tiene que ver con el video de entrada.

```
from tracking.centroidtracker import CentroidTracker
from tracking.trackableobject import TrackableObject
from imutils.video import VideoStream
from imutils.video import FPS
import numpy as np
import argparse
import imutils
import time
import dlib
import cv2
```

***Figura 34.*** Importación de paquetes necesarios.
Fuente: (Rosebrock, 2018)

Ahora se requiere introducir los “argumentos de línea de comando”, es decir, son indicadores que se proporciona a un programa en tiempo de la ejecución, dispone de información adicional que nos sirve en el programa cuando está trabajando.

Dada que la aplicación se basa en aprendizaje profundo de redes neuronales convolucionales los argumentos a declarar tienen que ver específicamente con el detector utilizado.

```
ap = argparse.ArgumentParser()
ap.add_argument("-p", "--prototxt", required=True,
    help="path to Caffe 'deploy' prototxt file")
ap.add_argument("-m", "--model", required=True,
    help="path to Caffe pre-trained model")
ap.add_argument("-i", "--input", type=str,
    help="path to optional input video file")
ap.add_argument("-o", "--output", type=str,
    help="path to optional output video file")
ap.add_argument("-c", "--confidence", type=float, default=0.4,
    help="minimum probability to filter weak detections")
ap.add_argument("-s", "--skip-frames", type=int, default=30,
    help="# of skip frames between detections")
args = vars(ap.parse_args())
```

***Figura 35.*** Argumentos de línea empleado en el detector de objetos SSD.
Fuente: (Rosebrock, 2018)

Como se puede observar en la Figura 35, se establece 6 argumentos los cuales se va a detallar a continuación:

- Argumento "--prototxt": Se establece para direccionar la ruta de archivo de Caffe.
- Argumento "--model": Direccionamiento al modelo CNN pre entrenado de Caffe.
- Argumento "--input": Dirección del archivo de entrada de video, en caso que no se haga uso se ocupará la propia cámara web.
- Argumento "--output": Dirección de video de salida, si no se especifica no se grabará ningún video de la ejecución del programa.
- Argumento "--confidence": Se establece un valor por default de 0.4, hace referencia al umbral de probabilidad mínima para discriminar detecciones endebles.
- Argumento "--skip-frames": Se establece un valor por default de 30, hace referencia a la cantidad de Frames por segundo que espera el detector antes de volver a ejecutarse, en este caso el detector SSD de MobilNet.

Ahora establecemos parámetros del detector SSD.

```
CLASSES = ["background", "aeroplane", "bicycle", "bird", "boat",
    "bottle", "bus", "car", "cat", "chair", "cow", "diningtable",
    "dog", "horse", "motorbike", "person", "pottedplant", "sheep",
    "sofa", "train", "tvmonitor"]

print("[INFO] cargando modelo...")
net = cv2.dnn.readNetFromCaffe(args["prototxt"], args["model"])
```

***Figura 36.*** Parámetros del detector SSD.

Fuente: (Rosebrock, 2018)

Como se indica en la Figura 35 se inicializará la lista de clases que admite el detector, en el caso de nuestro trabajo solo se condiciona al reconocimiento de personas, si llega a ser el caso de un cambio en la aplicación tenemos otras opciones para detectar objetos como ejemplo vehículos. A continuación de establecer las clases, cargamos nuestro detector pre entrenado SSD para poder hacer uso en nuestro algoritmo.

Cargado el algoritmo de detección continuamos con la declaración para la entrada de video.

```
if not args.get("input", False):
    print("[INFO] Inicializando video...")
    vs = VideoStream(src=0).start()
    time.sleep(2.0)

else:
    print("[INFO] Inicializando video externo...")
    vs = cv2.VideoCapture("http://<direccion IP>:<puerto de conexion>/video")
    vs = cv2.VideoCapture("rtsp://<user_passwd>@<direccion IP>:<puerto>/
        <stream path>! decodebin! videoconvert! xvimagesink /video")
```

1

2

***Figura 37.*** Inicialización para la entrada de video.

Como se indica en la Figura 37, el inicio del condicionamiento (if) hace referencia para el uso de la cámara integrada en este caso del módulo Jetson Tx1. La segunda parte indica el uso para una

cámara externa bajo el entorno IP. Se encuentran establecidas dos declaraciones por la razón de mostrar los casos en los que se puede hacer uso, el (1) se lo puede ocupar cuando se utilice una cámara integrada en un teléfono celular esto a través de la aplicación IP WebCam, la (2) declaración es la que en este trabajo se ocupó, la cual se trata de una cámara independiente IP

Continuaremos con las inicializaciones que se requieren para el algoritmo.

```
writer = None

W = None
H = None

ct = CentroidTracker(maxDisappeared=40, maxDistance=50)
trackers = []
trackableObjects = {}

totalFrames = 0
totalDown = 0
totalUp = 0

fps = FPS().start()
```

***Figura 38.*** Inicialización de parámetros restantes

Fuente: (Rosebrock, 2018)

- W y H: Dimensiones del marco de video, posterior se conectará a cv2.VideoWriter.
- ct: Instancia para el seguimiento de centroide.
- Trackers: Lista que almacena los rastreadores de correlación dlib.
- trackableObjects: Asignación de un ID para el seguimiento del objeto.
- totalFrames: Valor total de frames procesados.
- totalDown y totalUp: Variables que van registrando el conteo de personas que ingresan o salen del lugar evaluado.

