CENTRO DE INGENIERÍA Y DESARROLLO INDUSTRIAL

REPORTE DE PROYECTO INDUSTRIAL

"Implementación de un Contador Inteligente con Módulo Computacional Neuronal MOVIDIUS"

ESPECIALIDAD DE TECNÓLOGO EN MECATRÓNICO

PRESENTA

Ing. Fausto Alfonso Angeles Bautista

Tutor Académico

Dr. Leonardo Barriga Rodríguez

Santiago de Querétaro, Querétaro. Octubre 2019

Agradecimientos.

Mediante este trabajo realizado en el Centro de Ingeniería y Desarrollo Industrial, sede Querétaro, permito mostrar el fruto de la investigación, recomendaciones, correcciones y aprendizaje que culmina en el proyecto aquí presentado. Proyecto Industrial que fue realizado gracias a diversas personas involucradas, a quienes deseo dar su debido agradecimiento en este apartado.

En primer lugar, quiero agradecer a Dios por concederme esta maravillosa oportunidad, donde se me ha permitido seguir desarrollando habilidades técnicas, mientras fortalecía aprendizajes, así como formaba nuevo conocimiento en áreas donde no me había desenvuelto.

Agradecer a mi familia, padres y hermanos, quienes durante toda la vida me han brindado su apoyo y amor, con el cual me ha sido posible siempre salir adelante, gracias a que me enseñaron que el éxito consiste en ser constante y nunca darse por vencido.

A mi tutor el Dr. Leonardo Barriga, quien me dio la confianza en el desarrollo del proyecto, ya que fue gracias a sus clases presentadas durante la especialidad donde me genero el interés en el procesamiento por visión, así como a sus consejos, recomendaciones y asesorías, mediante las cuales me fue posible encaminar el proyecto en el sentido correcto, le doy mi más sincero agradecimiento por todo el apoyo brindado, gracias.

También quiero dedicar este segmento a mis compañeros, con quienes entable una valiosa amistad que, durante las clases, conversaciones y trabajo en conjuntos de noches en vela, programando, armando circuitos, moviendo motores, haciendo el debido escrito de cada trabajo con quienes me apoye para retroalimentar y mejorar trabajos encargados por los profesores. Profesores a quienes les estoy agradecido porque siempre buscaron una manera de transmitir su conocimiento y que cada uno de nosotros nos desarrolláramos como ingenieros, gracias, profesores, gracias, compañeros.

Por último, pero no menos importante, quiero agradecer al CIDESI.

Gracias a todos, este trabajo fue gracias a ustedes.

Fausto Alfonso Angeles Bautista

Resumen del Proyecto.

Ya en las últimas décadas, términos como inteligencia artificial, aprendizaje profundo, redes neuronales han tomado relevancia en el campo de la ingeniería e incluso en campo externos de la ingeniería, como en las producciones cinematográficas donde estos términos son llevados a niveles que la ingeniería aspira. Observar como por medio de cámaras seguridad en calles, centros comerciales y demás establecimientos, localizan a una persona en concreto.

Si bien esto parece irreal, actualmente existen campos de la inteligencia artificial que estudian la capacidad de analizar y comprender imágenes e identificar si la imagen es una persona, algún vehículo o animal; e incluso poder identificar el sexo de la persona, así como si es un infante, el tipo de vehículo. Estos análisis son posibles implementando programas y modelos de Aprendizaje Profundo, son una mejora de redes neuronales haciendo uso de grandes cantidades de datos, *Big Data*.

Este proceso de inferencia consume una gran cantidad de recursos del equipo, y fue hasta 2016 donde una compañía de California, con el nombre de *Movidius*^{TM}, saca al mercado el primer módulo de aceleración de *Deep Learning*. Así bien, el proyecto hace uso de este tipo de módulos en la detección de personas o vehículos para comprobar el rendimiento e iniciar nuevos proyectos, enfocados en visión.

Palabras Clave: *Deep Learning*, *Inferencia*, Redes Neuronales, *Modelos*, *Visión*, *Movidius*^{TM}.

Tabla de Contenido.

Agradecimientos. III

Resumen del Proyecto. IV

Tabla de Contenido. V

Índice de Ilustraciones y Tablas. VIII

Índice de Ilustraciones. VIII

Índice de Tablas. XIII

Capitulo 1: Generalidades de Proyecto. 1

1.1 Introducción del Proyecto. 2

1.2 Antecedentes del Proyecto. 2

1.3 Planteamiento del Problema. 3

1.4 Justificación del Proyecto. 3

1.5 Hipótesis del Proyecto. 4

1.6 Objetivo General del Proyecto. 4

1.7 Objetivos Específicos del Proyecto. 4

1.8 Alcances y limitaciones del Proyecto. 5

Capitulo 2: Fundamentos. 6

2.1 Introducción. 8

2.2 Distribución de Intel® de OpenVINO™ Toolkit. 8

2.2.1 Model Optimizer (Optimizador de Modelo). 9

2.2.2 Inference Engine (Maquina de Inferencia). 12

2.3 Intel® Movidius™ Neuronal Compute Stick (Movidius™ NCS). 14

2.3.1 Movidius™ NCS, Arquitectura del Dispositivo. 15

2.4 Sistemas y Programas Desarrollados. 17

2.4.1 Sistemas C3 y C4. 17

2.4.2 Implementaciones de referencia dentro de OpenVINO™. 18

Capitulo 3: Desarrollo del Proyecto. 20

3.1 Introducción. 22

3.2 Requerimientos del proyecto 22
3.2.1 Descarga del Software del Toolkit OpenVINO™. 22
3.2.2 Descarga del Modelos para implementación. 23
3.2.3 Videos de Prueba. 26
3.2.4 Equipos de Inferencia. 26
3.3 Flujo de Inferencia. 27
3.3.1 Selección de Parámetros e Inicialización de Modelos 27
3.3.2 Proceso de Inferencia 30
3.4 Creación y ejecución de la aplicación. 32
3.4.1 Creación de Aplicación. 32
3.4.2 Ejecución de la aplicación. 34
3.4.3 Interfaz del programa. 35

Capitulo 4: Experimentación y Resultados. 38
4.1 Introducción. 40
4.2 Resultados de Conteo de Personas. 40
4.2.1 Resultados de Conteo de Personas mediante CPU de equipo Dell. 40
4.2.2 Resultados de Conteo de Personas mediante CPU de equipo Zotac. 44
4.2.3 Resultados de Conteo de Personas mediante VPU de equipo Movidius™ NCS. 47
4.2.4 Comparación de Rendimiento en Conteo de Personas. 50
4.3 Resultados de Conteo de Vehículos. 53
4.3.1 Resultados de Conteo de Vehículos mediante CPU de equipo Dell. 53
4.3.2 Resultados de Conteo de Vehículos mediante CPU de equipo Zotac. 57
4.3.3 Resultados de Conteo de Vehículos mediante VPU de equipo Movidius™ NCS. 62
4.3.4 Comparación de Rendimiento en Conteo de Vehículos. 66

Capitulo 5: Conclusiones. 70
5.1 Conclusiones. 72

Capitulo 6: Proyectos Futuros. 74
6.1 Introducción. 76

6.2 Uso de dos o más Movidius NCS. 76
6.3 Uso de Intel Neuronal Compute Stick 2. 76
6.4 Detección con dos modelos, peatones y vehículos. 77
6.5 Creación de modelos y aplicación del Model Optimizer. 78

Apendice A: Glosario. 80
Apendice B: Nomenclatura. 84

Referencias Bibliografía 88

Índice de Ilustraciones y Tablas.

Índice de Ilustraciones.

Ilustración 2.1: Logotipo de OpenVINO™. 8
Ilustración 2.2: Flujo de trabajo de OpenVINO™. 9
Ilustración 2.3: Flujo de trabajo del Model Optimizer. 10
Ilustración 2.4: Modelo Caffe de clasificación - Topología de Resnet269. 11
Ilustración 2.5: Flujo de trabajo de Inference Engine. 12
Ilustración 2.6: Formatos de Puntos Flotantes. 14
Ilustración 2.7: Intel® Movidius™ Neuronal Compute Stick 14
Ilustración 2.8: Arquitectura de Movidius™ NCS. 15
Ilustración 2.9: Arquitectura de la Familia Myriad™ 2. 16
Ilustración 2.10: Myriad™ 2 VPU. 17
Ilustración 2.11: Control Rooms 17
Ilustración 2.12: Centro de Comando, Control, Comunicaciones y Computo. 18
Ilustración 2.13: Solución de notificador de zona restringida. 19
Ilustración 2.14: Solución de sistema recuento de personas. 19
Ilustración 2.15: Solución de contador de estacionamiento 19

Ilustración 3.1: Pagina Web para descargar OpenVINO™ 22
Ilustración 3.2: Instalación del OpenVINO™. 23
Ilustración 3.3: Leyenda de Inicialización del Ambiente de OpenVINO™ 23
Ilustración 3.4: Lista de Modelos. 24
Ilustración 3.5: Modelo de detección de peatones SSD. 25
Ilustración 3.6: Videos de prueba en para inferencia. 26
Ilustración 3.7: Flujo de Inferencia por el API Inference Engine 27
Ilustración 3.8: Proceso de Inferencia. 31
Ilustración 3.9: Descarga de Subsistema de Linux. 32
Ilustración 3.10: Creación de Directorio para Programa. 32
Ilustración 3.11: Generación de Archivos make. 33
Ilustración 3.12: Creación de programa exitosa. 33
Ilustración 3.13: Ejecución de aplicación exitosa. 35
Ilustración 3.14: Interfaz final del programa. 36

Ilustración 4.1: Condiciones Iniciales de la CPU de Equipo Dell 41
Ilustración 4.2: Condiciones de la CPU de Equipo Dell durante Inferencia. 41
Ilustración 4.3: Conteo Ciudad - CPU Dell - 1 Video. 41
Ilustración 4.4: Conteo Ciudad - CPU Dell - 2 Video. 41
Ilustración 4.5: Conteo Ciudad - CPU Dell - 3 Video. 41
Ilustración 4.6: Conteo Ciudad - CPU Dell - 4 Video 41
Ilustración 4.7: Conteo Metro - CPU Dell - 1 Video. 42
Ilustración 4.8: Conteo Metro - CPU Dell - 2 Video. 42
Ilustración 4.9: Conteo Metro - CPU Dell - 3 Video. 42
Ilustración 4.10: Conteo Metro - CPU Dell - 4 Video. 42
Ilustración 4.11: Conteo Mall - CPU Dell - 1 Video. 43
Ilustración 4.12: Conteo Mall - CPU Dell - 2 Video. 43
Ilustración 4.13: Condiciones Iniciales de la CPU de Equipo Zotac. 44
Ilustración 4.14: Condiciones de la CPU de Equipo Zotac durante Inferencia. 44
Ilustración 4.15: Conteo Ciudad - CPU Zotac - 1 Video. 44
Ilustración 4.16: Conteo Ciudad - CPU Zotac - 2 Video. 44
Ilustración 4.17: Conteo Ciudad - CPU Zotac - 3 Video. 45
Ilustración 4.18: Conteo Ciudad - CPU Zotac - 4 Video. 45
Ilustración 4.19: Conteo Metro - CPU Zotac - 1 Video. 45
Ilustración 4.20: Conteo Metro - CPU Zotac - 2 Video. 45
Ilustración 4.21: Conteo Metro - CPU Zotac - 3 Video. 45
Ilustración 4.22: Conteo Metro - CPU Zotac - 4 Video. 45
Ilustración 4.23: Conteo Mall - CPU Zotac - 1 Video. 46
Ilustración 4.24: Conteo Mall - CPU Zotac - 2 Video. 46
Ilustración 4.25: Condiciones de la CPU de Equipo Zotac durante Inferencia de un video en VPU. 47
Ilustración 4.26: Condiciones de la CPU de Equipo Zotac durante Inferencia de dos o más videos en VPU 47
Ilustración 4.27: Conteo Ciudad - VPU - 1 Video. 48
Ilustración 4.28: Conteo Ciudad - VPU - 2 Video. 48
Ilustración 4.29: Conteo Ciudad - VPU - 3 Video. 48
Ilustración 4.30: Conteo Ciudad - VPU - 4 Videos 48
Ilustración 4.31: Conteo Metro - VPU - 1 Video. 49
Ilustración 4.32: Conteo Metro - VPU – 2 Video. 49

Ilustración 4.33: Conteo Metro - VPU - 3 Video. 49
Ilustración 4.34: Conteo Metro - VPU - 4 Video. 49
Ilustración 4.35: Conteo Mall - VPU - 1 Video. 50
Ilustración 4.36: Conteo Mall - VPU - 2 Video. 50
Ilustración 4.37: Grafica Comparativa de rendimiento en Ciudad. 51
Ilustración 4.38: Grafica Comparativa de rendimiento en Metro 52
Ilustración 4.39: Grafica Comparativa de rendimiento en Metro 52
Ilustración 4.40: Conteo Vehículos - CPU Dell - 1 Video. 53
Ilustración 4.41: Conteo Vehículos - CPU Dell - 2 Video. 53
Ilustración 4.42: Conteo Vehículos - CPU Dell - 3 Video. 54
Ilustración 4.43: Conteo Vehículos - CPU Dell - 4 Video. 54
Ilustración 4.44: Conteo Vehículos - CPU Dell - 5 Video. 54
Ilustración 4.45: Conteo Vehículos - CPU Dell - 6 Video. 54
Ilustración 4.46: Conteo Vehículos - CPU Dell - 7 Video. 54
Ilustración 4.47: Conteo Vehículos - CPU Dell - 8 Video 54
Ilustración 4.48: Conteo Vehículos - CPU Dell - 9 Video. 54
Ilustración 4.49: Conteo Vehículos - CPU Dell - 10 Video. 54
Ilustración 4.50: Conteo Vehículos - CPU Dell - 11 Video. 55
Ilustración 4.51: Conteo Vehículos - CPU Dell - 12 Video. 55
Ilustración 4.52: Conteo Vehículos - CPU Dell - 13 Video. 55
Ilustración 4.53: Conteo Vehículos - CPU Dell - 14 Video. 55
Ilustración 4.54: Conteo Vehículos - CPU Dell - 15 Video. 55
Ilustración 4.55: Conteo Vehículos - CPU Dell - 16 Video. 55
Ilustración 4.56: Conteo Vehículos - CPU Dell - 17 Video. 55
Ilustración 4.57: Conteo Vehículos - CPU Dell - 18 Video. 55
Ilustración 4.58: Conteo Vehículos - CPU Dell - 19 Video. 56
Ilustración 4.59: Conteo Vehículos - CPU Dell - 20 Video. 56
Ilustración 4.60: Conteo Vehículos - CPU Dell - 21 Video. 56
Ilustración 4.61: Conteo Vehículos - CPU Dell - 22 Video. 56
Ilustración 4.62: Conteo Vehículos - CPU Zotac - 1 Video 57
Ilustración 4.63: Conteo Vehículos - CPU Zotac - 2 Video 57
Ilustración 4.64: Conteo Vehículos - CPU Zotac - 3 Video 57
Ilustración 4.65: Conteo Vehículos - CPU Zotac - 4 Video 57
Ilustración 4.66: Conteo Vehículos - CPU Zotac - 5 Video 57

Ilustración 4.67: Conteo Vehículos - CPU Zotac - 6 Video..... 57
Ilustración 4.68: Conteo Vehículos - CPU Zotac - 7 Video..... 58
Ilustración 4.69: Conteo Vehículos - CPU Zotac - 8 Video..... 58
Ilustración 4.70: Conteo Vehículos - CPU Zotac - 9 Video..... 58
Ilustración 4.71: Conteo Vehículos - CPU Zotac - 10 Video..... 58
Ilustración 4.72: Conteo Vehículos - CPU Zotac - 11 Video..... 58
Ilustración 4.73: Conteo Vehículos - CPU Zotac - 12 Video..... 58
Ilustración 4.74: Conteo Vehículos - CPU Zotac - 13 Video..... 58
Ilustración 4.75: Conteo Vehículos - CPU Zotac - 14 Video..... 58
Ilustración 4.76: Conteo Vehículos - CPU Zotac - 15 Video..... 59
Ilustración 4.77: Conteo Vehículos - CPU Zotac - 16 Video..... 59
Ilustración 4.78: Conteo Vehículos - CPU Zotac - 17 Video..... 59
Ilustración 4.79: Conteo Vehículos - CPU Zotac - 18 Video..... 59
Ilustración 4.80: Conteo Vehículos - CPU Zotac - 19 Video..... 59
Ilustración 4.81: Conteo Vehículos - CPU Zotac - 20 Video..... 59
Ilustración 4.82: Conteo Vehículos - CPU Zotac - 21 Video..... 59
Ilustración 4.83: Conteo Vehículos - CPU Zotac - 22 Video..... 59
Ilustración 4.84: Conteo Vehículos - CPU Zotac - 23 Video..... 60
Ilustración 4.85: Conteo Vehículos - CPU Zotac - 24 Video..... 60
Ilustración 4.86: Conteo Vehículos - CPU Zotac - 25 Video..... 60
Ilustración 4.87: Conteo Vehículos - VPU - 1 Video. 62
Ilustración 4.88: Conteo Vehículos - VPU - 2 Video. 62
Ilustración 4.89: Conteo Vehículos - VPU - 3 Video. 62
Ilustración 4.90: Conteo Vehículos - VPU - 4 Video. 62
Ilustración 4.91: Conteo Vehículos - VPU - 5 Video. 62
Ilustración 4.92: Conteo Vehículos - VPU - 6 Video. 62
Ilustración 4.93: Conteo Vehículos - VPU - 7 Video. 63
Ilustración 4.94: Conteo Vehículos - VPU - 8 Video. 63
Ilustración 4.95: Conteo Vehículos - VPU - 9 Video. 63
Ilustración 4.96: Conteo Vehículos - VPU - 10 Video. 63
Ilustración 4.97: Conteo Vehículos - VPU - 11 Video. 63
Ilustración 4.98: Conteo Vehículos - VPU - 12 Video. 63
Ilustración 4.99: Conteo Vehículos - VPU – 13 Video..... 63
Ilustración 4.100: Conteo Vehículos - VPU - 14 Video. 63

Ilustración 4.101: Conteo Vehículos - VPU - 15 Video. 64
Ilustración 4.102: Conteo Vehículos - VPU - 16 Video. 64
Ilustración 4.103: Conteo Vehículos - VPU - 17 Video. 64
Ilustración 4.104: Conteo Vehículos - VPU - 18 Video. 64
Ilustración 4.105: Conteo Vehículos - VPU - 19 Video. 64
Ilustración 4.106: Conteo Vehículos - VPU - 20 Video. 64
Ilustración 4.107: Conteo Vehículos - VPU - 21 Video. 64
Ilustración 4.108: Conteo Vehículos - VPU - 22 Video. 64
Ilustración 4.109: Conteo Vehículos - VPU - 23 Video. 65
Ilustración 4.110: Conteo Vehículos - VPU - 24 Video. 65
Ilustración 4.111: Conteo Vehículos - VPU - 25 Video. 65
Ilustración 4.112: Grafica Comparativa de rendimiento en Metro 68

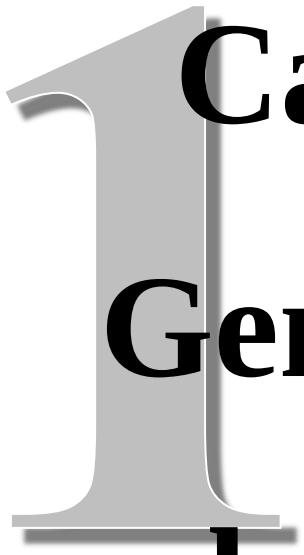
Ilustración 6.1: Inferencia en múltiples dispositivos. 76
Ilustración 6.2: NCS 2. 77
Ilustración 6.3: Comparación de rendimiento de ambos módulos de aceleración neuronal. 77
Ilustración 6.4: Detección de Peatones y Vehículos. 78

Índice de Tablas.

Tabla 2.1: Dispositivos de inferencia para los distintos Plugin. 13
Tabla 2.2: Formatos de modelos. 13
Tabla 2.3: Soporte de formato de modelos. 14

Tabla 3.1: Características de equipos de inferencia. 26
Tabla 3.2: Configuración de capas de salida. 29
Tabla 3.3: Inicialización de Modelos. 29
Tabla 3.4: Comando de Ejecución. 35

Tabla 4.1: Características de equipos de inferencia para generación de resultados. 40
Tabla 4.2: FPS en Ciudad - CPU Dell 42
Tabla 4.3: FPS en Metro - CPU Dell 43
Tabla 4.4: FPS en Mall - CPU Dell 43
Tabla 4.5: FPS en Ciudad - CPU Zotac. 45
Tabla 4.6: FPS en Metro - CPU Zotac 46
Tabla 4.7: FPS en Mall - CPU Zotac 46
Tabla 4.8: FPS en Ciudad - VPU 48
Tabla 4.9: FPS en Metro - VPU 49
Tabla 4.10: FPS en Mall – VPU. 50
Tabla 4.11: Comparación de FPS en Ciudad. 50
Tabla 4.12: Comparación de FPS en Metro. 51
Tabla 4.13: Comparación de FPS en Metro. 52
Tabla 4.14: FPS en conteo de Vehículos desde CPU Dell. 56
Tabla 4.15: FPS en conteo de Vehículos desde CPU Zotac. 61
Tabla 4.16: FPS en conteo de Vehículos desde CPU Zotac (a). 65
Tabla 4.17: FPS en conteo de Vehículos desde CPU Zotac (b). 66
Tabla 4.18: Comparación de FPS en conteo de Vehículos 67



Capitulo 1: Generalidades de Proyecto.

1.1 Introducción del Proyecto.

Si bien sabemos que la inteligencia artificial, es una subdisciplina del campo de la informática que se enfoca en la creación de máquinas que imiten el comportamiento humano; uno de los comportamientos que se busca imitar es la capacidad de analizar y comprender imágenes.

El proyecto comienza con el uso del *Toolkit* (paquete de herramientas) de desarrollo que está abierto para el público interesado en la creación de proyectos enfocados en visión para creación de programas en áreas industriales, de comercio, para ciudades inteligentes, sistema de vigilancia inclusive en el sector salud. Si bien el proyecto está enfocado en la parte de la vigilancia de áreas, en proyectos futuros se puede adaptar a otras áreas.

La implementación del programa se basa en programas de detección del *SSD* (*Single Shoot Multibox Detections*) que tiene como objetivo tener múltiples detecciones dentro de un *frame* mandando las coordenadas del objeto detectado para así dibujar las *Bounding Boxes* y mandar todos estos *frames* y datos de diversos videos a la interfaz de usuario.

1.2 Antecedentes del Proyecto.

El uso de *Movidius™ Neuronal Compute Stick* como un módulo de aceleración de *Deep Learning* para el campo de Visión por Computadora comenzó el primer trimestre del 2018, desde entonces ha tenido una gran aceptación por los interesados en el uso de un dispositivo para el desarrollo de aplicaciones en el ámbito del procesamiento de imágenes.

Cabe destacar que el uso de *Movidius™ NCS* no es un requerimiento para la visión por computadora ya que previamente se ha podido realizar un procesamiento por medio de la *CPU* y/o la *GPU*, pero como comenta la descripción del producto, este dispositivo permite la aceleración del proceso, ya que la unidad está construida con el fin del procesamiento de imágenes y videos a través de redes neuronales mediante populares modelos de entrenamiento que vienen integrados en el paquete de herramientas que provee el fabricante, todo este procesamiento es posible debido a que *Movidius™ NCS* está integrado con una Unidad de Procesamiento de Visión (*VPU*).

OpenVINO™ es el paquete de herramientas que permite el desarrollo de programas con o sin el uso de *Movidius™ NCS*.

1.3 Planteamiento del Problema.

Actualmente se vive en un mundo globalizado en constante movimiento, de constantes cambios, y solo basta detenerse un momento para prestar atención y visualizar diferentes problemas a nuestro alrededor, que con algo de ingenio se logran solucionar.

Diversos sistemas de vigilancia en distintos campos requieren de personal que vigile las imágenes que se presentan en pantallas, para conocer de eventos acontecidos en cierta área o tomar medidas para los acontecimientos de las áreas. Estos problemas son encontrados en áreas donde esté algún circuito cerrado de cámaras de vigilancia, la detección de objetos mediante modelos pre-entrenados permite localizar peatones, vehículos o algún otro objeto dentro de un área del *frame*, tener un análisis visual de estos datos para mejorar la seguridad y servicios, con fines de tener ciudades inteligentes.

El uso de personal resulta ineficiente para monitoreo de sistemas de vigilancia basados en circuitos cerrados de cámaras, ya que existen segundos o incluso minutos donde las pantallas dejan de ser vigiladas y ciertas cosas pueden ser omitidas, que pueden o no carecer de importancia, pero en caso de que estos datos sean importantes presenta un problema dentro de la vigilancia. El problema es resuelto mediante el procesamiento de imágenes por medios computacionales y la clasificación de objetos que se logra por medio de modelos de Aprendizaje Profundo.

1.4 Justificación del Proyecto.

El uso de personal de trabajo en áreas enfocadas al análisis de imágenes presenta un rango de error en el cual existe tiempo donde ciertos cuadros o *frames* del video son perdidos, especialmente cuando se requiere una respuesta rápida y continua.

Haciendo uso de un sistema de monitoreo por detección inteligente, con buenos modelos pre-entrenados podemos generar un conteo en áreas amplias donde exista una aglomeración, ya sea de peatones o vehículos, en donde la Visión por Computadora va a

permitir tener una mejor y más rápida respuesta que la propia visión humana, esto sumado a que cada equipo puede hacer la detección en diversos videos de manera simultánea, con lo anterior reduce el número de personal encargado a tareas de vigilancia.

Con esto último se pretende que diferentes áreas integren el programa en su sistema de vigilancia, para que el personal de vigilancia se enfoque más en áreas con movimientos o mayor aglomeración.

1.5 Hipótesis del Proyecto.

Las características que enuncia *Intel*® sobre su producto *Movidius*^{TM} *Neuronal Compute Stick* pretenden ser usada en equipos con bajas especificaciones que cuenten con un puerto *USB*, este producto de *Intel*® debe ser usado con un formato de modelos de Deep Learning de menor precisión, ya que únicamente es compatible con tal formato.

En base a ello se espera que el tiempo de respuesta de un equipo con modelos de menor precisión en *Movidius*^{TM} *NCS* sea igual o incluso superior al tiempo de respuesta de equipos con mejores especificaciones técnicas con modelos de mayor precisión. En los resultados del proyecto se va a aprobar o desaprobar la hipótesis según el comportamiento del programa con el equipo con el *NCS* y el equipo sin el NCS.

1.6 Objetivo General del Proyecto.

Realizar un programa que aplique visión por computadora y modelos de aprendizaje profundo que permita el conteo de manera inteligente de objetos animados o inanimados en múltiples videos de manera simultánea para solución de fines específicos del área.

1.7 Objetivos Específicos del Proyecto.

- Ejecución de un programa en lenguaje *C++* compatible con modelos *Deep Learning* de Detección.
- Añadir Single *Shoot Multibox Detecction* (*SSD*) en el programa con el fin de contar las detecciones.
- Visualizar en la ventana del programa los *Frames* de los diferentes videos.

- Seleccionar el *VPU Myriad* incluido en *Movidius™ NCS* como dispositivo para realizar la inferencia.
- Realizar la inferencia en el *CPU* de un par de equipos de cómputo para comparar el rendimiento de ambos dispositivos.

1.8 Alcances y limitaciones del Proyecto.

La práctica industrial únicamente pretende implementar un programa para el conteo inteligente mediante *Movidius™ Neuronal Compute Stick* con fines de reducir la carga computacional del trabajo de inferencia de los modelos en la *CPU* de un equipo de cómputo. Así mismo no se busca la creación de modelos *IR* (*Intermediate Representation*), si no únicamente hacer uso de los modelos *IR* que facilita *Intel*®.

Capitulo 2:

Fundamentos.

2.1 Introducción.

Este capítulo presenta los diversos temas que se centran principalmente en el *toolkit* de *Intel*®, ya que en este paquete de herramientas se presenta el *software* que permite el uso de *Movidius*^{TM} *NCS* y del *CPU* de un equipo de cómputo como *hardware* para el desarrollo de programas de visión.

2.2 Distribución de Intel® de OpenVINO™ Toolkit.

El *toolkit* de *OpenVINO*^{TM}, ilustración 2.1, despliega un conjunto de aplicaciones y soluciones que buscan emular la visión humana. Basada en *Convolutional Neuronal Networks* (*CNN*), extendiendo de este paquete a la visión por Computadora (*CV*) a través de hardware de *Intel*®, maximizando el rendimiento de emulación de los equipos.

Ilustración 2.1: Logotipo de OpenVINO^{TM}.

Las siglas *VINO* vienen de *Visual Inferencing and Neuronal Network Optimization* (Inferencia Visual y Optimización de Redes Neuronales). Este *toolkit* tiene como finalidad los siguientes puntos:

- Habilitar el *Deep Learning Inference on the Edge* basada en *CNN*.
- Soporte de ejecución a través de *Hardware* como ***CPU*** potenciados por procesadores *Intel*®, *Intel*® *Integrated Graphics*, *Intel*® *FPGA*, ***Intel*® *Movidius*^{TM} *Neuronal Compute Stick***, *Intel*® *Neuronal Compute Stick 2* y *Intel*® *Vision Accelerator Design with Intel*® *Movidius*^{TM} *VPUs*.
- Librerías para funciones de visión por computadora, así como *Kernels* de pre-optimización.
- Incluye leguaje abierto para estándares de visión por computadora, como son *OpenCV*, *OpenCL*^{TM}, y *OpenVX*.

El *toolkit* de *OpenVINO™* está conformado básicamente por dos herramientas, denominadas *API* (*Application Programming Interface*) para el desarrollo de nuevos programas, que no están enfocadas en el entrenamiento de modelos de *Deep Learning*, *OpenVINO™* está enfocada en la optimización de modelos pre-entrenados de *Deep Learning* y en el proceso de inferencia con estos modelos, como esquematiza la ilustración 2.2.

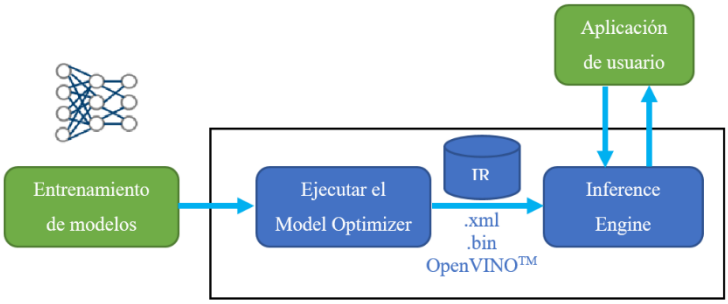


Ilustración 2.2: Flujo de trabajo de OpenVINO™.

Aprendiendo a ocupar estas dos herramientas, tanto el *Model Optimizer* como la *Inference Engine* y cumpliendo con el requerimiento de tener algún modelo de *Deep Learning* entrenado, se podrán generar aplicaciones para el usuario donde tendremos los resultados en tiempo real de la *Inference Engine*.

2.2.1 Model Optimizer (Optimizador de Modelo).

El *Model Optimizer* es una herramienta a través de líneas de comando para diferentes plataformas que facilita la transición entre el entrenamiento y el entorno de desarrollo de *OpenVINO™*, ajustando modelos estáticos en modelos de *Deep Learning* para su óptima ejecución dentro del dispositivo seleccionado para el proceso de inferencia en la *Inference Engine*.

El proceso del *Model Optimizer* da por hecho que uno ya cuenta con modelo de red ya entrenado usando suporte de *Deep Learning Framework*. La ilustración 2.3 muestra la típica esquematización del flujo de trabajo para desarrollo de un modelo de *Deep Learning* entrenado.

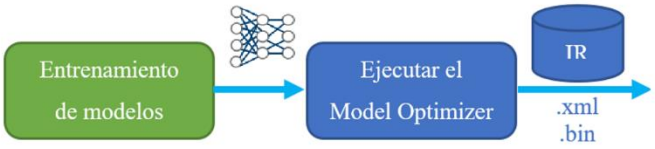


Ilustración 2.3: Flujo de trabajo del Model Optimizer.

Como se plantea en la ilustración 2.3, el *Model Optimizer* produce un *Intermediate Represenation* (*IR*) de la red, este modelo de representación intermedia puede ser leída, cargada e inferida por la *Inference Engine.* La representación intermedia es un par de archivos que describen el modelo:

- .xml: Archivo de Topología – Describe la Topología de red.
- .bin: Archivo de datos entrenados – Contiene el peso en datos de binario.

2.2.1.1 Optimización de Modelos de Deep Learning en Modelos IR.

El entrenamiento de modelos es hecho típicamente en centros de datos de gama alta, usando populares *framework* de entrenamiento como *Caffe*, *TensorFlow* y *MXNet*. El *Model Optimizer* convierte estos modelos entrenados con sus propios formatos originales en modelos *IR* que describen su topología.

Dentro del *Model Optimizer*, hay varias optimizaciones al comportamiento de esta herramienta que son independientes del dispositivo usado. Para explicar mejor la optimización de los modelos observemos la ilustración 2.4.

En esta ilustración observamos el modelo de la izquierda es el modelo original, mientras que el de la derecha es una conversión resultado del proceso del *Model Optimizer*, con capas *BatchNorm* y *ScaleShift* (conceptos usados dentro del *Model Optimizer*) fundidas dentro de una convolución de pesos en vez de estar constituidas por capas separadas.

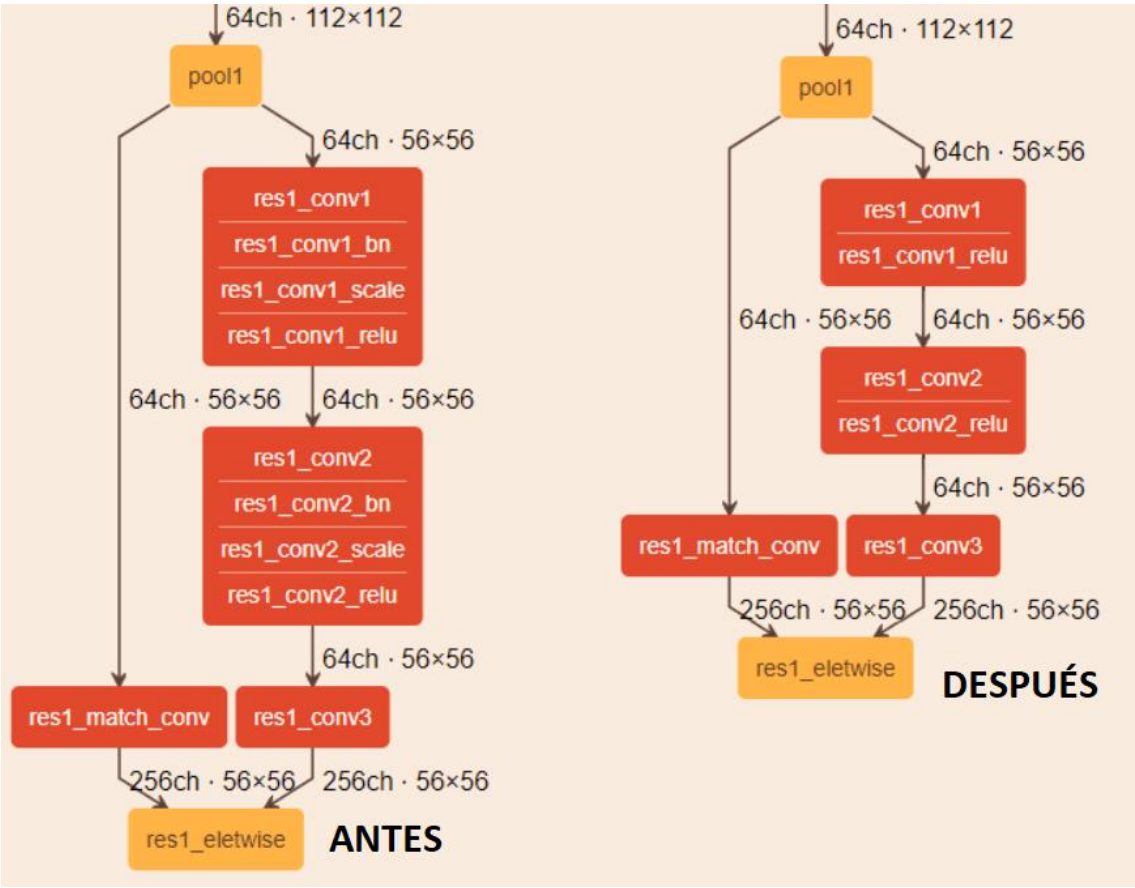


Ilustración 2.4: Modelo Caffe de clasificación - Topología de Resnet269.

Las características en el uso de modelos son:

- **Parámetros de escala/valor de la imagen:** La correcta configuración de los parámetros de escala/valor de la imagen de entrada coincidan con el *Model Optimizer* cuando se realice un procesamiento previo. Esto permite que la herramienta se optimice y así acelere el proceso dentro de la *Inference Engine*.
- **Entrada RGB vs BGR**: Los modelos de red manejan entradas *BGR*, y las entradas de imágenes normalmente manejan el formato *RGB* pero el *Model Optimizer* hace el cambio desde la primera convolución, evitando que el desarrollador haga cambios en la imagen durante el proceso de inferencia.
- **Resultados de Precisión de modelos IR**: Los modelos adquieren cierta precisión por ejemplo *FP16* o *FP32*, lo cual afecta directamente a su rendimiento. Esto se explicará de mejor manera en el tema 2.2.2.1 ya que involucra con los *Inference Engine Plugin*.

2.2.2 Inference Engine (Máquina de Inferencia).

Después de haber hecho uso del *Model Optimizer* para la creación del modelo de *Intermediate Representatión* (*IR*), se usa la *Inference Engine* para inferir en los datos de entrada que ingresamos desde la aplicación, como se esquematiza en la ilustración 2.5.

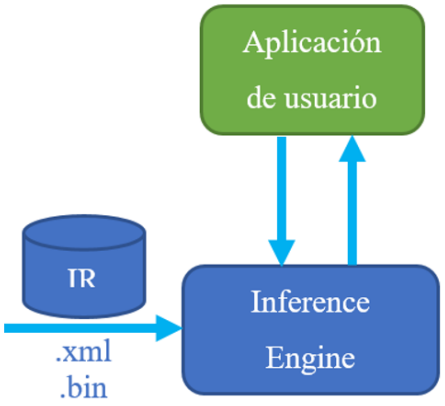


Ilustración 2.5: Flujo de trabajo de Inference Engine.