A continuación, se establecerá parámetros en el marco del video.

```
while True:

    frame = vs.read()
    frame = frame[1] if args.get("input", False) else frame

    if args["input"] is not None and frame is None:
        break

    frame = imutils.resize(frame, width=500)
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)

    if W is None or H is None:
        (H, W) = frame.shape[:2]

    if args["output"] is not None and writer is None:
        fourcc = cv2.VideoWriter_fourcc(*"MJPG")
        writer = cv2.VideoWriter(args["output"], fourcc, 30, (W, H), True)
```

***Figura 39.*** Parámetros de ajuste en la entrada de video.

Fuente: (Rosebrock, 2018)

En la Figura 39 se puede observar el ajuste de tamaño de entrada de video al valor de 500, de igual forma se realiza el intercambio de canales de color por RGB para el pre procesamiento del marco del video.

En la parte final, se crea una instancia la cual indica la ruta de salida de video, con la finalidad de tener evidencia cada vez que el algoritmo se ejecute.

Creación de estados en el proceso del algoritmo, como se observa en la Figura 39.

En la Figura 40 se establece diferentes “estados” que se detallaran a continuación:

- Espera: En este estado se indica que se está esperando hasta que los objetos sean detectados.
- Detectando: En este estado el algoritmo de detección está en funcionamiento
- Rastreando: En este estado entra en funcionamiento el algoritmo de seguimiento

```
status = "Espera"
rects = []

if totalFrames % args["skip_frames"] == 0:

    status = "Detectando"
    trackers = []

    blob = cv2.dnn.blobFromImage(frame, 0.007843, (W, H), 127.5)
    net.setInput(blob)
    detections = net.forward()
```

***Figura 40.*** Estados que intervienen en el algoritmo.

Fuente: (Rosebrock, 2018)

Proceso para discriminar detecciones débiles y de igual manera restringir la detección de solo personas.

```
for i in np.arange(0, detections.shape[2]):

    confidence = detections[0, 0, i, 2]

    if confidence > args["confidence"]:

        idx = int(detections[0, 0, i, 1])

        if CLASSES[idx] != "person":
            continue
```

***Figura 41.*** Discriminación de objetos y detecciones.

Fuente: (Rosebrock, 2018)

Asignación del cuadro delimitador para cada persona detectada, dicho cuadro nos ayudara en el siguiente proceso que interviene el seguimiento.

```
box = detections[0, 0, i, 3:7] * np.array([W, H, W, H])
(startX, startY, endX, endY) = box.astype("int")

tracker = dlib.correlation_tracker()
rect = dlib.rectangle(startX, startY, endX, endY)
tracker.start_track(rgb, rect)

trackers.append(tracker)
```

***Figura 42.*** Cuadro delimitador de la persona detectada.

Fuente: (Rosebrock, 2018)

Proceso en el que se ejecuta el “estado” de seguimiento o tracking, obteniendo valores de coordenadas en el marco de video para posterior registro de entrada o salida de la persona.

```
else:

    for tracker in trackers:

        status = "Rastreando"

        tracker.update(rgb)
        pos = tracker.get_position()

        startX = int(pos.left())
        startY = int(pos.top())
        endX = int(pos.right())
        endY = int(pos.bottom())

        rects.append((startX, startY, endX, endY))
```

***Figura 43.*** Estado que interviene el tracking.

Fuente: (Rosebrock, 2018)

Ahora se procede a “dibujar” una línea horizontal dentro del marco del video, en la cual las personas detectadas deben cruzar para ser rastreadas e identificar su dirección.

```
cv2.line(frame, (0, H // 2), (W, H // 2), (0, 255, 255), 2)

objects = ct.update(rects)
```

***Figura 44.*** *Línea de frontera para conteo.*

Fuente: (Rosebrock, 2018)

Casi como parte final se complementa la función del seguimiento donde se realiza el reconocimiento si la persona se dirige hacia arriba o hacia abajo dentro del marco.

```
for (objectID, centroid) in objects.items():

    to = trackableObjects.get(objectID, None)

    if to is None:
        to = TrackableObject(objectID, centroid)

    else:

        y = [c[1] for c in to.centroids]
        direction = centroid[1] - np.mean(y)
        to.centroids.append(centroid)

        if not to.counted:

            if direction < 0 and centroid[1] < H // 2:
                totalUp += 1
                to.counted = True

            elif direction > 0 and centroid[1] > H // 2:
                totalDown += 1
                to.counted = True

    trackableObjects[objectID] = to
```

***Figura 45.*** Reconocimiento para el conteo de personas.

Fuente: (Rosebrock, 2018)

Para la validación en el contador de personas se lo realiza tomando los valores de las coordenadas de todo el conjunto de centroides calculados, posterior se realiza el cálculo para la dirección tomando la diferencia entre la ubicación actual y la media de la ubicación anterior, si la dirección resulta que es negativa indica que la persona se mueve hacia arriba (entrando) y el centroide está por encima de la línea de frontera. Si la dirección resulta con valor positivo indica que la persona se mueve hacia abajo (sale) y el centroide está por debajo de la línea de frontera (Rosebrock, 2018)