La *Inference Engine* es una librería de *C++* con un conjunto de clases para inferir con los datos de entrada, que vienen siendo las imágenes, y conseguir los resultados. La librería de *C++* proporciona un *API* para la lectura de los modelos de representación intermedia, así como el ajuste de entrada y salida de los formatos de datos, y la ejecución de los modelos en los dispositivos.

La *Inference Engine* hace uso de una arquitectura para acceder a dispositivos externos al equipo como *Movidius™ NCS* o al mismo *CPU*, recibiendo el nombre *Inference Engine Plugin*. *Inference Engine Plugin* es un componente de software que contiene una implementación complementos para la inferencia en ciertos dispositivos de hardware de *Intel*®: *CPU*, *GPU*, *VPU*, *FPGA*, etc. Cada *Plugin* se implementa dentro de un *API* unificado que adicionalmente proporciona que se pueda trabajar con diversos *Plugin* sin alterar el programa.

2.2.2.1 Optimización específica para los dispositivos Plugin de Inferencia.

La *Inference Engine* soporta diferentes dispositivos (*CPU*, *GPU*, *Intel® Movidius™ Myriad 2 VPU*, *Intel® Movidius™ Myriad™ X VPU*, *Intel® Vision Accelerator Design*

with Intel® MovidiusTM vision Processing Units (*VPU*) y *FPGA*), y cada uno de ellos corresponde a un *plugin*.

Para desarrollo del proyecto únicamente nos centraremos en dos *plugin* con los cuales se trabaja en el desarrollo del proyecto, mediante sus librerías que contienen implementaciones complementarias para la inferencia en su dispositivo en particular. La tabla 2.1 contiene los *plugins* usados.

Plugin	Tipo de Dispositivo
CPU	*Procesadores Intel® Xeon®* con *Intel® AVX2* y *Intel®AVX512, Procesadores Intel® Core™ Processors con Intel ® AVX2, Procesadores Intel® Atom® Processors con Intel® SSE.*
MYRIAD	*Intel® Movidius™ Neuronal Compute Stick* impulsado por *Intel® Movidius™ Myriad™ 2, Intel® Neuronal Compute Stick 2* impulsado por *Intel® Movidius™ Myriad™ X*

Tabla 2.1: Dispositivos de inferencia para los distintos Plugin.

La *Inference Engine* puede inferir modelos en distintos formatos. Este tema proporciona información para la configuración óptima para cada *plugin*, la tabla 2.2 muestra la descripción de los formatos con los que trabajamos.

Termino	Tipo de Dispositivo
Formato FP32	*Formato Punto Flotante de Single-precision.*
Formato FP16	*Formato Punto Flotante de Half-precision.*

Tabla 2.2: Formatos de modelos.

Estos formatos los podemos explicar de mejor manera en base a la ilustración 2.6, donde observamos cómo están conformados estos dos formatos, por un Bit que determina el signo del dato, otros pocos al exponente y el resto, son los bits significativos también llamado mantisa o fracción.

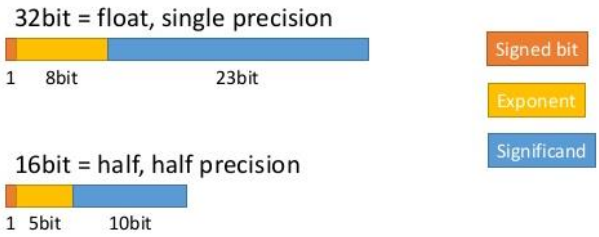


Ilustración 2.6: Formatos de Puntos Flotantes.

Si bien ya tocamos los temas de los formatos de los modelos y de los dispositivos de *plugin* a los que soporta en el proceso de inferencia, vemos la relación en la tabla 2.3.

Plugin	FP32	FP16
CPU	*Soportado*	*No Soportado*
MYRIAD	*No Soportado*	*Soportado*

Tabla 2.3: Soporte de formato de modelos.

2.3 Intel® Movidius™ Neuronal Compute Stick (Movidius™ NCS).

Movidius™ Neuronal Compute Stick, ilustración 2.7, un módulo para la aceleración de aprendizaje profundo, *Deep Learning*, contenido en un dispositivo estándar *USB 3.0*, permitiendo una comunicación más veloz de la información. Usado en aplicaciones de inferencia para prototipos de inteligencia artificial sin necesidad de estar conectado a la red.

Ilustración 2.7: Intel® Movidius™ Neuronal Compute Stick

Este dispositivo está impulsado por una Unidad de Procesamiento por Visión, con nombre de *Intel® Movidius™ Myriad™ 2*, tiene como característica ser líder de clase,

además cuenta con un ultra bajo consumo. Con soporte de populares redes neuronales de *framework* como *Caffe* y *Tensorflow* o con compilación propia comúnmente mediante los tipos de capa.

2.3.1 Movidius™ NCS, Arquitectura del Dispositivo.

Movidius™ NCS está impulsada por la *Vision Processing Unit Intel® Movidius™ Myriad™ 2*, contando con un almacenamiento dinámico de bajo consumo de *4Gb* de *LPDDR3 DRAM* con el cual *MovidiusTM NCS* está equipada

Este dispositivo es usado al ser conectado a una maquina anfitrión (*Host*) usando la interfaz USB del dispositivo, el estándar *USB 3.0* suele usarse en dos modos, súper velocidades de 5 *Gbps* o altas velocidad 480 *Mbps*. Esta arquitectura la podemos observar a grandes rasgos en la ilustración 2.8.

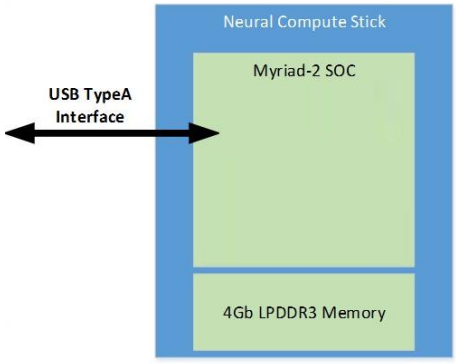


Ilustración 2.8: Arquitectura de Movidius™ NCS.

2.3.1.1 Intel® Movidius™ Myriad™ 2 Arquitectura del VPU.

La parte fundamental del *Movidius™ NCS* es su *VPU*, ya que es el primer procesador de visión que permanece siempre activo. Ofrece visión artificial de alto rendimiento en entornos limitados respecto a fuentes de alimentación.

La reputación que ha generado la familia de procesadores *Myriad™ 2* es por su bajo consumo y alto rendimiento, procesadores que están transformando las posibilidades del dispositivo. *Myriad™ 2* entrega a los desarrolladores acceso inmediato al avanzado núcleo de procesamiento de visión, mientras les permite desarrollar proyectos patentados que proporcionen una verdadera diferencia entre la competencia del mercado.

Profundizando en las características que mejor definen a este procesador son:

- **Diseño de ultra bajo consumo**: Para dispositivos móviles y conectados donde la vida útil de la batería es crítica, *Myriad™ 2* de *Intel*® proporciona una forma de combinar aplicaciones de visión avanzada en un perfil de bajo consumo. Esto permite nuevas aplicaciones de visión en pequeños factores que antes no eran posibles.
- **Procesador de alto rendimiento:** Acercar las tecnologías de visión a las capacidades de visión humana con la conexión del dispositivo. *Movidius™ Myriad™ 2* de *Intel*® permite aplicaciones de visión avanzadas que son con los procesadores convencionales.
- **Arquitectura programable**: *Myriad™ 2* es flexible para que los desarrolladores implementen proyectos patentados y diferentes, impulsados por este procesador. La biblioteca optimizada con la cual cuenta brinda a los fabricantes del dispositivo la posibilidad de trabajar con la arquitectura desde un nivel nuclear, sin la posibilidad de duplicar esa arquitectura, que muestra la ilustración 2.9.

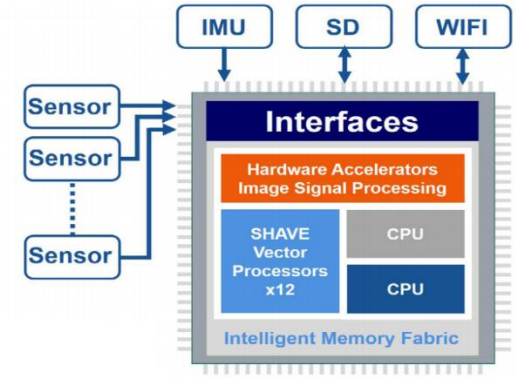


Ilustración 2.9: Arquitectura de la Familia Myriad™ 2.

- **Tamaño de área de la huella**: Para ahorrar espacio dentro e dispositivos móviles, portátiles e integrados, la *VPU Myriad™ 2* de *Intel*® ha sido diseñado de un tamaño pequeño, como lo muestra la ilustración 2.10, de tal manera que este procesador puede ser integrado fácilmente en productos existentes.

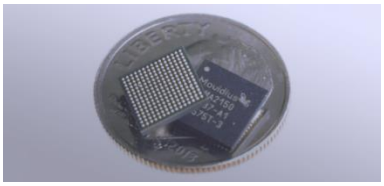


Ilustración 2.10: Myriad™ 2 VPU.

- **Detalles adicionales del Chip**: La arquitectura de Intel® MovidiusTM MyriadTM 2 comprende un conjunto de interfaces, un conjunto de aceleradores de imagen/visión mejorados, un grupo de 12 procesadores *VLIW* vectoriales llamados procesadores *SHAVE*s, estos procesadores son usados para la aceleración de redes neuronales por la ejecución de partes de las redes neuronales en paralelo, y una estructura de memoria inteligente que reúne los recursos de procesamiento para permitir una eficiencia energética en el dispositivo.

2.4 Sistemas y Programas Desarrollados.

Ya en 2019 han sido creado sistema de monitoreo, así como se han ido implementando programas mediante el software de *OpenVINO™*, esto nos permite conocer las posibilidades del producto con el que se trabaja.

2.4.1 Sistemas C3 y C4.

Los Centros de Monitoreo o Centros de Control, también llamados *Control Rooms*, ilustración 2.11, son espacios en donde una serie de operadores monitorean datos o video para reaccionar de manera inmediata ante alarmas, emergencias o eventos haciendo uso de sistemas de información para agilizar su interpretación. [14]

Ilustración 2.11: Control Rooms

Un ejemplo de Centro de Monitoreo de Seguridad es un Centro de Comando, Control, Comunicaciones y Cómputo, también llamado C4, ilustración 2.12. Estos Centros pueden operar en 3 niveles: municipal, estatal y federal, y se mantienen en operación 24/7 bajo normas y procedimientos estrictos que los llevan a reducir al máximo sus tiempos de atención a emergencias. [14]

Ilustración 2.12: Centro de Comando, Control, Comunicaciones y Computo.

Para lograr esto se suministran equipos de altas especificaciones, tanto audiovisuales como de Seguridad Física, soportados por un equipo técnico especializado siempre dispuesto a proporcionar el mantenimiento que se requiera. [14]

2.4.2 Implementaciones de referencia dentro de OpenVINO™.

OpenVINO™ ha permitido a los desarrolladores la implementación de programas en el campo de la visión por computadora, estas soluciones que presenta por medio de su módulo de aceleración de inferencia *Movidius™ NCS*:

- **Notificador de Zona Restringida**: Monitoreo de un área de trabajo para alertar cuando algún trabajador entre en una zona de riesgo, como se muestra en la ilustración 2.13.

Ilustración 2.13: Solución de notificador de zona restringida.

- **Sistema de Recuento de Personas**: Detección de personas en un área y determinación de personas por *frame*, ilustración 2.13.

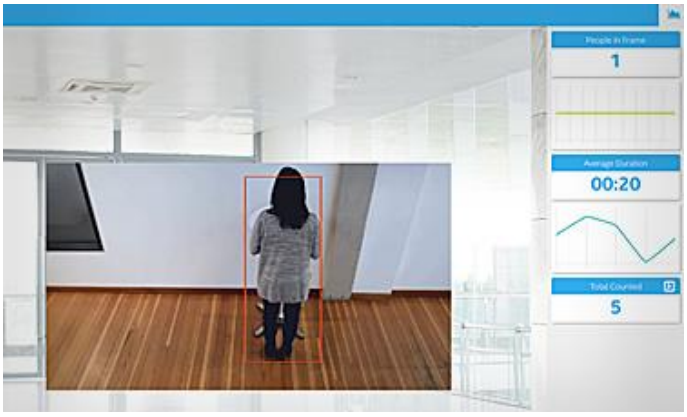


Ilustración 2.14: Solución de sistema recuento de personas.

- **Contador de Estacionamiento**: Monitoreo de áreas de estacionamiento, así como el conteo de vehículos que entran al área y que salen de la misma, ilustración 2.14.

Ilustración 2.15: Solución de contador de estacionamiento

3 Capitulo 3: Desarrollo del Proyecto.

3.1 Introducción.

Este capítulo busca llevar punto a punto la metodología que permite comprender el proceso necesario para poder ejecutar el programa, así como realizar cambios en el mismo para trabajos futuros.

3.2 Requerimientos del proyecto

Si bien en el capítulo dos contiene dos temas, el software y parte del hardware requerido para la implementación del programa de visión, en este tema se entra en detalles sobre la adquisición de estos *softwares* principalmente.

3.2.1 Descarga del Software del Toolkit OpenVINO™.

La comienza con el software de *OpenVINO™*, que se encuentra disponible de manera gratuita en el enlace, https://software.intel.com/en-us/openvino-toolkit/choose-download, ilustración 3.1, para diferentes sistemas operativos, tal como es *Linux* y *Windows*.

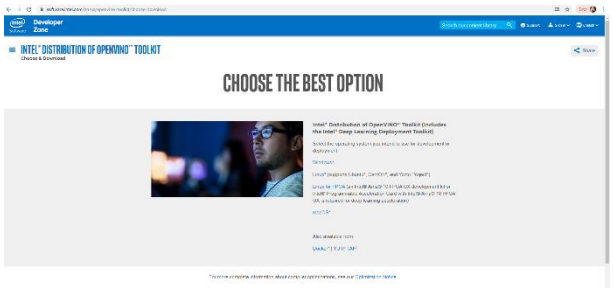


Ilustración 3.1: Pagina Web para descargar OpenVINO™.

En proyecto hace uso del *toolkit* en el sistema operativo de *Linux,* debido a que el *toolkit* en Windows tiene un menor tiempo de desarrollo y hace uso de 2 programas que en Linux ya vienen incluidos, los programas son *Visual Studio 2015* o *2017*, *Python* y *CMake*.

Una vez seguido los pasos, como los de la ilustración 3.2, el equipo tiene a su disposición el *toolkit* de desarrollo. Este software hace uso de la terminal del equipo, así que al hacer uso del programa requerimos iniciar el ambiente mediante la activación del ambiente con la siguiente línea de comando:

```
source ~/intel/openvino/bin/setupvars.sh
```

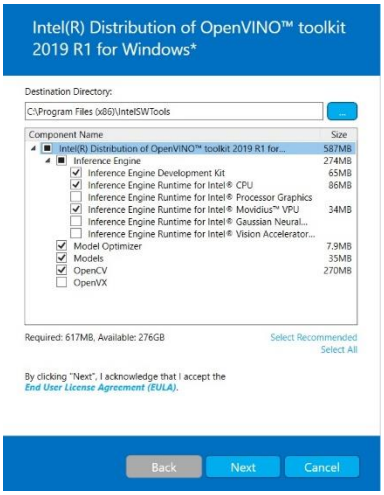


Ilustración 3.2: Instalación del OpenVINO™.

Con esta línea de comando aparece el enunciado de la ilustración 3.3 que indica que podemos correr programas haciendo uso del *Model Optimizer* y de la *Inference Engine* al momento de ejecutar el comando del programa.

Ilustración 3.3: Leyenda de Inicialización del Ambiente de OpenVINO™.

Esto cumple el primero de los requisitos para poder trabajar con el software.

3.2.2 Descarga del Modelos para implementación.

Como segundo punto de requerimiento del proyecto, requiere tener modelos de detección de peatones y vehículos, respetando lo indicado en la tabla sobre los formatos de los modelos que soportan los dispositivos donde se realiza el proceso de inferencia de modelos. Con conexión a internet se descarga los modelos de detección *IR* (*Intermediate Representation*), ya que son con estos modelos con los que podemos trabajar en la *Inference Engine*, de descargar modelo en alguna otra representación de *framework* de las siguientes extensiones:

- *.caffemodel* – Modelos de *Caffe Framework.*
- *.pb* – Modelos de *TensorFlow Framework.*
- *.params* – Modelos de *MXNet Framework.*
- *.onnx* – Modelos de *ONNX Framework.*
- *.nnet* – Modelos de *Kaldi Framework.*

En caso de que el modelo pre-entrenado haga uso de estar extensiones, se procede a ocupar el *Model Optimizer* para la optimización de estos modelos y poder trabajar con ellos la implementación del programa. Para mayor información sobre esta herramienta, el siguiente enlace contiene una guía:

- https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html

El enfoque del proyecto se encuentra en la inferencia, mas no en la creación de modelos de entrenamiento, así que desde la terminal se ingresa las siguientes líneas:

```
cd ~/intel/openvino/deployment_tools/tools/model_downloader
```

```
python3 downloader.py --print_all
```

Con estos comandos, la terminal despliega una lista de modelos disponibles, la ilustración 3.4 es un parte de la lista, ya sea modelos de clasificación, detección, segmentación, entre otros *OpenVINO™* cuenta una fácil descarga de estos modelos.

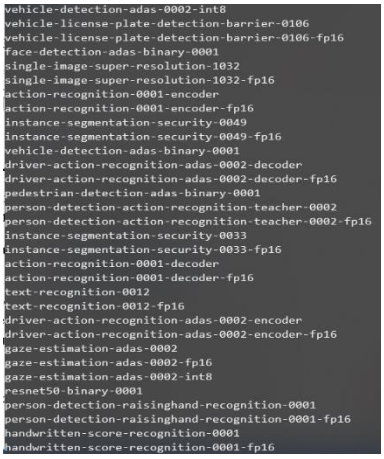


Ilustración 3.4: Lista de Modelos.

El proyecto utiliza dos modelos de detección, y como parte de los objetivos es la comparación de rendimiento en diferentes dispositivos *plugin*, el *Movidius*^{TM} *Myriad*^{TM} *2* (el *VPU*) y la *CPU*, descargamos dos formatos por cada modelo de detección, el formato *FP16* para el *VPU* y el *FP32* para la *CPU* como indica la tabla 2.2, dando un total de 4 modelos que debe contar nuestro equipo.

- pedestrian-detection-adas-0002
    - pedestrian-detection-adas-0002-fp16 y pedestrian-detection-adas-0002-fp32
- vehicle-detection-adas-0002
    - vehicle-detection-adas-0002-fp16 y vehicle-detection-adas-0002-fp32

Todos éstos modelos son denominados modelos de detección *SSD* (*Single Shoot Multibox Deteccion*), ilustración 3.4, como su nombre lo indica, permiten tener diferentes detecciones dentro del mismo *frame*, característica que nos va permitir tener el conteo en el programa, cada detección es indicada mediante dos coordenadas, las cuales sirven para dibujar las *bounding boxes*, que son los recuadros que indican donde se encuentra el objeto, que en fines del proyecto son peatones o vehículos.