Como último paso se indica en el video la información de la cuenta que va generando el algoritmo.

```
        text = "ID {}".format(objectID)
        cv2.putText(frame, text, (centroid[0] - 10, centroid[1] - 10),
            cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 255, 0), 2)
        cv2.circle(frame, (centroid[0], centroid[1]), 4, (0, 255, 0), -1)

    info = [
        ("Entra", totalUp),
        ("Sale", totalDown),
        ("Estado", status),]

    for (i, (k, v)) in enumerate(info):
        text = "{}: {}".format(k, v)
        cv2.putText(frame, text, (10, H - ((i * 20) + 20)),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (0, 0, 255), 2)
```

***Figura 46.*** Información de valores según el conteo de personas.

Fuente: (Rosebrock, 2018)

# CAPITULO 5

# 5. PRUEBAS Y ANÁLISIS DE RESULTADOS

## 5.1. Escenario

En el presente trabajo, el prototipo contador de personas fue evaluado en la garita principal de la Universidad de las Fuerzas Armadas ESPE como se puede observar en la Figura 47. Esto realizado con la previa autorización de los militares de turno encargados de la seguridad de la universidad, las diferentes pruebas se llevaron a cabo dentro de noviembre a diciembre del 2019.

***Figura 47.*** Garita principal de entrada de la Universidad "ESPE".

La entrada de video a ocupar en el presente trabajo es muy importante, por lo cual se ha utilizado una cámara independiente IP por las configuraciones que le permite al usuario.

***Figura 48.*** Cámara IP

Las pruebas a realizar establece diferentes parámetros de configuración tanto en la entrada de video como en el algoritmo, se realizara varias pruebas en dos ambientes controlados distintos, es decir, luz natural que comprende el ambiente durante el día y por otro lado luz artificial o baja iluminación que comprende el ambiente de la noche, cabe acotar que se realiza esta evaluación por el motivo del cambio de iluminación que se genera y que puede producir mayor error del prototipo en diferente ambiente.

## 5.2. Configuraciones del prototipo

### 5.2.1. Configuración en el video de entrada.

Previamente con la entrada de video se realizó diferentes pruebas en las cuales se llegó a definir los siguientes parámetros:

- **Resolución de video:** Este parámetro es importante ya que define la calidad con la que se va a poder visualizar, el algoritmo contador de personas no acepta valores altos como ejemplo 1024x768, esto se debe principalmente por costo computacional que llevaría esos valores de entrada, dicho aquello el valor que mejores resultados proporciono es la resolución de 640x480.
- **Flujo de datos:** Este parámetro hace referencia a la cantidad de datos que proporcionara la cámara al módulo Jetson Tx1, dicho valor no tiene que ser ni muy alto ni muy bajo, ya que en los dos casos puede llegar a no distinguirse bien los objetos en la imagen o por otro lado sobrecarga la información volviendo lento el video de entrada. El valor con mejores resultados es el de 384k.
- **Fotogramas por segundo:** Este parámetro es similar al concepto del flujo de datos mencionado anteriormente, de igual forma no puede ser ni muy alto ni bajo el valor, básicamente influye el costo computacional de la tarjeta utilizada. El valor con mejores resultados es el de 10 FPS.

### 5.2.2. Configuraciones en el algoritmo.

En el presente trabajo para la detección de objetos se hace uso de un algoritmo pre entrenado, el cual es el SSD de disparo único, básicamente este método reconoce objetos cada $N$ – frames, dicho valor como se puede observar en la Figura 35 se encuentra por default con 31, de igual manera en la Figura 37 se encuentra un valor que se puede observar al generar la instancia “ct” que hace referencia al uso de “CentroidTracker”, los valores que están conformando tienen relación al número máximo de fotogramas consecutivos que permanece el (ID) del objeto rastreado en video y el valor de la distancia euclidiana para discriminar nuevos o existentes objetos.

Dicho valor que interviene que permanezca el objeto rastreado en pantalla no puede tener un valor alto, esto se debe por la razón que el ID del objeto contabilizado se quedaría demasiados fotogramas generando la perdida de nuevas detecciones y conteos que puede aparecer en la secuencia del video, el valor tampoco puede ser bajo por la pérdida rápida del ID en el marco del video llegando a no lograr contabilizarle y generar errores consecutivos. Después de previas pruebas se llegó a definir como el mejor, (maxDisappeared=30).

En la Figura 49 se puede apreciar un ejemplo con un valor (maxDisappeared=40), los ID 4, 9,10,11 que se encuentran en la zona de conteo que registra la salida del objeto y los ID 5,6 que se encuentran en la otra zona de conteo de los objetos que ingresan luego de ser contabilizados se quedan más tiempo en el marco del video por los fotogramas configurados, generando la perdida de nuevas detecciones y seguimiento para el contador de personas.