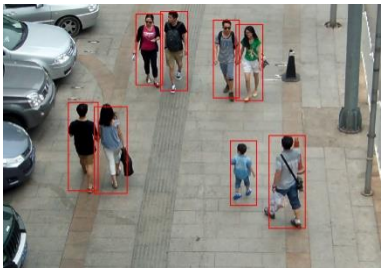


Ilustración 3.5: Modelo de detección de peatones SSD.

Estos modelos al ser descargados cuentan con dos archivos cada uno, que corresponden al modelo *IR*, un archivo para la descripción de la topología del modelo, el archivo .xml y por otro lado el archivo de los pesos binario, el archivo .bin, estos archivos son descargados mediante los siguientes comandos en la terminal:

```
python3 downloader.py –name xxxxxx-detection-adas-0002-fpxx
```

3.2.3 Videos de Prueba.

Las librerías incluidas en el *toolkit* de *OpenVINO*^{TM} facilitan la transformación de las imágenes o videos en datos binarios, para ser procesado dentro del ambiente de *OpenVINO*, y como parte de los objetivos, la implementación de una inferencia en múltiples videos se realizará con videos de prueba de la siguiente página:

- https://www.pexels.com/videos

Los videos de esta página permiten ver el comportamiento de los modelos seleccionados en diferentes ambientes de iluminación, ya sea si analizamos el conteo de personas o vehículos en la ciudad, campo, sistema metro, centros comerciales, ejemplo de video de prueba en la ilustración 3.6.

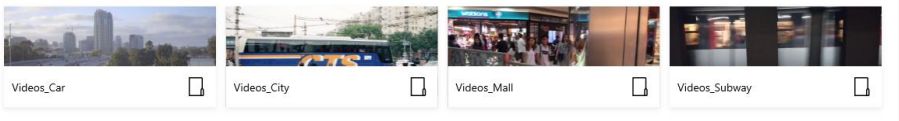


Ilustración 3.6: Videos de prueba en para inferencia.

3.2.4 Equipos de Inferencia.

La *Inference Engine*, parte del *Toolkit*, realiza el proceso de inferencia en distintos equipos, el proyecto realiza este proceso en tres equipos diferentes para posteriormente realizar comparaciones, las especificaciones de estos equipos los encontramos en la tabla 3.1.

Ilustración	Marca	Modelo	Procesador	RAM
	Zotac	ZBOX CI543 Nano	Intel Core i5-6300U-2.50 GHz	8Gb
	Dell	Inspirion 14z 5423	Intel Core i7-3517U - 1.90GHz	8Gb
	Movidius	Movidius NCS	Movidius Myriad 2	4 Gb

Tabla 3.1: Características de equipos de inferencia.

3.3 Flujo de Inferencia.

Cumpliendo los requerimientos para la ejecución del programa, se inicia el proceso de inferencia, el cual se radica en uso del *API Inference Engine* del *Toolkit* de *OpenVINO*^{TM}, la ilustración 3.7 muestra los pasos principales dentro del flujo de inferencia.

Si bien el proceso de la *Inference Engine* es establecido en diferentes pasos, se analiza en dos partes, la primera corresponde a la inicialización del modelo y la configuración de los parámetros y la segunda parte comienza con la solicitud de Inferencia.

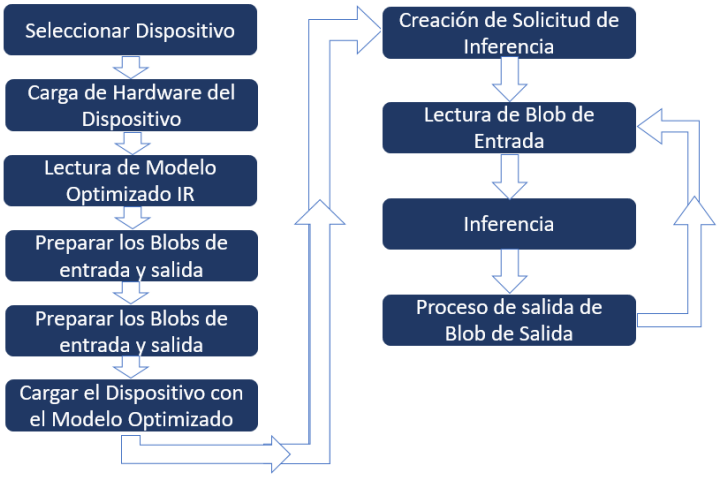


Ilustración 3.7: Flujo de Inferencia por el API Inference Engine

3.3.1 Selección de Parámetros e Inicialización de Modelos

Es por medio de la que la librería núcleo del *API*, `libinference_engine.so` que implementa y carga el modelo de representación intermedia (*IR*), y entra en funcionamiento en el dispositivo especifico *Plugin* seleccionado.

1) **Crear la *Inference Engine Core*** para acceder a los *plugins* internos del equipo, como es el *CPU* o si el equipo ya cuanta dentro de su arquitectura una Unidad de Procesamiento de Visión, en el proyecto este no el caso, ya que la *VPU* se encuentra dentro del *Movidius*^{TM} *NCS*.
2) **La creación de un lector IR** para la lectura del modelo, eso quiere decir que va leer el archivo de extensión "*.xml*", que indica la topología del modelo y el archivo de pesos binarios de extensión "*.bin*".

La función *CNNNetReader*, lee los archivos *IR* dentro un objeto de la clase *CNNNetwork*. Esta clase representa la red dentro de la memoria del equipo.

3) **La configuración de entradas y salidas es una solicitud**, volviendo al uso de las clases con las cuales cuenta el API, se adquiere la información de las entradas y salidas por estos métodos *getNetwork*, *getInputsInfo* y *getOutputsInfo*.

   Opcionalmente colocar el número de formato del modelo (la precisión del modelo) y las capas en la memoria. Esto para tener las correctas configuraciones soportadas.

   La posibilidad de entradas de cualquier tamaño. Para esto se realiza un cambio de tamaño en cada entrada con un algoritmo de reajuste de tamaño a la configuración deseada dentro de la información de entrada.

   Otras de las configuraciones, es el formato de color soportado. Si bien, lo normal es que la *Inference Engine* asuma que el formato color sea *BGR* y si es esto la conversión es desactivada. Los formatos de entrada soportados de la *Inference Engine* son los siguientes:

   - *RGB* a *BGR*
   - *RGBX* a *BGR*
   - *BGRX* a *BGR*
   - *NV12* a *BGR*

   Donde *X* es un canal que pasa a ser ignorado durante la inferencia. Para activar la conversión, colocar el formato deseado para cada entrada en la información apropiada de entrada.

   Si se corre la inferencia para múltiples imágenes a la vez, es posible la construcción en *batch* con función de pre-procesamiento.

   Si este paso es saltado durante la contracción de la aplicación los valores por defecto son:

   - No existe algoritmo de redimensión para configuración de entrada.
   - El formato de color es *RAW*, que indica que la aplicación no requiere conversión de color
   - La precisión de entrada es de *FP32*.
   - Las capas de entrada es *NCHW*.

- Las capas de salida dependen de su número de dimensiones, como indica la tabla 3.2.

Numero de dimensiones	**5**	**4**	**3**	**2**	**1**
Capa	NCDHW	NCHW	CHW	NC	C

Tabla 3.2: Configuración de capas de salida.

4) **Cargar el modelo dentro del dispositivo** requiere de la función *LoadNetwork*. Esto crea una red ejecutable desde un objeto de red. La red ejecutable es asociada con un único dispositivo *hardware*. Esto es posible para crear tantas redes como se necesitan y usarla de manera simultánea (esto limitado por los recursos del *hardware*). El segundo parámetro es la configuración para el *plugin*. Este es mapeo por pares: (nombre del parámetro, valor del parámetro). Escoger un dispositivo con soporte necesario, tabla 2.3 para la compilación de la red en el dispositivo.

Lo realizado en el proyecto que hace énfasis en la selección de parámetros lo podemos resumir en la ilustración 3.8 Ya una vez configurados los parámetros, ya lo podemos implementar los modelos de detección a los diferentes *hardware* para comenzar la inferencia.

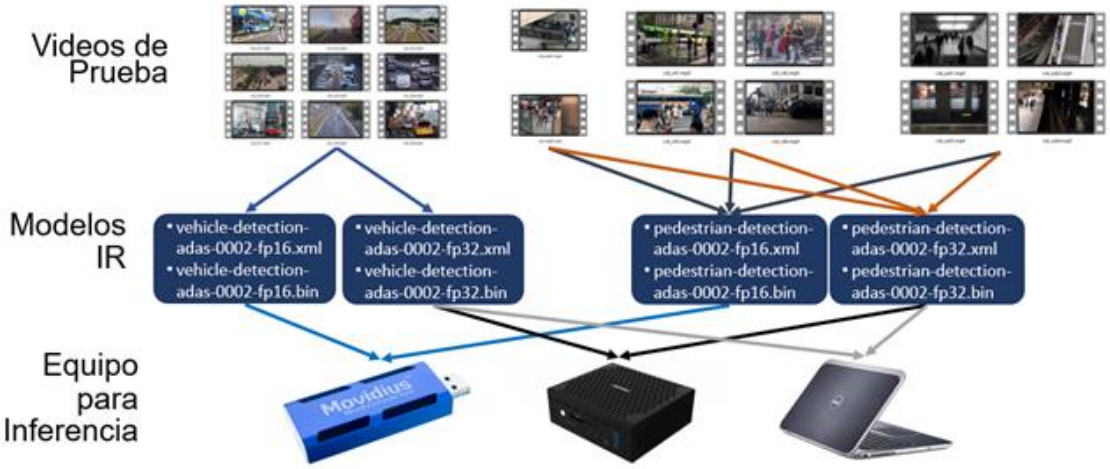


Tabla 3.3: Inicialización de Modelos.

3.3.2 Proceso de Inferencia

Ya teniendo una correcta configuración, la ejecución del programa del programa procede sin error de parámetros de modelo, el software realiza la el proceso de inferencia mediante los siguientes pasos:

5) **Creación de la solicitud de Inferencia** es con la función *CreateInferRequest*, permitiendo el inicio de la inferencia.
6) **Preparación de datos entrada**, este paso requiere que analicemos que tipo de aplicación queremos, ya que se pueden generar tres diferentes maneras de dar solución a la aplicación, las cuales son:
    a. Óptimo funcionamiento para una red.
    b. Óptimo funcionamiento para cascadas de redes (la salida de una red es la entrada de otra)
    c. Óptimo funcionamiento para manejo de ROI (Regiones de interés)

    Dependiendo de la aplicación varía el uso de funciones del *API*, el proyecto se centra en una red, mediante la conversión de los *frames* en datos binarios llamados *Blobs*, la función es *GetBlob*. De esta forma el proceso de inferencia consume menos recursos ya que todo el proceso es realizado con datos binarios facilitando él envió de estos datos al hardware donde se realiza la inferencia.
7) **Realizar la Inferencia de manera asíncrona o síncrona** con diferentes funciones para las solicitudes.

    La función *StartAsync* (para manera asíncrona) regresa automáticamente y comienza la inferencia sin bloquear el lazo *main*, la función *Infer* (para manera síncrona) bloquea el lazo *main* y regresa cuando la inferencia es completa. La función *Wait* para esperar que el resultado pase a estar disponible para la solicitud asíncrona.

    Hay 3 maneras de realizar de realizar este proceso:

    - Especifica una duración máxima para bloquear. Este método es bloqueado hasta que el periodo de tiempo indicado transcurra o el resultado esté disponible, lo que suceda primero.

- La función *RESULTREADY* espera hasta que el resultado esté disponible.
- La función *STATUS_ONLY* inmediatamente regresa la solicitud de estatus, sin importar la condición de bloqueado o interrumpido del lazo.

Ambas solicitudes pueden ser llamadas de diferentes lazos del programar sin temer que el programa se corrompa o falle.

8) **Examinar los *Blob* de salida y procesar la salida**, este proceso es complementado con *OpenCV* para presentar una interfaz con los resultados de la inferencia.

El proyecto cumple con estos pasos pero una vez creada la solicitud de inferencia, el programa envía la entrada de datos, videos, a un *batch* o lote el cual va permitir que realicemos la inferencia de diferentes videos en un solo programa, entonces realiza los pasos 6, 7 y 8 para el primer video del lote, realiza la inferencia, procesa el resultado y continua con el segundo video del lote, cabe destacar que el programa también funciona si solo incluye un video de entrada. Al finalizar los videos del lote, vuelve al primer video y realiza el proceso de inferencia para el segundo *frame* del video y así sucesivamente, esto esquematizado en la ilustración 3.9. Todo este proceso es realizado de la manera asíncrona ya que nunca bloquea el lazo *main*.

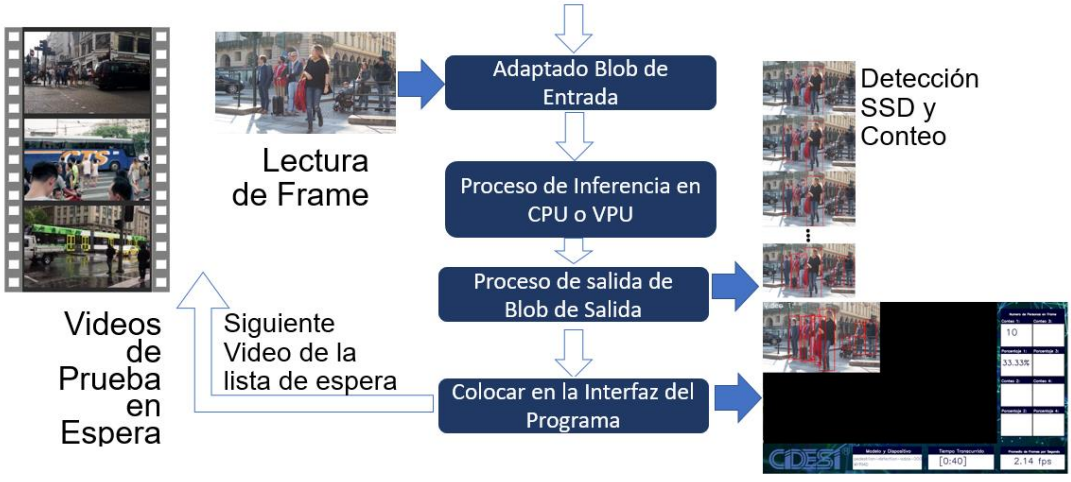


Ilustración 3.8: Proceso de Inferencia.

3.4 Creación y ejecución de la aplicación.

Al tener completo el programa se procede a la creación de un ejecutable para tener un *software* el cual pueda ser ejecutado con diferentes modelos *IR* con su respectivo dispositivo de hardware para realizar la inferencia y los videos o video con los que se desea trabajar.

3.4.1 Creación de Aplicación.

Este proceso se realiza desde la terminal del equipo, con sistema operativo de Ubuntu o bien con un subsistema operativo de Linux con Ubuntu, que es posible descargar de manera gratuita desde la Microsoft Store, ilustración 3.9.

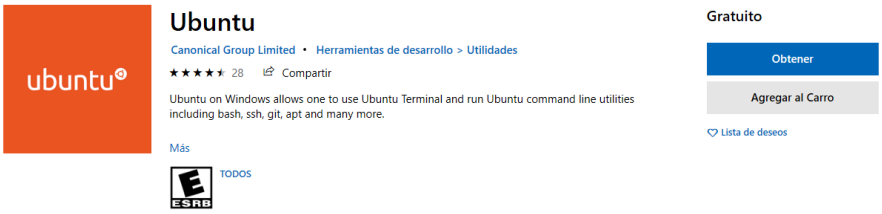


Ilustración 3.9: Descarga de Subsistema de Linux.

Bien puede ser uno u otra, el proyecto hace uso de ambas para comparar el rendimiento, la creación del proyecto requiere que entremos a la terminal y crear un directorio donde se va a crear el programa, ilustración 3.10.

```
fausto@DESKTOP-POQA001:/mnt/c/Users/Fausto/Desktop/Proyecto_Industrial$ mkdir Contador
fausto@DESKTOP-POQA001:/mnt/c/Users/Fausto/Desktop/Proyecto_Industrial$ cd Contador/
fausto@DESKTOP-POQA001:/mnt/c/Users/Fausto/Desktop/Proyecto_Industrial/Contador$
```

Ilustración 3.10: Creación de Directorio para Programa.

Una vez que estamos en el directorio, ingresamos el siguiente código de línea con la dirección del programa a implementar, mediante la generación de los archivos *make* que requiere el toolkit de *OpenVINO™*, es importante que tengamos el ambiente de *OpenVINO* activo que se indicó en tema 3.2.1:

```
cmake
/mnt/c/Users/Fausto/Desktop/Proyecto_Industrial/IE_Programa
```

Suele tardar hasta un minuto y tenemos una pantalla parecida a la ilustración 3.11:

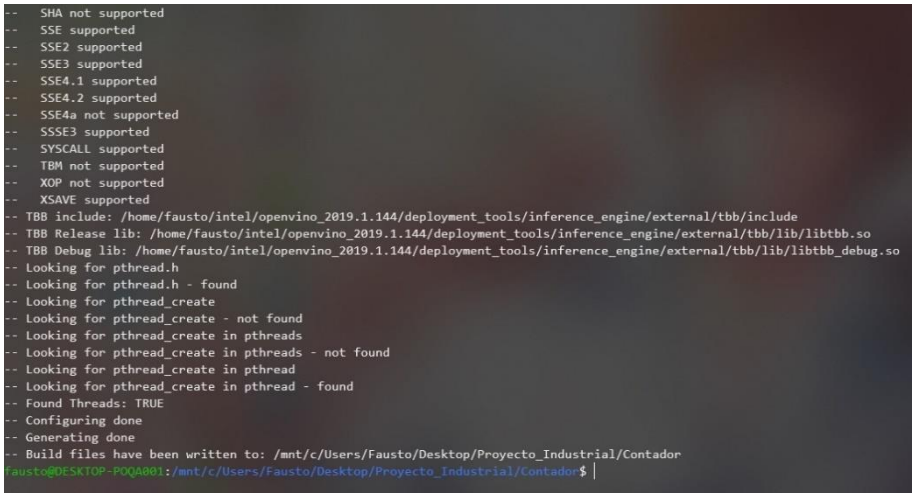


Ilustración 3.11: Generación de Archivos make.

Con los archivos *make* creados, basta con ingresar la siguiente línea de comando para construir el programa, con el nombre del proyecto, de no existir error dentro del código se presenta una pantalla similar a la ilustración 3.12.

```
make project_ind_crowd_subway_det_reid
```

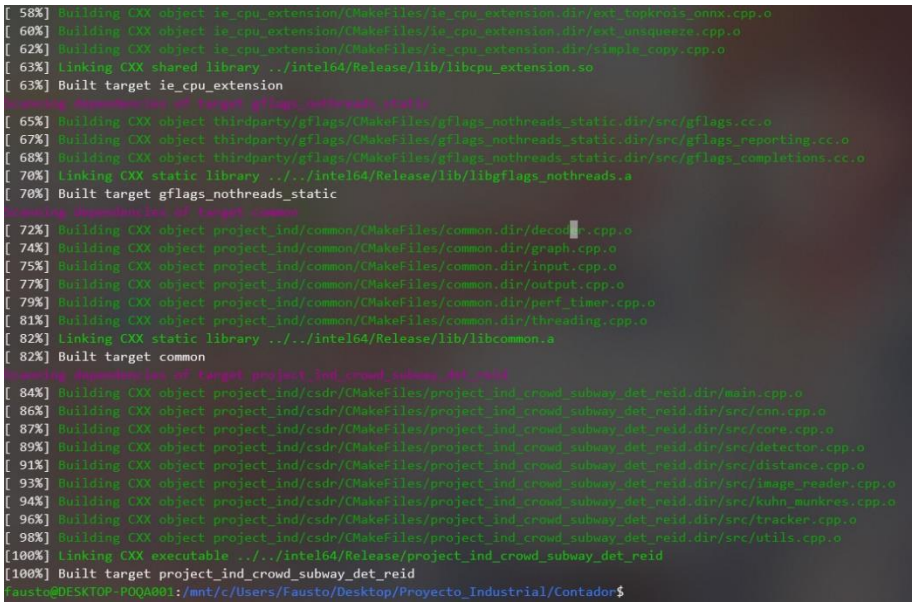


Ilustración 3.12: Creación de programa exitosa.