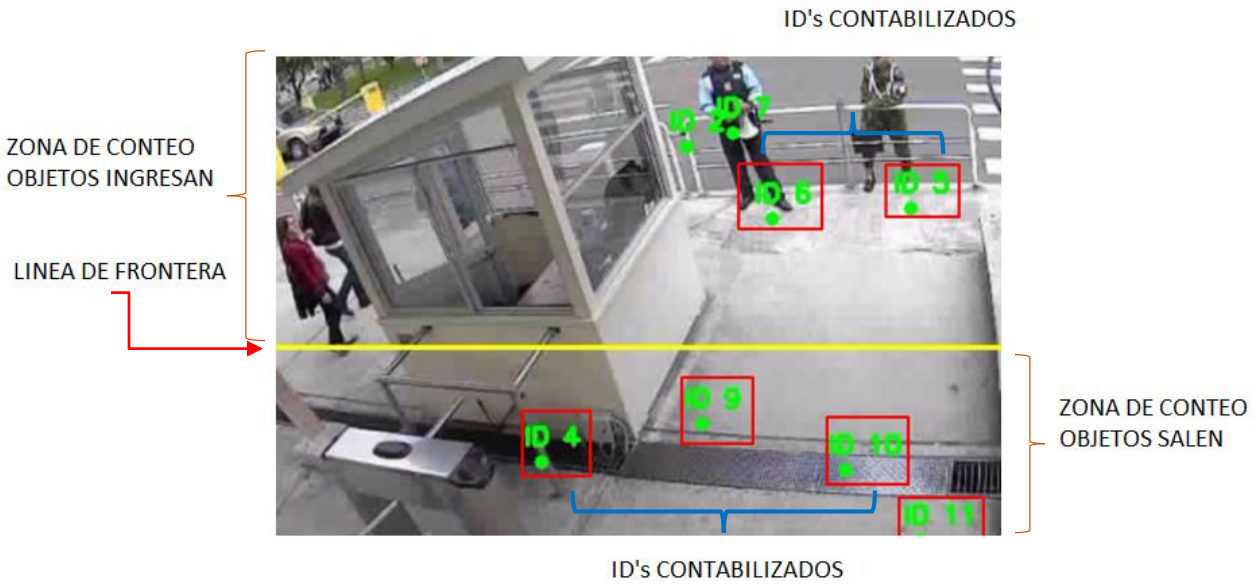



***Figura 49.*** Inconveniente si no se configura el valor que conforma “CentroidTracker”

El valor en el que interviene el algoritmo de detección será evaluado con diferentes *N*- frames junto con las configuraciones de video y algoritmo previamente mencionadas, esto se realizará con la finalidad de determinar los parámetros adecuados para el contador de personas en los dos ambientes controlados propuestos.

## 5.3. Pruebas realizadas en el ambiente a luz natural (Día).

Las pruebas a realizar toman en cuenta las configuraciones mencionadas anteriormente.

### 5.3.1. Primer tipo de configuración.

Las configuraciones realizadas en esta primera prueba se detallan a continuación:

- Resolución de video y flujo de datos: 640x480 / 384k
- Fotogramas por segundo: 10 FPS.
- Valor de *N* – frames para la detección: 5 FPS.
- Valor de rastreo (maxDisappeared): 30 FPS.

**Tabla 2.**
*Resultados de la primera configuración en el escenario del día.*

| ***ESCENARIO DIA*** | ***NUMERO DE PERSONAS QUE INGRESAN*** | ***NUMERO DE PERSONAS QUE SALEN*** |
|---|---|---|
| ***Personas contadas por un observador externo*** | 28 | 11 |
| ***Personas contadas por el prototipo*** | 20 | 7 |
| ***Porcentaje de acierto*** | 71.43 % | 63.63% |

Esta configuración se realizó en los dos ambientes propuestos, el análisis del resultado se encuentra detallado en la evaluación que se realizó en el ambiente con luz artificial (noche).

### 5.3.2. Segundo tipo de configuración.

Las configuraciones realizadas en esta segunda prueba se detallan a continuación:

- Resolución de video y flujo de datos: 640x480 / 384k
- Fotogramas por segundo: 10 FPS.
- Valor de *N* – frames para la detección: 15 FPS.
- Valor de rastreo (maxDisappeared): 30 FPS.

**Tabla 3.**
*Segunda configuración de parámetros.*

| ***ESCENARIO*** <br> ***DIA*** | ***NUMERO DE PERSONAS QUE INGRESAN*** | ***NUMERO DE PERSONAS QUE SALEN*** |
|---|---|---|
| ***Personas contadas por un observador externo*** | ***8*** | ***11*** |
| ***Personas contadas por el prototipo*** | ***7*** | ***7*** |
| ***Porcentaje de acierto*** | ***87.5 %*** | ***63.4%*** |

Esta configuración se realizó en los dos ambientes propuestos, el análisis del resultado se encuentra detallado en la evaluación que se realizó en el ambiente con luz artificial (noche).

### 5.3.3. Tercer tipo de configuración.

Las configuraciones realizadas en esta tercera prueba se encuentran detalladas en la Tabla 3.

- Resolución de video y flujo de datos: 640x480 / 384k
- Fotogramas por segundo: 10 FPS.
- Valor de *N* – frames para la detección: 9 FPS.
- Valor de rastreo (maxDisappeared): 30 FPS.