Este proceso de construcción de la aplicación requiere un periodo de tiempo largo de 5 minutos la primera vez, si existen errores y se hacen cambios en el código, el tiempo de creación es mucho menor, la velocidad suele variar dependiendo del equipo.

3.4.2 Ejecución de la aplicación.

Una vez creada la aplicación, la ejecución de éste es sencilla, lo primero es movernos en el directorio de la terminal en donde se creó el programa, si nos encontramos en la misma dirección basta con el siguiente comando para llegar a la carpeta deseada.

```
cd intel64/release
```

Esa es la ubicación del programa, el programa requiere un modelo y datos de entrada, si no se selecciona dispositivo, por defecto el programa indica que será la *CPU*, así que el modelo que indiquemos debe ser de formato *fp32*; si fuese *fp16*, el programa despliega un error y no despliega la interfaz, el programa requiere el siguiente comando, en el que llamas al nombre del proyecto creado, como el siguiente comando.

```
./project_ind_crowd_subway_det_reid
```

El código de línea anterior despliega un error, ya que no cuenta con los requisitos necesarios. La tabla 3.3 indica estos requerimientos:

Comando	Descripción
-i	Seleccionar dirección o direcciones de datos de entrada, ya sea imágenes o videos, si no se especifica, la dirección precisa y se indica la carpeta, toma todos los videos y/o imágenes y los indica como entrada. Por cuestiones de visibilidad el límite es de 25 imagines/videos.
-m_det	Seleccionar dirección precisa del archivo de topología del modelo (*.xml*), dentro de la misma carpeta debe de encontrarse el archivo de pesos (*.bin*).
-d_det	Seleccionar dispositivo *CPU* o *MYRIAD*, de no especificar, el programa asume que es *CPU*.

-nc	Indicar el número de cámaras conectadas, de no indicar, el valor por defecto es 0. Este es el verdadero uso del programa, en el uso de diversas cámaras de seguridad, pero las pruebas se realizan con los videos de prueba.

Tabla 3.4: Comando de Ejecución.

El código de línea siguiente es un ejemplo de la ejecución de la aplicación, sí todo es correcto, se despliega el siguiente texto en la terminal, ilustración 3.13.

```
./project_ind_crowd_subway_det_reid -i
/mnt/c/Users/Fausto/Desktop/Proyecto_Industrial/Videos/Videos_City/vid_
cit2.mp4 -m_det
/mnt/c/Users/Fausto/Desktop/Proyecto_Industrial/IR_models_Sub/fp32/Tran
sportation/object_detection/pedestrian/mobilenet-reduced-
ssd/dldt/pedestrian-detection-adas-0002.xml
```

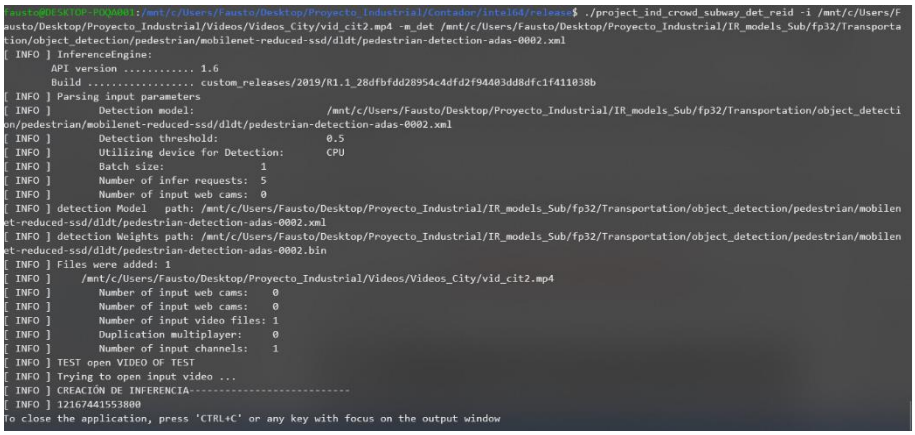


Ilustración 3.13: Ejecución de aplicación exitosa.

3.4.3 Interfaz del programa.

Aparte del texto de la ilustración 3.13, se despliega una ventana que contiene la interfaz, ilustración 3.14, en la que visualizamos el video indicado en la terminal, así como el conteo y porcentaje respecto a un número máximo de personas en el área designada, si esta excede el 50% de capacidad, el color de los números pasa a ser rojo, esto para que el operador del sistema preste mayor atención a estos videos de mayor congestión.

La interfaz también muestra el nombre del modelo y el dispositivo *plugin* que lo está procesando, el tiempo de procesamiento y velocidad que nos permite conocer el rendimiento en los resultados.

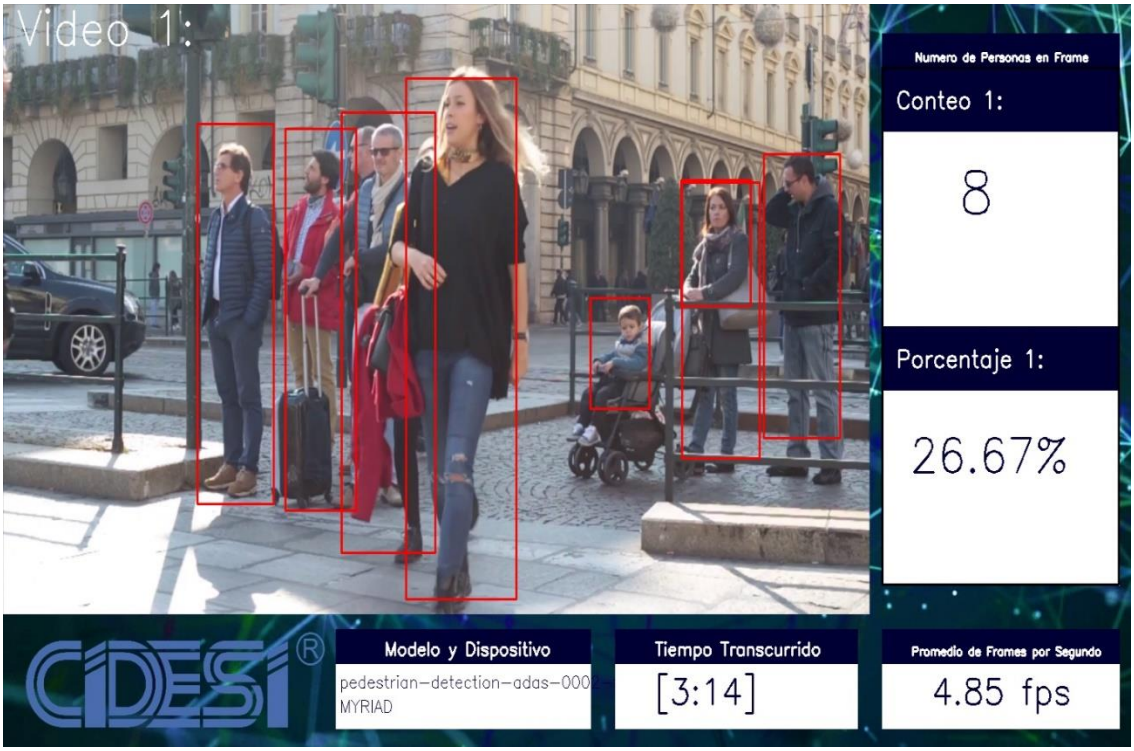


Ilustración 3.14: Interfaz final del programa.

Ya que el proceso de inferencia es continuo y al finalizar el video, vuelve a comenzar, para finalizar el proceso de inferencia se requiere que en la terminal se presione “ctrl + c”, si solo se cierra la ventana de la interfaz, esta vuelve a desplegarse.

Capitulo 4: Experimentación y Resultados.

4.1 Introducción.

Una vez que el desarrollo ha finalizado, se tiene una aplicación a la cual se pone a prueba durante este capítulo para analizar su rendimiento bajo distintos parámetros, como es el equipo de inferencia, el número de videos de entrada y el modelo usado que determina el objeto de detección.

Este capítulo hace uso de equipos para realizar el proceso de inferencia, equipos que se indican en la tabla 3.1, a la cual ahora etiquetaremos como tabla 4.1:

Ilustración	Marca	Modelo	Procesador	RAM
	Zotac	ZBOX CI543 Nano	Intel Core i5-6300U-2.50 GHz	8Gb
	Dell	Inspirion 14z 5423	Intel Core i7-3517U - 1.90GHz	8Gb
	Movidius	Movidius NCS	Movidius Myriad 2	4 Gb

Tabla 4.1: Características de equipos de inferencia para generación de resultados.

4.2 Resultados de Conteo de Personas.

El conteo de personas se realiza en tres ambientes, la ciudad, el metro y un centro comercial. Y de cada uno se extrae la velocidad promedio en *frames* por segundo.

4.2.1 Resultados de Conteo de Personas mediante CPU de equipo Dell.

El equipo Dell cuenta con el subsistema de Linux y un programa que permite abrir ventanas para visualizar la interfaz, aprovechando el sistema operativo de Windows, mostraremos la gráfica de porcentaje de uso del CPU. La ilustración 4.1 muestra el porcentaje de uso de la CPU en condiciones normales.

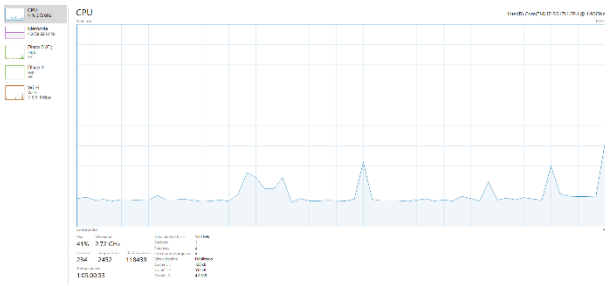


Ilustración 4.1: Condiciones Iniciales de la CPU de Equipo Dell

La ilustración 4.2 muestra una gráfica del porcentaje de uso de la CPU al realizar la inferencia de un solo video del 100%. Para inferencia de 2 o más videos, el resultado de igual manera es del 100%

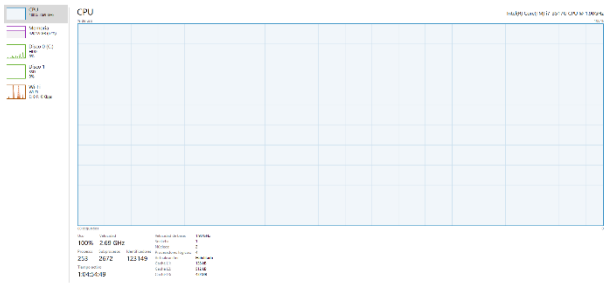


Ilustración 4.2: Condiciones de la CPU de Equipo Dell durante Inferencia.

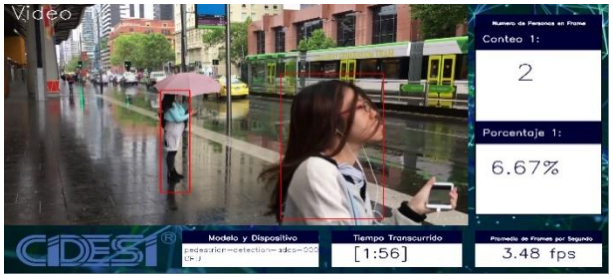


Ilustración 4.3: Conteo Ciudad - CPU Dell - 1 Video.

Ilustración 4.4: Conteo Ciudad - CPU Dell - 2 Video.

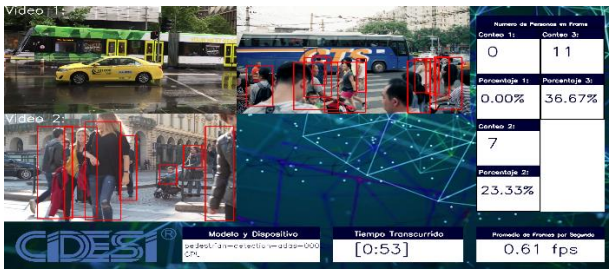


Ilustración 4.5: Conteo Ciudad - CPU Dell - 3 Video.

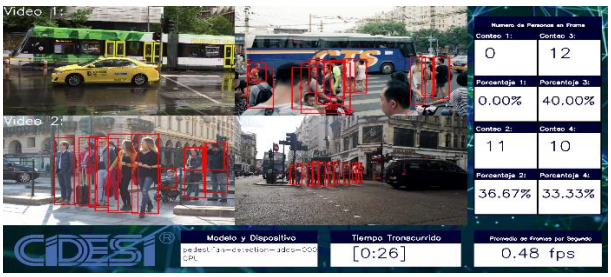


Ilustración 4.6: Conteo Ciudad - CPU Dell - 4 Video

Las ilustraciones 4.3 a 4.6 muestran los resultados de la aplicación con un modelo de detección de peatones en una ciudad, cada ilustración muestra una velocidad promedio de *FPS*, para facilitar la lectura de datos, ingresamos esta información en la tabla 4.2.

Num. de Videos de Entrada	FPS Promedio
1	3.48
2	1.41
3	0.61
4	0.48

Tabla 4.2: FPS en Ciudad - CPU Dell

Las ilustraciones 4.7 a 4.10 muestran los resultados de la aplicación con un modelo de detección de peatones en un metro.

Ilustración 4.7: Conteo Metro - CPU Dell - 1 Video.

Ilustración 4.8: Conteo Metro - CPU Dell - 2 Video.

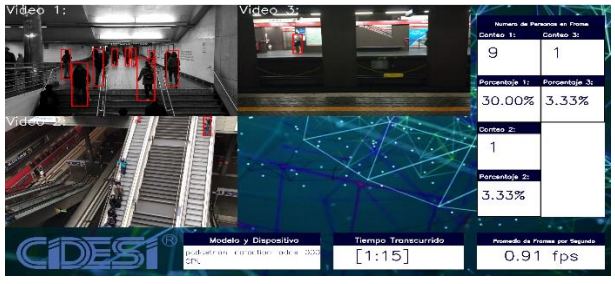


Ilustración 4.9: Conteo Metro - CPU Dell - 3 Video.

Ilustración 4.10: Conteo Metro - CPU Dell - 4 Video.

Al igual que en la ciudad asignamos los valores de *FPS* en la tabla 4.3.

Num. de Videos de Entrada	FPS Promedio
1	3.91
2	1.53
3	0.91
4	0.65

Tabla 4.3: FPS en Metro - CPU Dell

Las ilustraciones 4.11 y 4.12 muestran el último ambiente, un centro comercial y reúnen los datos de FPS en la tabla 4.4.

Ilustración 4.11: Conteo Mall - CPU Dell - 1 Video.

Ilustración 4.12: Conteo Mall - CPU Dell - 2 Video.

Num. de Videos de Entrada	FPS Promedio
1	4.19
2	1.12

Tabla 4.4: FPS en Mall - CPU Dell

4.2.2 Resultados de Conteo de Personas mediante CPU de equipo Zotac.

El equipo de marca Zotac cuenta con Ubuntu como sistema operativo, en este equipo, el consumo del *CPU* sin realizar la inferencia es como lo muestra la gráfica de la ilustración 4.13:

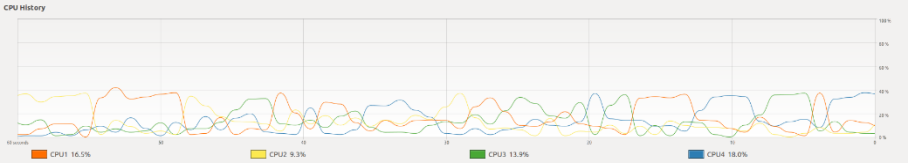


Ilustración 4.13: Condiciones Iniciales de la CPU de Equipo Zotac.

El equipo entrega resultados similares que el equipo Dell, de igual manera en la ilustración 4.14 refleja como al realizar la inferencia en al menos un video, esta consume todos los recursos del *CPU*, siendo nuevamente parecido al equipo Dell.

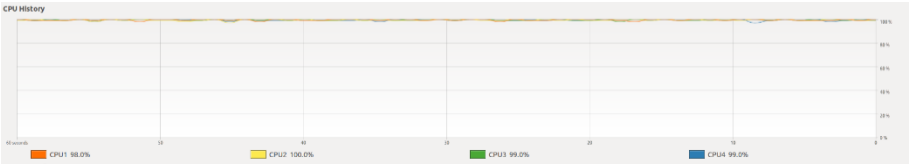


Ilustración 4.14: Condiciones de la CPU de Equipo Zotac durante Inferencia.

Después se realiza el proceso de inferencia en los mismos ambientes que con el equipo Dell para posteriormente realizar una comparación del rendimiento, comenzando en la ciudad con las ilustraciones 4.14 a 4.17 y en la tabla 4.5 junta los datos obtenidos del promedio de *FPS*.

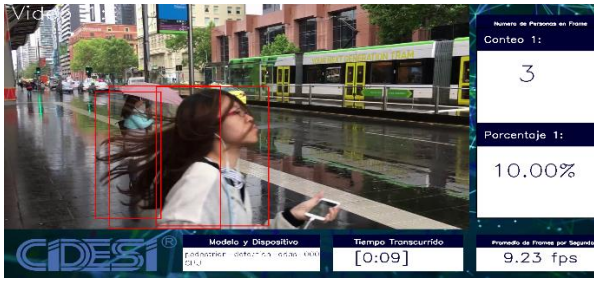


Ilustración 4.15: Conteo Ciudad - CPU Zotac - 1 Video.

Ilustración 4.16: Conteo Ciudad - CPU Zotac - 2 Video.

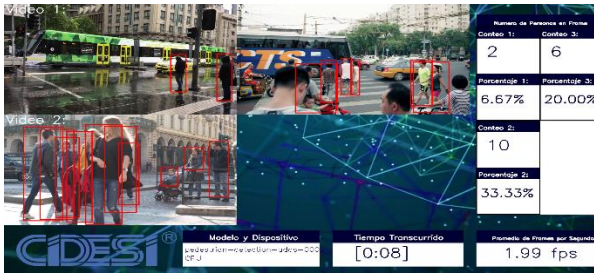


Ilustración 4.17: Conteo Ciudad - CPU Zotac - 3 Video.

Ilustración 4.18: Conteo Ciudad - CPU Zotac - 4 Video.

Num. de Videos de Entrada	FPS Promedio
1	9.23
2	3.10
3	1.99
4	1.08

Tabla 4.5: FPS en Ciudad - CPU Zotac.

Las ilustraciones 4.19 a 4.22 es el conteo de personas mediante la *CPU* del equipo Zotac y guardamos estos datos en la tabla 4.6.

Ilustración 4.19: Conteo Metro - CPU Zotac - 1 Video.