**Tabla 4.**
*Tercera configuración de parámetros.*

| ***ESCENARIO*** <br> ***DIA*** | ***NUMERO DE PERSONAS QUE INGRESAN*** | ***NUMERO DE PERSONAS QUE SALEN*** |
|---|---|---|
| ***Personas contadas por un observador externo*** | 7 | 6 |
| ***Personas contadas por el prototipo*** | 7 | 5 |
| ***Porcentaje de acierto*** | 100 % | 83.33 % |

Como se puede apreciar en este caso las configuraciones generan un equilibrio entre el video de entrada y el algoritmo contador de personas, se logra un porcentaje de acierto y precisión bastante aceptable.

Cabe mencionar mientras no exista ese acople adecuado en las configuraciones no funcionara de manera óptima ni la detección ni el seguimiento de objetos.


***Figura 50.*** Prueba realizada con la mejor configuración.

### 5.3.4. Resultados obtenidos ambiente luz natural (Día)

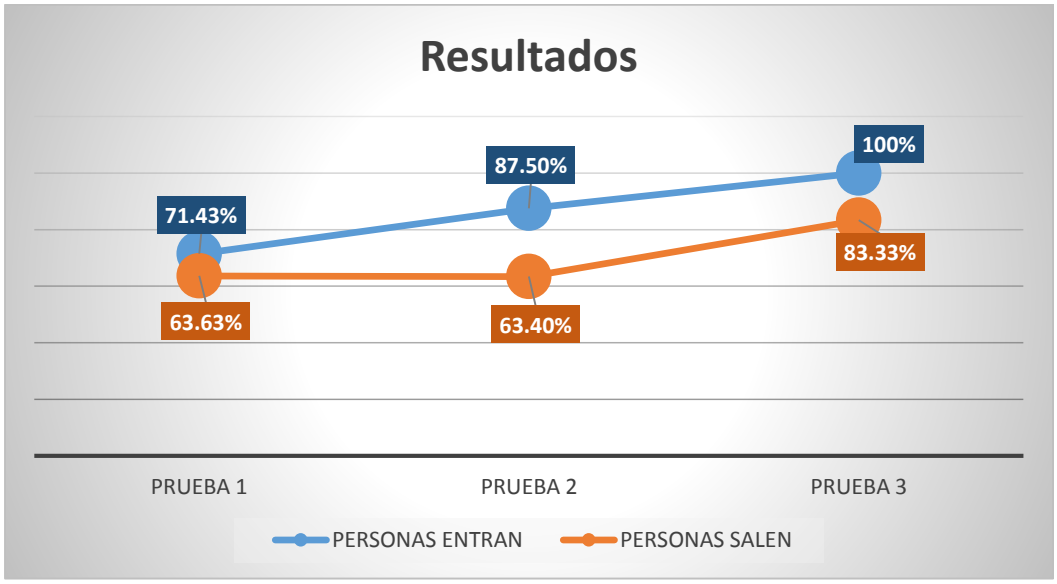



***Figura 51.*** Gráfica de resultado, ambiente luz natural (Día)

La Figura 51 indica el resultado obtenido en el ambiente a luz natural (Día), los valores muestran que a medida se fue modificando el valor en el que se ejecuta la detección se mejoró el porcentaje

de acierto llegando así al valor adecuado, cabe mencionar que los valores a medida de cada cambio no resultan tan pésimos y esto se debe por la iluminación de luz natural que se dispone en este ambiente ayudando en la entrada de video que registra la cámara utilizada. El mejor porcentaje de acierto se puede considerar como una medida de confianza aceptable para que el prototipo sea ocupado con normalidad en el ambiente a luz natural (día).

## 5.4. Pruebas realizadas en el ambiente con luz artificial (Noche)

### 5.4.1. Primer tipo de configuración.

Las configuraciones realizadas en esta primera prueba se detallan a continuación:

- Resolución de video y flujo de datos: 640x480 / 384k
- Fotogramas por segundo: 10 FPS.
- Valor de *N* – frames para la detección: 5 FPS.
- Valor de rastreo (maxDisappeared): 30 FPS.

**Tabla 5.**
*Primera configuración de parámetros en el escenario de la noche.*

| ***ESCENARIO***<br><br>***DIA*** | ***NUMERO DE PERSONAS QUE INGRESAN*** | ***NUMERO DE PERSONAS QUE SALEN*** |
|---|---|---|
| ***Personas contadas por un observador externo*** | 5 | 15 |
| ***Personas contadas por el prototipo*** | 3 | 10 |
| ***Porcentaje de acierto*** | 60 % | 66.6% |

Al evaluar esta configuración baja significativamente el nivel de confianza del contador de personas, esto se debe básicamente por la carga de información que ahora procesa el módulo Jetson Tx1, ya que se estableció con menos *N* – frames el conjunto de detecciones que se envía es más rápido y lo que genera es que se vuelva lento el video y no llegue a procesar toda la información con lo cual el limitante es el hardware ocupado. Cabe mencionar que la misma configuración se evaluó en el Día y genero mejor resultado esto se debe por la intensidad de luz natural que se recibe en ese periodo.

### 5.4.2. Segundo tipo de configuración.

Las configuraciones realizadas en esta segunda prueba se detallan a continuación:

- Resolución de video y flujo de datos: 640x480 / 384k
- Fotogramas por segundo: 10 FPS.
- Valor de *N* – frames para la detección: 15 FPS.
- Valor de rastreo (maxDisappeared): 30 FPS.

**Tabla 6.**
*Segunda configuración de parámetros escenario de la noche.*

| ***ESCENARIO*** <br> ***DIA*** | ***NUMERO DE PERSONAS QUE INGRESAN*** | ***NUMERO DE PERSONAS QUE SALEN*** |
|---|---|---|
| ***Personas contadas por un observador externo*** | 10 | 19 |
| ***Personas contadas por el prototipo*** | 3 | 16 |
| ***Porcentaje de acierto*** | 30 % | 84.21% |

Al evaluar esta configuración bajo aún más el nivel de confianza del contador de personas, esto se debe a que el valor de N – frames como es mayor se demora un poco más el detector en enviar el conjunto de objetos detectados, esto genera que en grupos de estudiantes que al mismo tiempo entren y salgan de la garita no van a poder ser todos detectados, esto se puede observar mejor en la Figura 46. Cabe mencionar que la misma configuración se evaluó en el Día y genero mejor resultado esto se debe por la intensidad de luz natural que se recibe en ese periodo.


***Figura 52.*** Prueba del algoritmo con valor de N – frames alto para el detector.

### 5.4.3. Tercer tipo de configuración.

Las configuraciones realizadas en esta segunda prueba se detallan a continuación:

- Resolución de video y flujo de datos: 640x480 / 384k

- Fotogramas por segundo: 10 FPS.
- Valor de *N* – frames para la detección: 9 FPS.
- Valor de rastreo (maxDisappeared): 30 FPS.

**Tabla 7.**
*Tercera configuración de parámetros escenario de la noche.*

| ***ESCENARIO*** <br> ***DIA*** | ***NUMERO DE PERSONAS QUE INGRESAN*** | ***NUMERO DE PERSONAS QUE SALEN*** |
|---|---|---|
| ***Personas contadas por un observador externo*** | 21 | 9 |
| ***Personas contadas por el prototipo*** | 19 | 8 |
| ***Porcentaje de acierto*** | 90.47 % | 88.88% |

Esta configuración es la misma que genero el mejor resultado en el escenario del Día, a pesar de que se baja un poco el nivel de confianza es todavía aceptable para poder mantener una sola configuración en todas las 24 horas si se desea, la cámara IP brinda la facilidad de cuando ya es muy poca o nada de la luz artificial en la noche, se activa sus infrarrojos y el video aparece en blanco y negro.

Cabe mencionar que en el escenario del día los resultados son mejores por la intensidad de luz que registra la cámara de igual forma si se desearía mejores resultados el limitante es en este caso la cámara utilizada.