Ilustración 4.20: Conteo Metro - CPU Zotac - 2 Video.

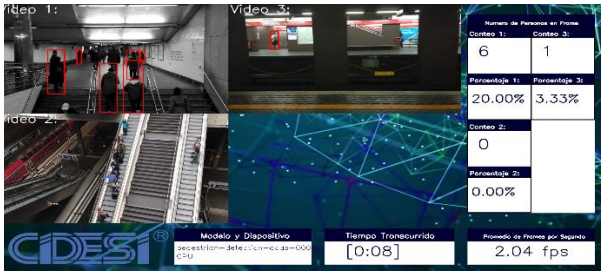


Ilustración 4.21: Conteo Metro - CPU Zotac - 3 Video.

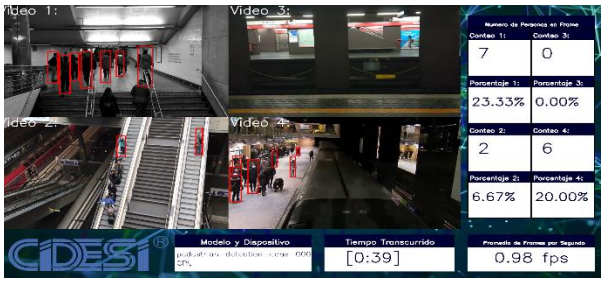


Ilustración 4.22: Conteo Metro - CPU Zotac - 4 Video.

Num. de Videos de Entrada	FPS Promedio
1	8.16
2	2.19
3	2.04
4	0.98

Tabla 4.6: FPS en Metro - CPU Zotac

Ya por último, la ilustración 4.23 y 4.24 muestran el resultado de la inferencia en el equipo Zotac en un centro comercial, con su respectiva tabla 4.7 con los resultados.

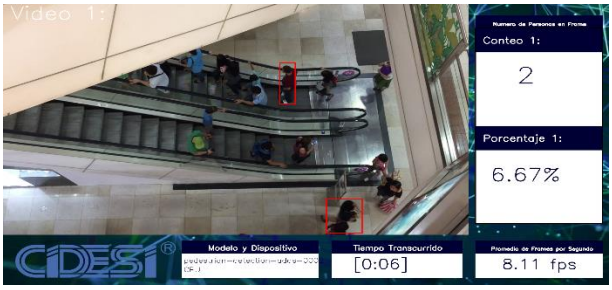


Ilustración 4.23: Conteo Mall - CPU Zotac - 1 Video.

Ilustración 4.24: Conteo Mall - CPU Zotac - 2 Video.

Num. de Videos de Entrada	FPS Promedio
1	8.11
2	3.43

Tabla 4.7: FPS en Mall - CPU Zotac

4.2.3 Resultados de Conteo de Personas mediante VPU de equipo Movidius™ NCS.

El último equipo para realizar pruebas de inferencia es *Movidius™ NCS*, el cual prometo un mejor comportamiento que los equipos anteriores, si bien el uso del VPU permite que al realizar la inferencia el porcentaje de uso del *CPU* no se eleve como sucede con el resto de los equipos.

Movidius™ NCS se ocupa en el equipo Zotac, debido a que no funciona en el equipo Dell ya que el uso del subsistema limita el lector de Dispositivo *Hardware* como la conexión *USB*, aunque esta característica que limita al subsistema de Linux será corregida en futuras actualizaciones de *Windows*. La ilustración 4.13 muestra como es el uso del *CPU* al no realizar ninguna inferencia, y la ilustración 4.25 muestra como es uso del *CPU* al realizar la inferencia de un video de entrada en un dispositivo externo al *CPU*.

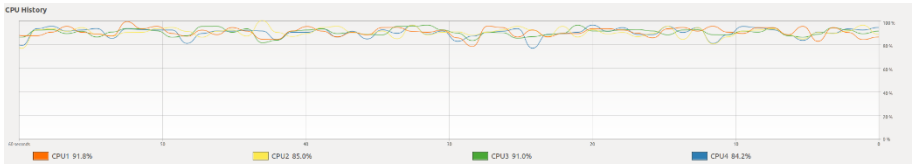


Ilustración 4.25: Condiciones de la CPU de Equipo Zotac durante Inferencia de un video en VPU.

Como se observa, ya no se consume todos los recursos de la *CPU*, pero aun así consume un 80% aproximadamente del uso, principalmente por el uso de la terminal ya que constantemente manda los datos Blob de entrada al dispositivo *Movidius™ NCS*, para ir hacer la inferencia en él *VPU Myriad™*, al verificar el proceso en la inferencia de dos videos, ilustración 4.26, este consume hasta el 100%.

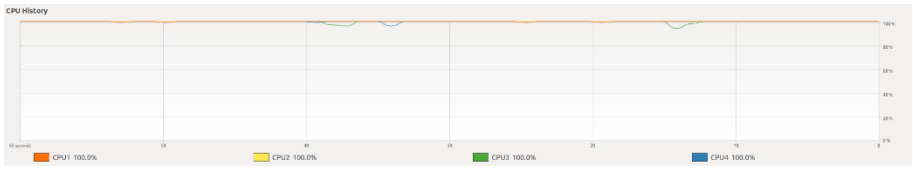


Ilustración 4.26: Condiciones de la CPU de Equipo Zotac durante Inferencia de dos o más videos en VPU

Las ilustraciones 4.27 a 4.30, muestran los resultados de inferencia mediante el uso de *Movidius™ NCS* en la ciudad, mientras que la tabla 4.8 reúne los datos para su posterior análisis.

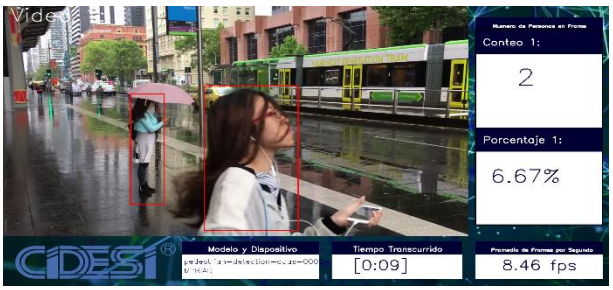


Ilustración 4.27: Conteo Ciudad - VPU - 1 Video.

Ilustración 4.28: Conteo Ciudad - VPU - 2 Video.

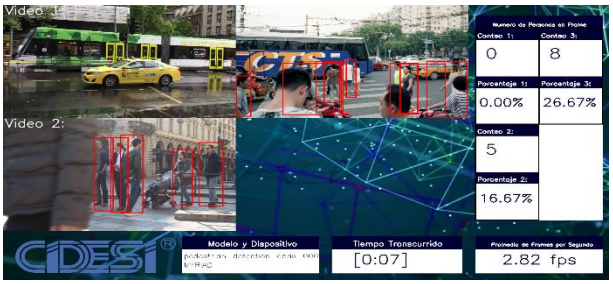


Ilustración 4.29: Conteo Ciudad - VPU - 3 Video.

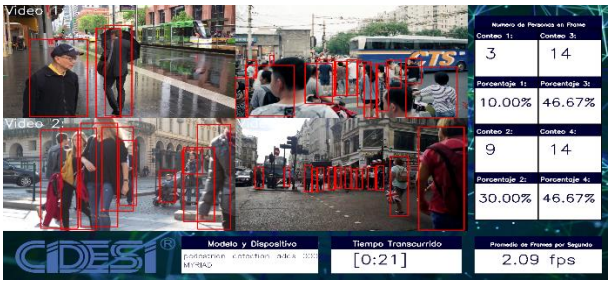


Ilustración 4.30: Conteo Ciudad - VPU - 4 Videos.

Num. de Videos de Entrada	FPS Promedio
1	8.46
2	4.24
3	2.82
4	2.09

Tabla 4.8: FPS en Ciudad - VPU

Prosiguiendo con el análisis con el acelerador de inferencia *Movidius™ neuronal Compute Stick* en el metro con las ilustraciones 4.31 a 4.34 del ambiente del metro, con los datos de *FPS* en la tabla 4.9.

Ilustración 4.31: Conteo Metro - VPU - 1 Video.

Ilustración 4.32: Conteo Metro - VPU – 2 Video.

Ilustración 4.33: Conteo Metro - VPU - 3 Video.

Ilustración 4.34: Conteo Metro - VPU - 4 Video.

Num. de Videos de Entrada	FPS Promedio
1	8.50
2	4.28
3	2.83
4	2.12

Tabla 4.9: FPS en Metro - VPU

Ya el último análisis con el modelo de detección de peatones son las ilustraciones 4.34 y 4.35, del conteo en centro comercial.

Ilustración 4.35: Conteo Mall - VPU - 1 Video.

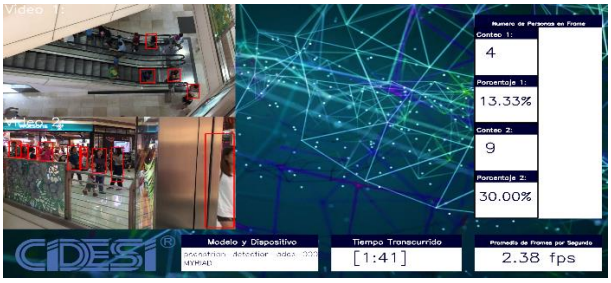


Ilustración 4.36: Conteo Mall - VPU - 2 Video.

Los datos del promedio de *frames* por segundo en la tabla 4.10**.**

Num. de Videos de Entrada	FPS Promedio
1	8.55
2	2.38

Tabla 4.10: FPS en Mall – VPU.

4.2.4 Comparación de Rendimiento en Conteo de Personas.

Al finalizar las pruebas con el modelo de detección de peatones en diferentes ambientes, se procede a realizar una comparación reuniendo los datos, primero las tablas 4.2, 4.5 y 4.8 para ver cómo fue su comportamiento con un ambiente similar de ciudad, estos datos se muestran en la tabla 4.11.

Num. de Videos de Entrada	FPS Promedio en Dell	FPS Promedio en Zotac	FPS Promedio en Movidius NCS
1	3.48	9.23	8.46
2	1.41	3.10	4.24
3	0.61	1.99	2.82
4	0.48	1.08	2.09

Tabla 4.11: Comparación de FPS en Ciudad.

Para una mejor interpretación de los datos de tabla 4.11, se incluye en una gráfica de la ilustración 4.37 para ver su comportamiento respecto al número de videos de entrada.

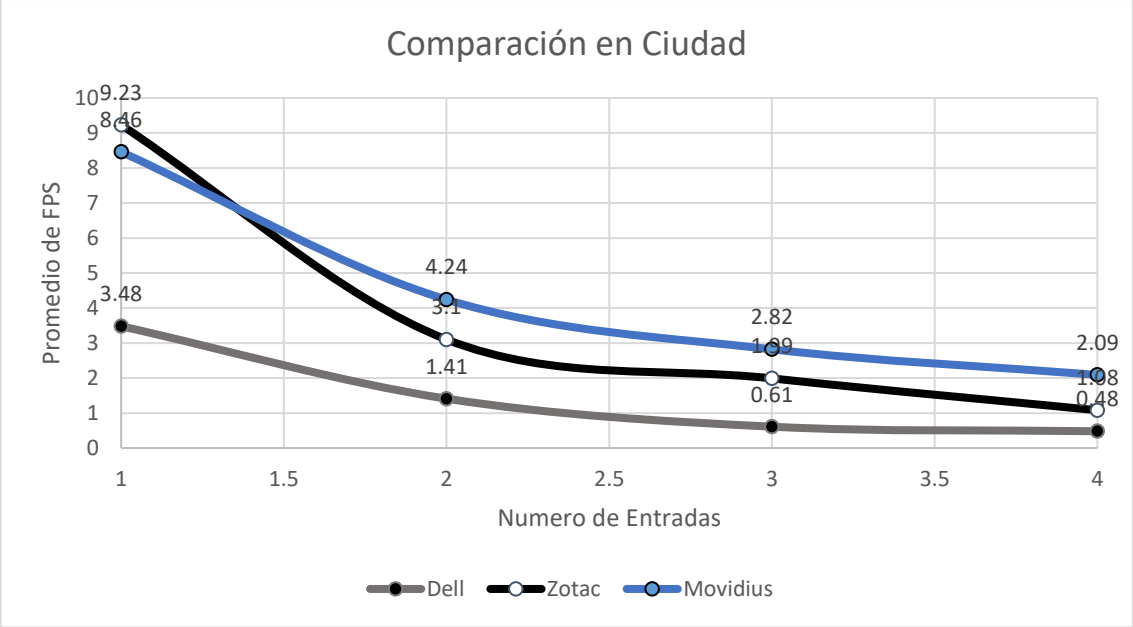


Ilustración 4.37: Grafica Comparativa de rendimiento en Ciudad.

Ahora con las tablas que respectan al metro, tabla 4.4, 4.6 y 4.9, se integran en la tabla 4.12, con su grafica para corroborar el rendimiento, ilustración 4.38.

Num. de Videos de Entrada	FPS Promedio en Dell	FPS Promedio en Zotac	FPS Promedio en Movidius NCS
1	3.91	8.16	8.50
2	1.53	2.19	4.28
3	0.91	2.04	2.83
4	0.65	0.98	2.12

Tabla 4.12: Comparación de FPS en Metro.

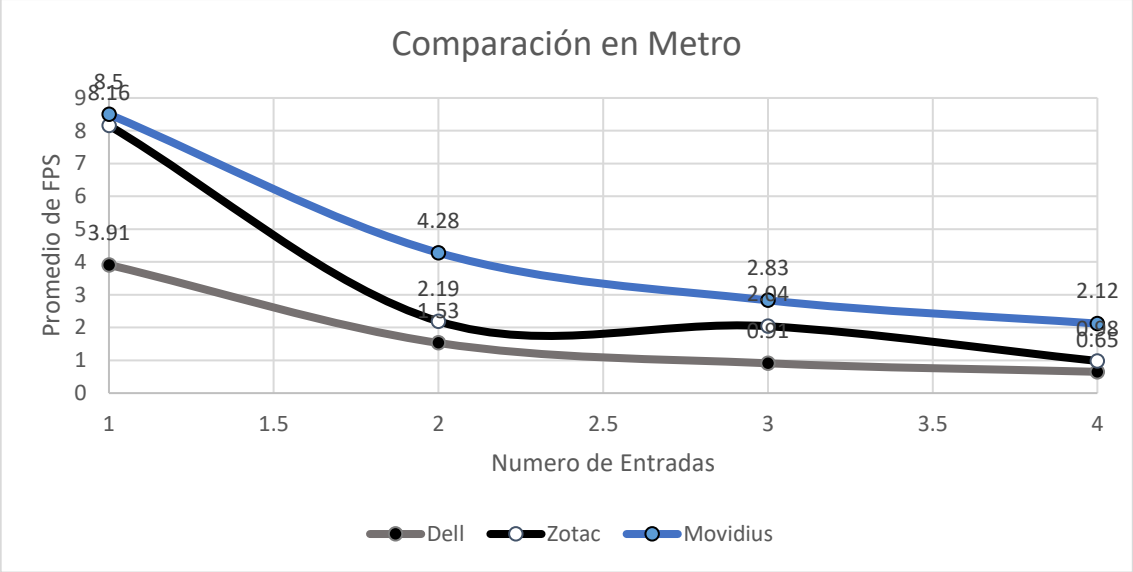


Ilustración 4.38: Grafica Comparativa de rendimiento en Metro

Las tablas 4.4, 4.7 y 4.10 son las tablas de datos que se incorporan en la tabla 4.13, para integrar en una gráfica comparativa de la ilustración 4.39.

Num. de Videos de Entrada	FPS Promedio en Dell	FPS Promedio en Zotac	FPS Promedio en Movidius NCS
1	4.19	8.11	8.55
2	1.12	3.43	2.38

Tabla 4.13: Comparación de FPS en Metro.

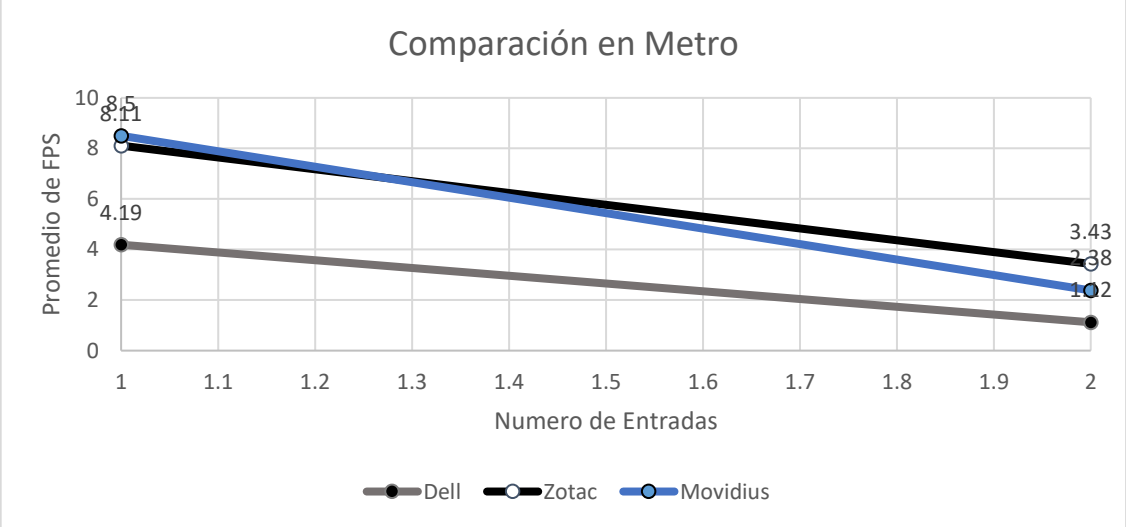


Ilustración 4.39: Grafica Comparativa de rendimiento en Metro

4.3 Resultados de Conteo de Vehículos.

Ya se observaron los resultados de conteo de peatones en diferentes ambientes, ahora al utilizar videos de prueba de vehículos en diversos ambientes, con la detección de vehículos.

Algunos de los datos generales que se observan en las ilustraciones 4.37 a 4.39 es que conforme el número entradas incrementa disminuye la velocidad promedio de inferencia, por lo que con estos videos de prueba se comprueba los límites que cada dispositivo permite.

4.3.1 Resultados de Conteo de Vehículos mediante CPU de equipo Dell.

Este equipo es el que ofrece la peor velocidad de inferencia según el comportamiento obtenido en el conteo de peatones, por lo que en las siguientes ilustraciones se analiza su comportamiento en el conteo de vehículos, pero para este ambiente se busca llevar el equipo al límite para conocer el número máximo de entradas permitidas y si esto varia respecto a los otros dispositivos.

Las ilustraciones 4.40 a 4.61 muestras los resultados de la aplicación hasta con 22 entradas de manera simultánea, al colocar una entrada más, el programa se corrompe. Cabe destacar que realizar la aplicación con este equipo resulta totalmente ineficiente, ya que arriba de 9 entradas, el tiempo de para la creación de la solicitud de inferencia es mayor a 3 minutos, llegando a 15 minutos cuando se introducen 22 videos.

Ilustración 4.40: Conteo Vehículos - CPU Dell - 1 Video.

Ilustración 4.41: Conteo Vehículos - CPU Dell - 2 Video.

Ilustración 4.42: Conteo Vehículos - CPU Dell - 3 Video.

Ilustración 4.43: Conteo Vehículos - CPU Dell - 4 Video.