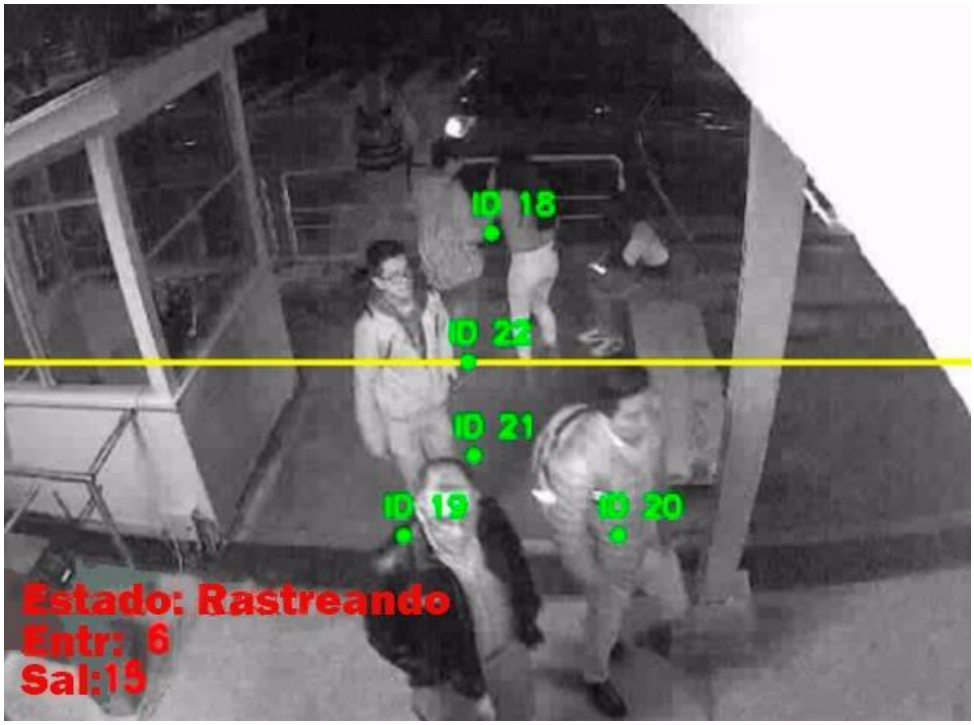



***Figura 53.*** Prueba con la mejor configuración ambiente de la noche.

### 5.4.4. Resultados obtenidos en ambiente a luz artificial (Noche)

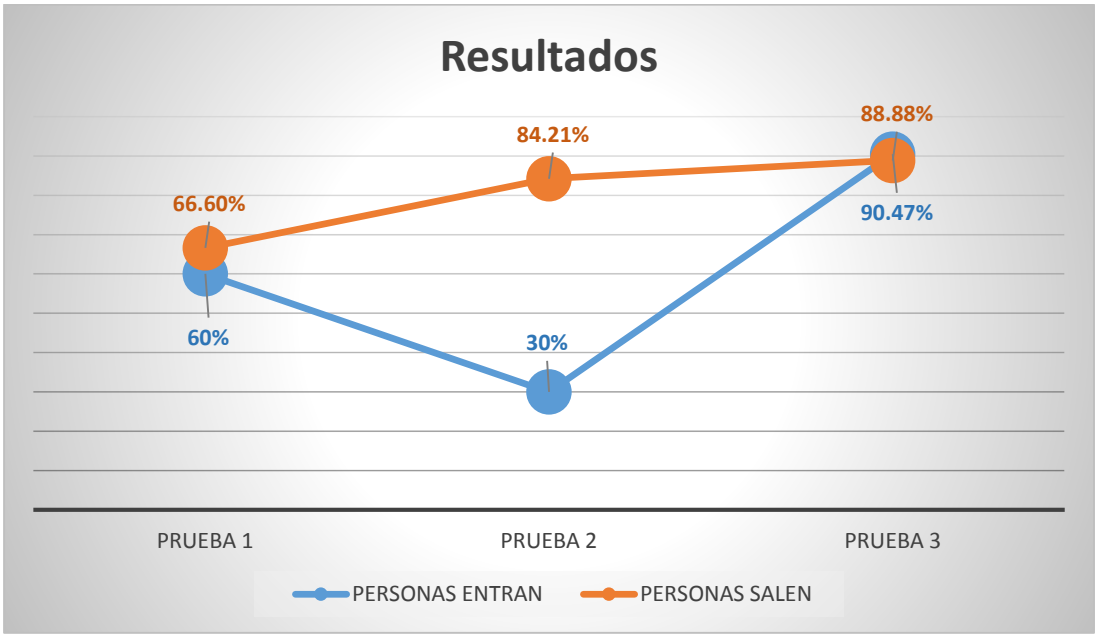



*Figura 54*. Gráfica de resultado, ambiente luz artificial (noche)

La Figura 54 nos indica los resultados en el ambiente a luz artificial o luz mínima (Noche), los valores muestran un porcentaje de acierto más bajo a comparación del otro ambiente evaluado, esto se debe por la razón de la iluminación del ambiente, la solución a este problema fue el uso de una cámara independiente IP que dispone de la característica de poder cambiar a modo Infrarrojo cuando se encuentra a condiciones de ambiente de baja iluminación, generando que la entrada de video se muestre en blanco y negro, debido a esto se obtuvo con la mejor configuración un porcentaje de acierto aceptable que se puede considerar una medida de confianza a utilizar en este ambiente.

# CAPITULO 6

# 6. CONCLUSIONES Y RECOMENDACIONES

## 6.1. Conclusiones

- Se implementó un prototipo que entrelaza un algoritmo de detección de objetos con el algoritmo de seguimiento de objetos para desarrollar un sistema de conteo de personas utilizando un hardware dedicado de Nvidia Jetson Tx1, bajo un sistema operativo GNU-Linux con la distribución de Ubuntu 18 haciendo uso de herramientas como Python y OpenCV, obteniendo un porcentaje de acierto aproximado del 90 % considerado aceptable en los escenarios evaluados.
- Se evaluó el prototipo en distintos ambientes de iluminación, es decir, a luz natural (día) y luz artificial o baja iluminación (noche), comprobando la efectividad de los parámetros establecidos en el algoritmo para el funcionamiento las 24 horas del día, manteniendo el porcentaje de acierto ante cualquier nivel de iluminación.
- En las pruebas realizadas del prototipo se determinó utilizar una cámara independiente IP, en este caso de la marca 3S Vision, configurando la resolución de la entrada de video a 640x480 px, como también el flujo de datos a 384Kbps hacia el módulo Jetson Tx1. Para la detección, el número de fotogramas de la entrada de video se configuro a 10 FPS, con lo que prevenimos que exista un desbalanceo en las funciones de reconocimiento y posterior conteo.

- El porcentaje de acierto que se obtuvo en cada ambiente evaluado es bastante bueno, pero cabe mencionar que el prototipo posee un valor de error inferior al 10%, que por lo general se presenta por la acumulación de personas. Esto se debe al número de *N*-frames que debe esperar el algoritmo de detección para poder iniciar y empezar todo el procedimiento de conteo.
- Se realizo un análisis previo en el cual se determinó el uso del algoritmo detector de objetos de disparo único SSD (Single Shot Detection), la principal característica que proporciona tiene relación a la velocidad y precisión generando un equilibrio entre los dos parámetros, esto beneficia en el entrelazamiento que debe existir con el algoritmo de seguimiento.
- Para la entrada de video se configuró en la cámara la resolución con la que va a trabajar el prototipo, se determinó el valor de 640x480 px por ser un parámetro intermedio. Para el caso de una alta resolución se genera una sobrecarga al módulo Jetson Tx1 o si es el caso de una resolución baja genera una mala calidad en la entrada de video. Por este motivo en el programa se estableció el valor de 500x480 px con la finalidad de optimizar la calidad de video y que pueda trabajar de la mejor manera el contador de personas.