Ilustración 4.44: Conteo Vehículos - CPU Dell - 5 Video.

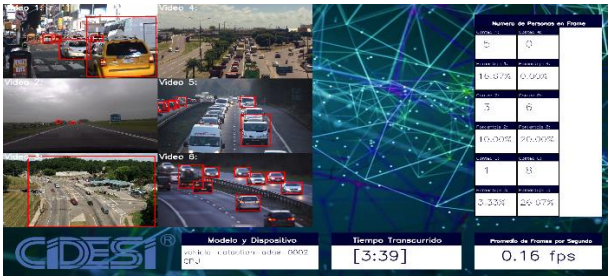


Ilustración 4.45: Conteo Vehículos - CPU Dell - 6 Video.

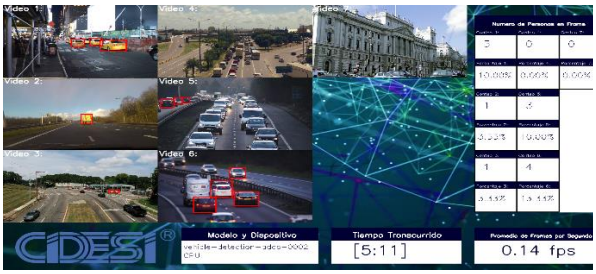


Ilustración 4.46: Conteo Vehículos - CPU Dell - 7 Video.

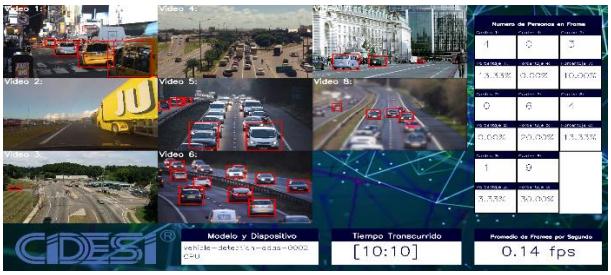


Ilustración 4.47: Conteo Vehículos - CPU Dell - 8 Video

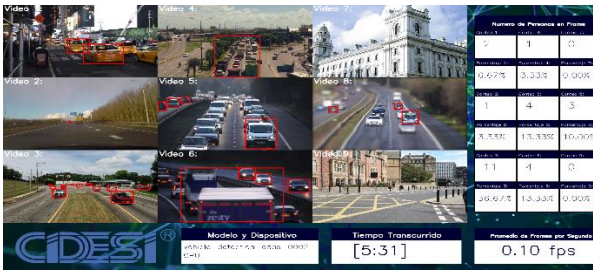


Ilustración 4.48: Conteo Vehículos - CPU Dell - 9 Video.

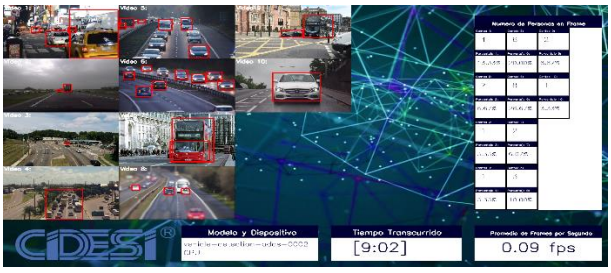


Ilustración 4.49: Conteo Vehículos - CPU Dell - 10 Video.

Ilustración 4.50: Conteo Vehículos - CPU Dell - 11 Video.

Ilustración 4.51: Conteo Vehículos - CPU Dell - 12 Video.

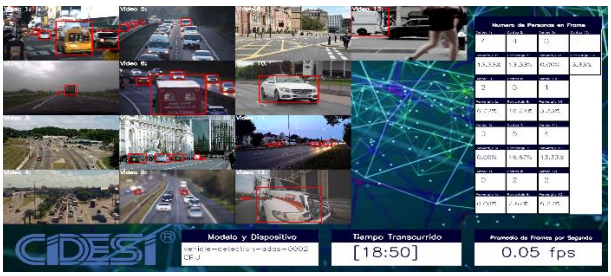


Ilustración 4.52: Conteo Vehículos - CPU Dell - 13 Video.

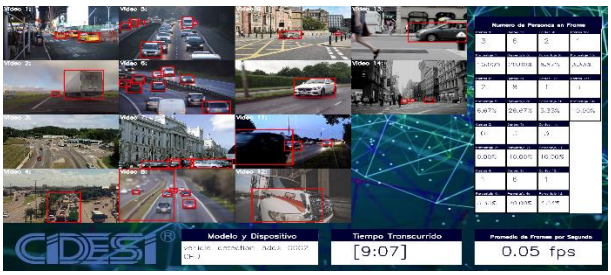


Ilustración 4.53: Conteo Vehículos - CPU Dell - 14 Video.

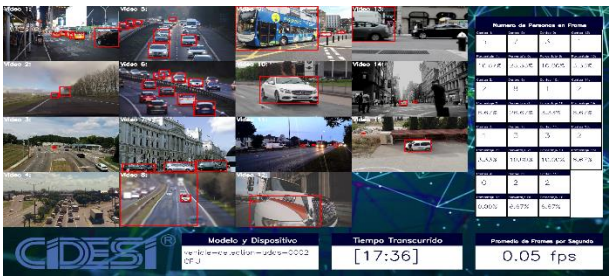


Ilustración 4.54: Conteo Vehículos - CPU Dell - 15 Video.

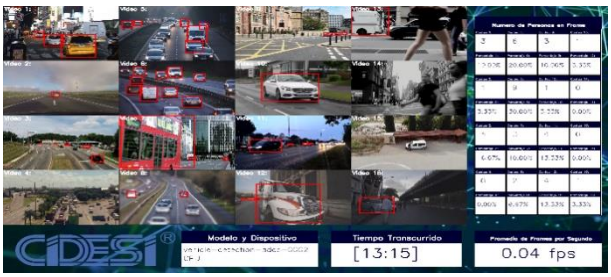


Ilustración 4.55: Conteo Vehículos - CPU Dell - 16 Video.

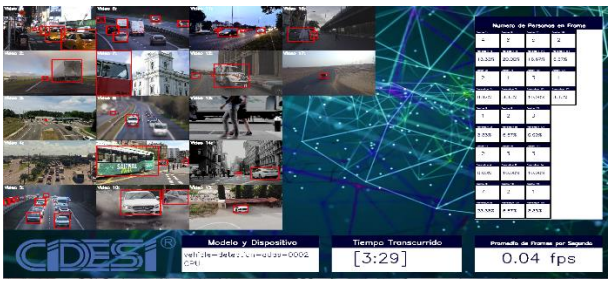


Ilustración 4.56: Conteo Vehículos - CPU Dell - 17 Video.

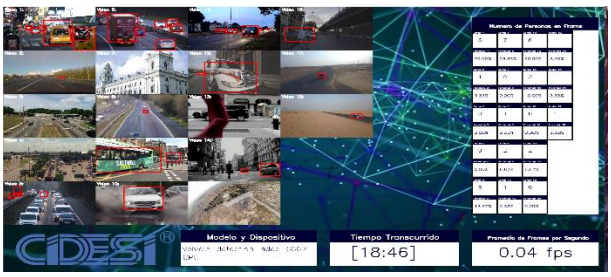


Ilustración 4.57: Conteo Vehículos - CPU Dell - 18 Video.

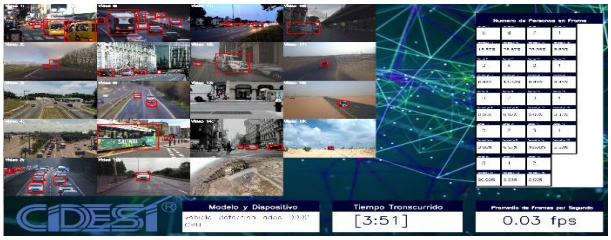


Ilustración 4.58: Conteo Vehículos - CPU Dell - 19 Video.

Ilustración 4.59: Conteo Vehículos - CPU Dell - 20 Video.

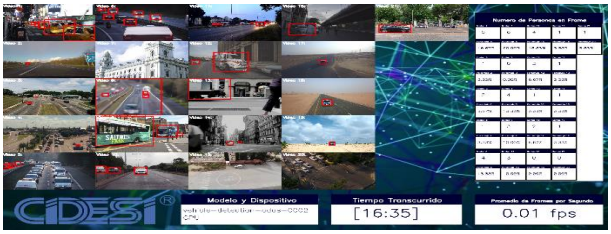


Ilustración 4.60: Conteo Vehículos - CPU Dell - 21 Video.

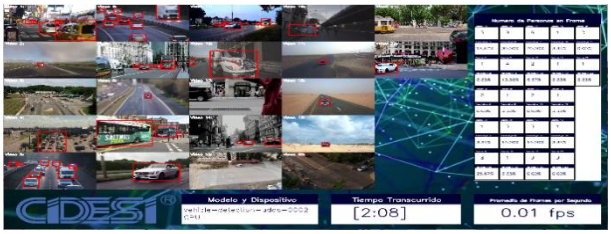


Ilustración 4.61: Conteo Vehículos - CPU Dell - 22 Video.

Al igual que el conteo de peatones, se reúnen los datos respecto al promedio de *FPS* del resultado de las aplicaciones respecto al número de entradas, en la tabla 4.14.

Num. de Entrada	Promedio de FPS	Num. de Entrada	Promedio de FPS
1	4.09	**14**	0.05
2	1.76	**15**	0.05
3	0.81	**16**	0.04
4	0.44	**17**	0.04
5	0.27	**18**	0.04
6	0.16	**19**	0.03
7	0.14	**20**	0.03
8	0.14	**21**	0.01
9	0.10	**22**	0.01
10	0.09	**23**	0.00
11	0.08	**24**	0.00
12	0.07	**25**	0.00
13	0.05	**26**	-

Tabla 4.14: FPS en conteo de Vehículos desde CPU Dell.

4.3.2 Resultados de Conteo de Vehículos mediante CPU de equipo Zotac.

Después de visualizar que el equipo Dell soporta hasta 22 entradas, en las siguientes ilustraciones está el rendimiento del Equipo Zotac con el sistema operativo de Ubuntu.

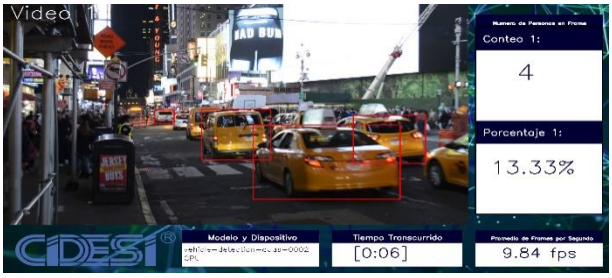


Ilustración 4.62: Conteo Vehículos - CPU Zotac - 1 Video.

Ilustración 4.63: Conteo Vehículos - CPU Zotac - 2 Video.

Ilustración 4.64: Conteo Vehículos - CPU Zotac - 3 Video.

Ilustración 4.65: Conteo Vehículos - CPU Zotac - 4 Video.

Ilustración 4.66: Conteo Vehículos - CPU Zotac - 5 Video.

Ilustración 4.67: Conteo Vehículos - CPU Zotac - 6 Video

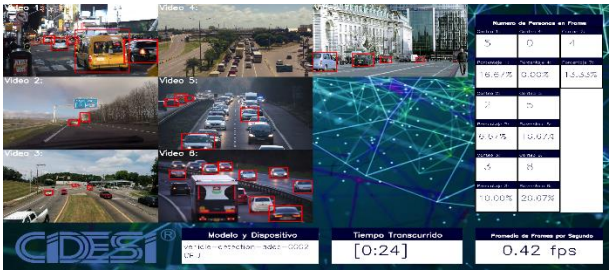


Ilustración 4.68: Conteo Vehículos - CPU Zotac - 7 Video.

Ilustración 4.69: Conteo Vehículos - CPU Zotac - 8 Video.

Ilustración 4.70: Conteo Vehículos - CPU Zotac - 9 Video.

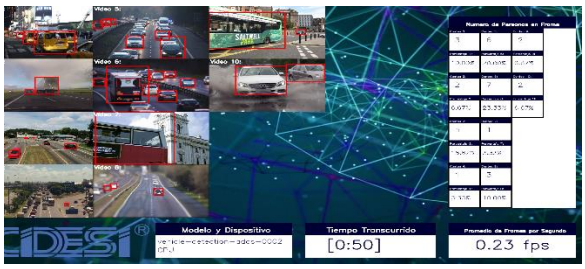


Ilustración 4.71: Conteo Vehículos - CPU Zotac - 10 Video.

Ilustración 4.72: Conteo Vehículos - CPU Zotac - 11 Video.

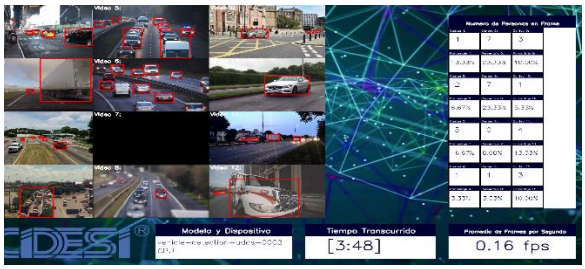


Ilustración 4.73: Conteo Vehículos - CPU Zotac - 12 Video.

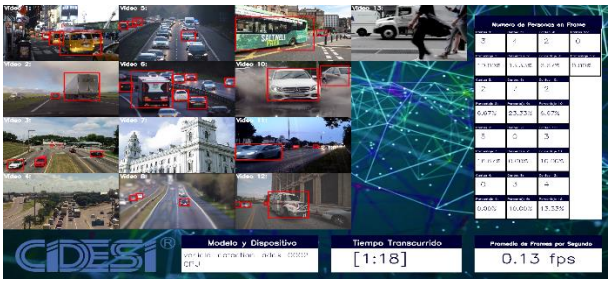


Ilustración 4.74: Conteo Vehículos - CPU Zotac - 13 Video.

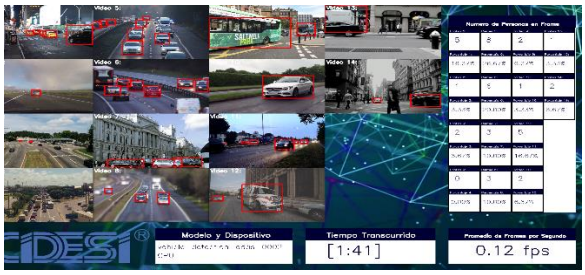


Ilustración 4.75: Conteo Vehículos - CPU Zotac - 14 Video.

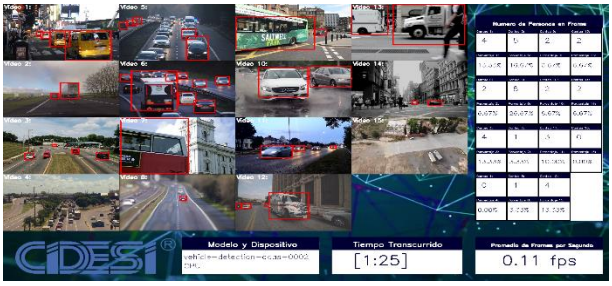


Ilustración 4.76: Conteo Vehículos - CPU Zotac - 15 Video.

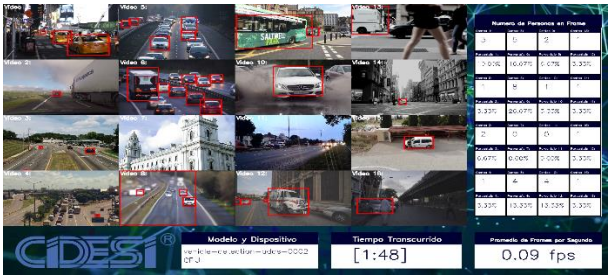


Ilustración 4.77: Conteo Vehículos - CPU Zotac - 16 Video

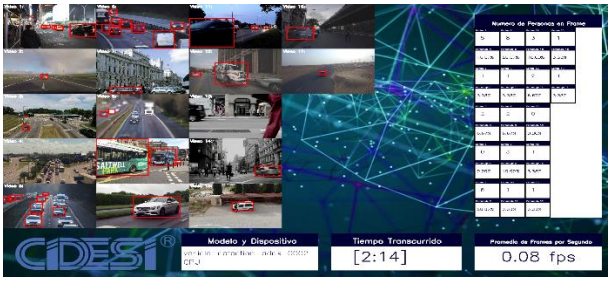


Ilustración 4.78: Conteo Vehículos - CPU Zotac - 17 Video.

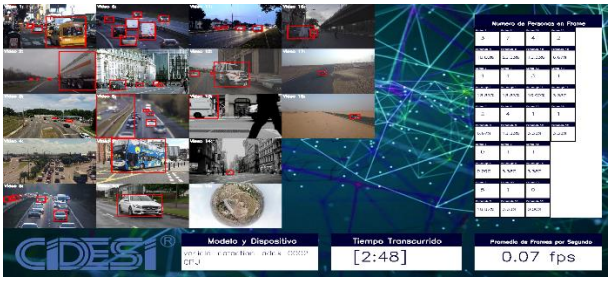


Ilustración 4.79: Conteo Vehículos - CPU Zotac - 18 Video.

Ilustración 4.80: Conteo Vehículos - CPU Zotac - 19 Video.

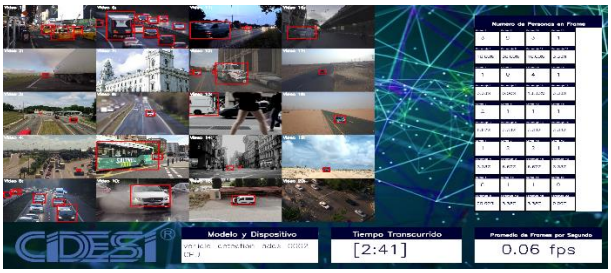


Ilustración 4.81: Conteo Vehículos - CPU Zotac - 20 Video.

Ilustración 4.82: Conteo Vehículos - CPU Zotac - 21 Video.

Ilustración 4.83: Conteo Vehículos - CPU Zotac - 22 Video.

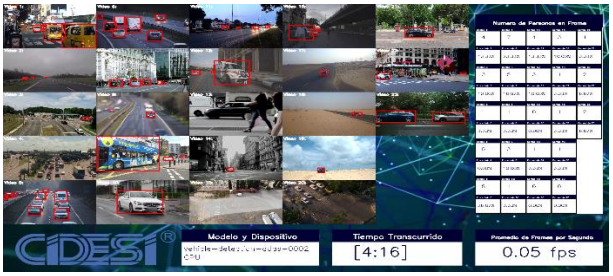


Ilustración 4.84: Conteo Vehículos - CPU Zotac - 23 Video.

Ilustración 4.85: Conteo Vehículos - CPU Zotac - 24 Video

.

Mediante el equipo de marca Zotac, se logró manejar resultados con entradas de 25 videos en cada *frame* de la interfaz, resultados que se presentan en las ilustraciones 4.62 a 4.86. Los resultados que se concentran en la tabla 4.15 para posterior comparación en la tabla 4.17, donde se puede observar cómo es el desempeño de los procesadores convencionales de *Intel*® comparado con procesadores dedicados a aplicaciones de visión, tal como es *Myriad 2*, también de Intel®.