## 6.2. Recomendaciones

- Para poder obtener un valor de confianza mucho más alto es necesario reemplazar el hardware que se utiliza, tanto el módulo de Nvidia como la cámara, la detección para que sea mucho más rápida requiere más costo computacional y en la entrada de video se

requiere mejor calidad de imagen para que se haga más fácil la detección y seguimiento en ambos escenarios.

- Para trabajos futuros este prototipo se puede utilizar en aplicaciones como: seguridad ejemplo cercas virtuales, de igual forma se puede ocupar para el conteo ya no en el caso de personas sino de vehículos, conteo de personas, pero ya no en una garita sino en otro ambiente como en la entrada de los buses, etc.

# 7. REFERENCIAS BIBLIOGRÁFICAS

3SVision. (2017). 3S Vision. Retrieved from http://www.3svision.us/prod_info.php?pid=97&PHPSESSID=3cqr6fhlci58hr3av9fj1m7122

Albiol, A., Mora, I., & Naranjo, V. (2001). Real-Time High Density People Counter Using Morphological Tools. IEEE.

Amazon. (2016). Amazon. Retrieved from https://www.amazon.es/Nvidia-Jetson-TX1 Desarrollo-Cortex-A57/dp/B01M7S6LH1

Apablaza, C. (2018). profesores.elo. Retrieved from http://profesores.elo.utfsm.cl/~agv/elo323/2s10/projects/ApablazaBustamante/desc.html#

Arenas, M. (2015, Noviembre 11). Xataka. Retrieved from https://www.xataka.com/componentes/nvidia-jetson-tx1-un-cerebro-para-desarrollar-aplicaciones-de-ia-por-menos-de-300-dolares

Atriainnovation. (2019, Octubre 3). ATRIAINNOVATION. Retrieved from https://www.atriainnovation.com/deep-learning-aplicaciones/

CV-TRICKS. (2017). CV-Tricks. Retrieved from https://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

Da, L., Bodong, L., & Weigang, Z. (2014). Real-time Moving Vehicle Detection, Tracking, and Counting System Implmented with OpenCV. IEEE.

Gandhi, R. (2018, Julio 9). Towards data science. Retrieved from https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

Guanghan, N., Zhi, Z., Chen, H., & Zhihai, H. (2016, Julio 19). Spatially Supervised Recurrent Convolutional Neural Networks for Vsual Object Tracking. Columbia, Estados Unidos.

Izquierdo, R. O. (2015). Detección y seguimiento de objetos móviles en secuencias de video. España: Universidad de Valladolid.

Jovann, P. (2019). Análisis de tráfico vehicular mediante visión artificial. Ambato: Universidad Técnica de Ambato.

Khandelwal, R. (2019, Noviembre 30). Towards data science. Retrieved from https://towardsdatascience.com/ssd-single-shot-detector-for-object-detection-using-multibox-1818603644ca

Kyu Beom, L., & Hyu Soung, S. (2019). An application of a deep learning algorithm for automatic detection of unexpected accidents under bad CCTV monitoring condition in tunnels. IEEE.

Liu, W., Anguelov, D., Erhan, D., Szagedy, C., Reed, S., Fu, C.-Y., & Berg, A. (2016). SSD: Single Shot MultiBox Detector. Springer International Publishing, 21-37.

Nogueria, V., Oliveira, H., Silva, J., Vieira, T., & Oliveira, K. (2019). RetailNet: A deep learning approach for people counting and hot spots detection in retail stores. . IEEE, 2377-5416.

Nvidia. (2019). NVIDIA DEVELOPER. Retrieved from https://developer.nvidia.com/nvidia-sdk-manager

NvidiaDeveloper. (2019, Febrero 5). Nvidia Developer. Retrieved from https://devtalk.nvidia.com/default/topic/1047050/how-to-interface-a-ip-camera-/

Quantamagazine. (2019). Quantamagazine. Retrieved from https://www.quantamagazine.org/

Rosebrock, A. (2018, Agosto 13). OpenCV People Counter. Retrieved from pyimagesearch: https://www.pyimagesearch.com/2018/08/13/opencv-people-counter/

Rouse, M. (2017, Abril). TechTarget. Retrieved from https://searchdatacenter.techtarget.com/es/definicion/Aprendizaje-profundo-deep-learning

SmartPanel. (2019, Abril 10). SmartPanel. Retrieved from https://www.smartpanel.com/que-es-deep-learning/

Tsong-Yi, C., Chao-Ho, C., & Da-Jinn, W. Y.-L. (2010). A People Counting System Based on Face-Detection. IEEE.

Wen, H., Wu, G., & Li, J. (2015). An Improved Tracking-Learning-Detection Method. Chinese Control Conference.

Wikipedia. (2019). Wikipedia La enciclopedia libre. Retrieved from https://es.wikipedia.org/wiki/CUDA

Yuncai, L., Pan, W., & Hongtao, W. (2018). Target Tracking Algorithm Based on Deep Learning and Multi-Video Monitoring . IEEE.

Zhu, D., Chen, J., Yany, X., Shao, H., & Su, L. (2018). Traffic Sign Detection Method of Improved SSD Based on Deep Learning. IEEE.