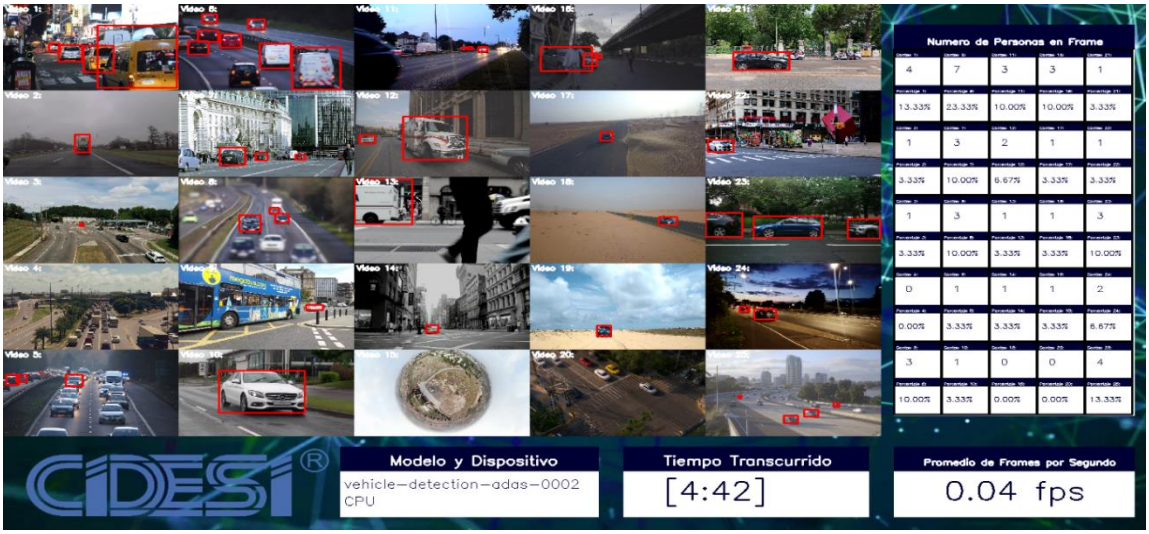


Ilustración 4.86: Conteo Vehículos - CPU Zotac - 25 Video.

Num. de Entrada	Promedio de FPS	Num. de Entrada	Promedio de FPS
1	9.84	**14**	0.12
2	3.16	**15**	0.11
3	2.25	**16**	0.09
4	1.08	**17**	0.08
5	0.85	**18**	0.07
6	0.55	**19**	0.07
7	0.42	**20**	0.06
8	0.31	**21**	0.05
9	0.27	**22**	0.05
10	0.23	**23**	0.05
11	0.19	**24**	0.03
12	0.16	**25**	0.04
13	0.13	**26**	-

Tabla 4.15: FPS en conteo de Vehículos desde CPU Zotac.

4.3.3 Resultados de Conteo de Vehículos mediante VPU de equipo Movidius™ NCS.

El ultimo equipo en el cual se realizan pruebas de la ejecución del programa es el equipo Zotac usando el módulo computacional neuronal *Movidius™ NCS*, ya que, al implementar este módulo, deja a un lado el procesador del equipo Zotac, enfocando el proceso de inferencia en la Unidad de Procesamiento de Visión *Myriad 2*.

Las ilustraciones 4.87 a 4.111 muestran los resultados obtenidos en este último dispositivo, estos se incluyen en la tabla 4.16 y 4.17.

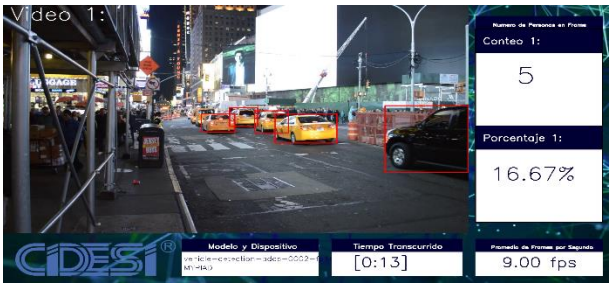


Ilustración 4.87: Conteo Vehículos - VPU - 1 Video.

Ilustración 4.88: Conteo Vehículos - VPU - 2 Video.

Ilustración 4.89: Conteo Vehículos - VPU - 3 Video.

Ilustración 4.90: Conteo Vehículos - VPU - 4 Video.

Ilustración 4.91: Conteo Vehículos - VPU - 5 Video.

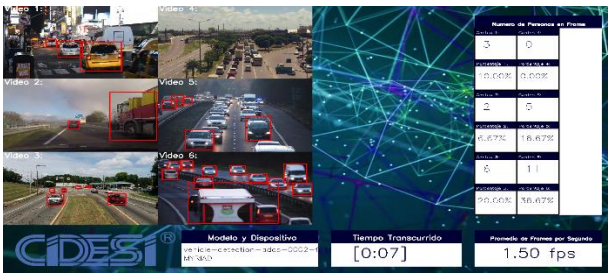


Ilustración 4.92: Conteo Vehículos - VPU - 6 Video.

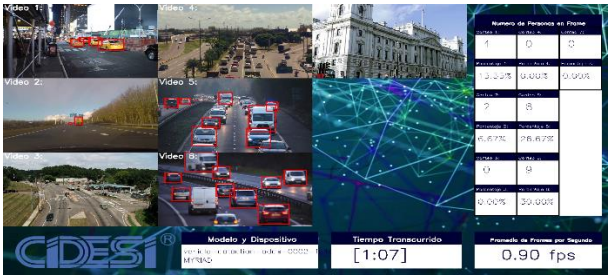


Ilustración 4.93: Conteo Vehículos - VPU - 7 Video.

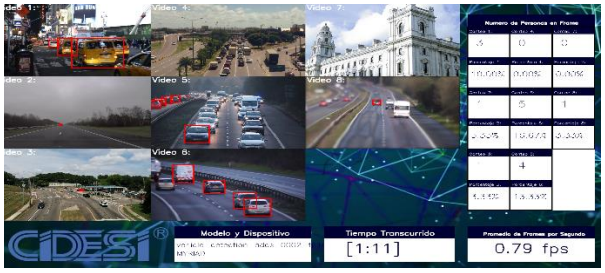


Ilustración 4.94: Conteo Vehículos - VPU - 8 Video.

Ilustración 4.95: Conteo Vehículos - VPU - 9 Video.

Ilustración 4.96: Conteo Vehículos - VPU - 10 Video.

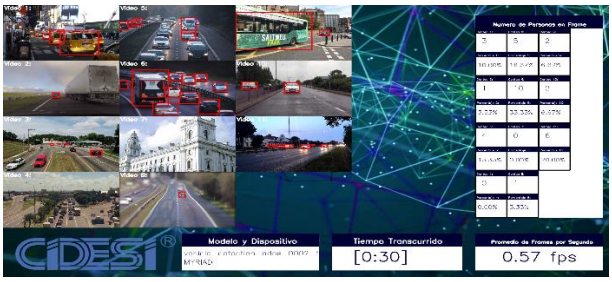


Ilustración 4.97: Conteo Vehículos - VPU - 11 Video.

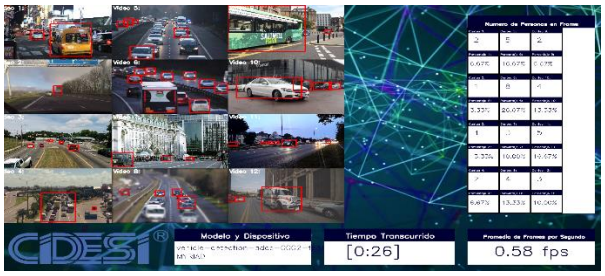


Ilustración 4.98: Conteo Vehículos - VPU - 12 Video.

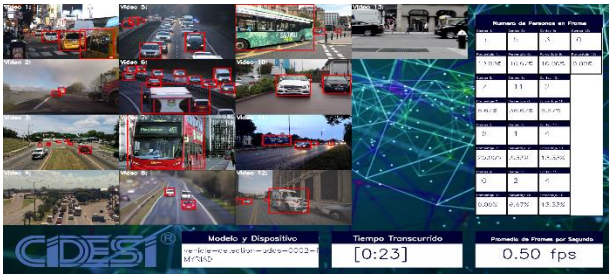


Ilustración 4.99: Conteo Vehículos - VPU – 13 Video.

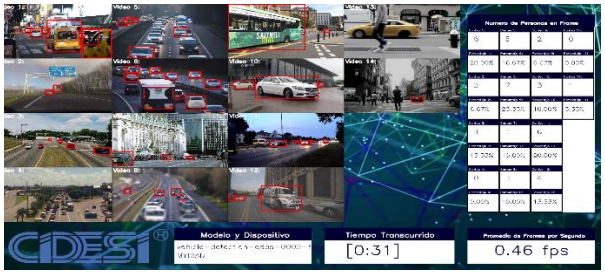


Ilustración 4.100: Conteo Vehículos - VPU - 14 Video.

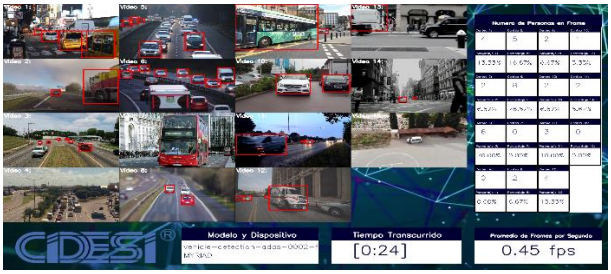


Ilustración 4.101: Conteo Vehículos - VPU - 15 Video.

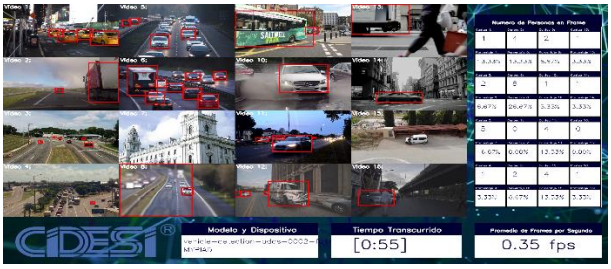


Ilustración 4.102: Conteo Vehículos - VPU - 16 Video.

Ilustración 4.103: Conteo Vehículos - VPU - 17 Video.

Ilustración 4.104: Conteo Vehículos - VPU - 18 Video.

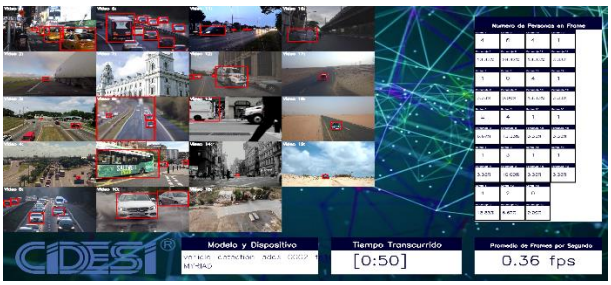


Ilustración 4.105: Conteo Vehículos - VPU - 19 Video.

Ilustración 4.106: Conteo Vehículos - VPU - 20 Video.

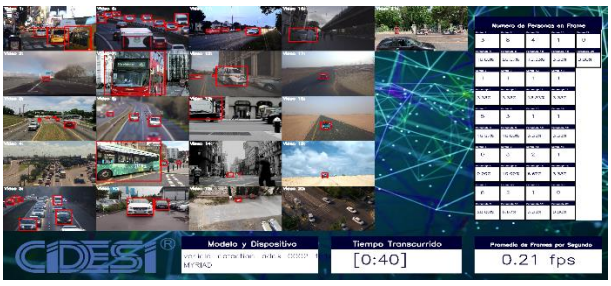


Ilustración 4.107: Conteo Vehículos - VPU - 21 Video.

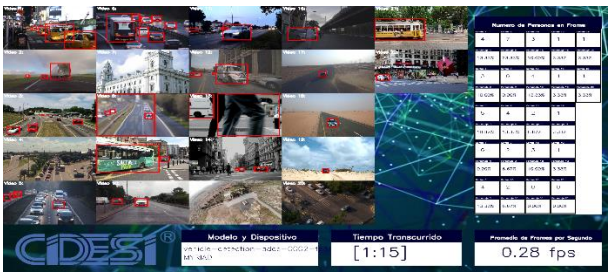


Ilustración 4.108: Conteo Vehículos - VPU - 22 Video.

Ilustración 4.109: Conteo Vehículos - VPU - 23 Video.

Ilustración 4.110: Conteo Vehículos - VPU - 24 Video.

Ilustración 4.111: Conteo Vehículos - VPU - 25 Video.

Num. de Entrada	Promedio de FPS	Num. de Entrada	Promedio de FPS
1	9.00	**10**	0.87
2	4.49	**11**	0.57
3	2.98	**12**	0.58
4	2.23	**13**	0.50
5	1.78	**14**	0.46
6	1.50	**15**	0.45
7	0.90	**16**	0.35
8	0.79	**17**	0.39
9	0.88	**18**	0.39

Tabla 4.16: FPS en conteo de Vehículos desde CPU Zotac (a).

Num. de Entrada	Promedio de FPS	Num. de Entrada	Promedio de FPS
19	0.36	**23**	0.25
20	0.26	**24**	0.27
21	0.21	**25**	0.24
22	0.28	**26**	-

Tabla 4.17: FPS en conteo de Vehículos desde CPU Zotac (b).

4.3.4 Comparación de Rendimiento en Conteo de Vehículos.

Al finalizar las pruebas del programa, se tienen 75 datos referente a la velocidad de frames por segundo, todos estos datos los cotejamos de las tablas 4.14 a 4.17. Si bien este segmento del capítulo que respecta al conteo de vehículos repite lo realizado con el conteo de peatones, pero estas pruebas se centran en el incremento de datos de entradas, ya que se incluyen en pantalla hasta 25 videos de manera simultánea.

El incluir un gran número de videos permite conocer las capacidades de los dispositivos, ya que a medida que se incrementa el número de entradas, las velocidad en los tres dispositivos comienza a disminuir hasta a llegar a un mínimo, al llegar a este número el programa produce un error ya que las especificaciones de este son inferiores al proceso que se está realizado, esto sucede con el equipo Dell, a pesar que el equipo concentra todos los recursos en el proceso al realizar la inferencia de 23 videos, este manda una leyenda en la terminal indicando un error. El límite de los otros equipos ya no se corroboró, ya que la interfaz al exceder el número de 25 videos, las pantallas donde se visualizan los videos toman dimensiones pequeñas que dificultan al operador entender los acontecimientos en el área.

Con el fin de facilitar la lectura de resultados, los valores de la tabla 4.18 son incluidos en la gráfica de la ilustración 4.112, en donde se observa como el *VPU* obtiene mejores resultados al incrementar el número de entradas en el programa.

Num. de Videos de Entrada	FPS Promedio en Dell	FPS Promedio en Zotac	FPS Promedio en Movidius NCS
1	4.09	9.84	9.00
2	1.76	3.16	4.49
3	0.81	2.25	2.98
4	0.44	1.08	2.23
5	0.27	0.85	1.78
6	0.16	0.55	1.50
7	0.14	0.42	0.90
8	0.14	0.31	0.79
9	0.10	0.27	0.88
10	0.09	0.23	0.87
11	0.08	0.19	0.57
12	0.07	0.16	0.58
13	0.05	0.13	0.50
14	0.05	0.12	0.46
15	0.05	0.11	0.45
16	0.04	0.09	0.35
17	0.04	0.08	0.39
18	0.04	0.07	0.39
19	0.03	0.07	0.36
20	0.03	0.06	0.26
21	0.01	0.05	0.21
22	0.01	0.05	0.28
23	0.00	0.05	0.25
24	0.00	0.03	0.27
25	0.00	0.04	0.24

Tabla 4.18: Comparación de FPS en conteo de Vehículos

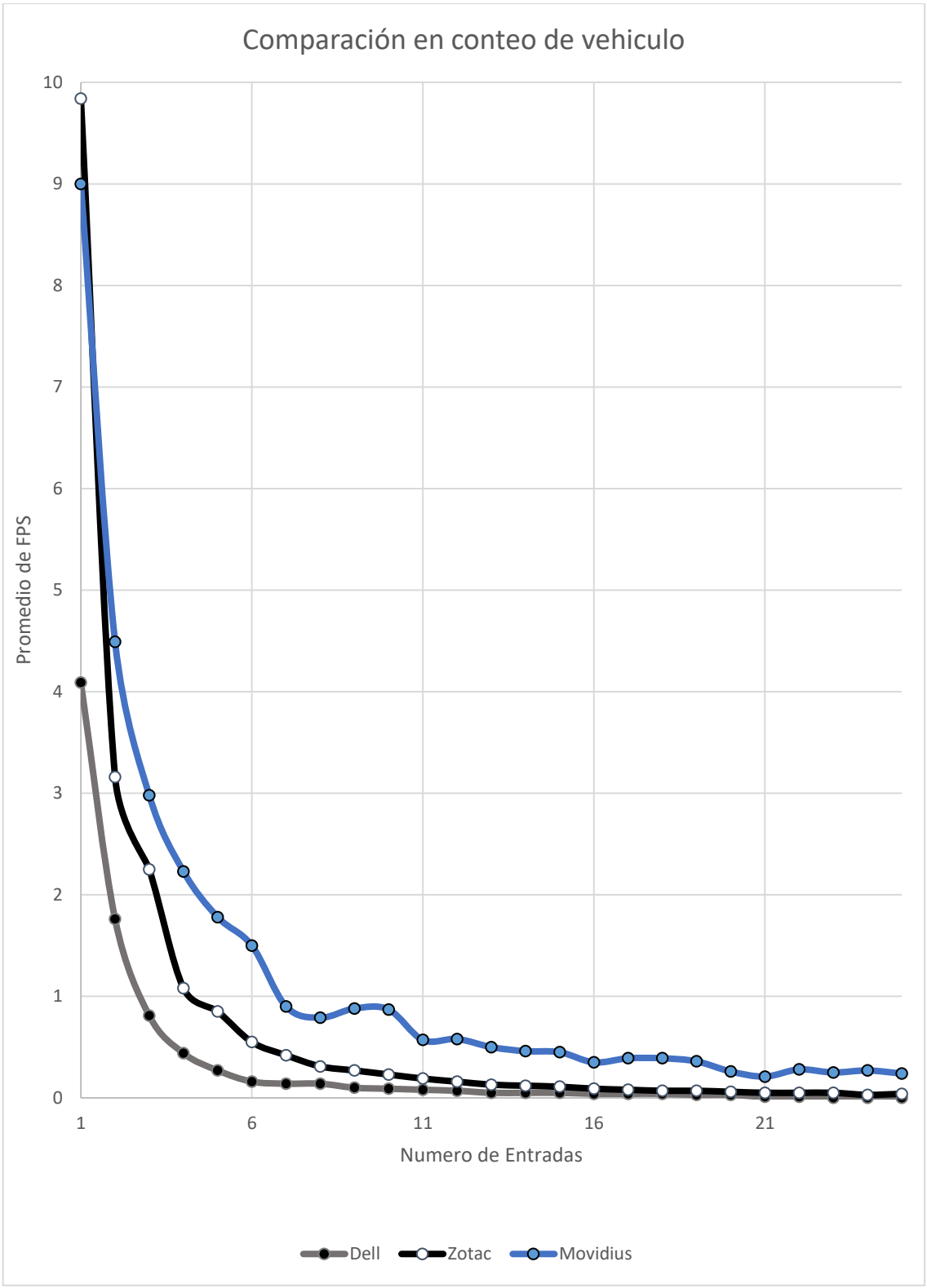


Ilustración 4.112: Grafica Comparativa de rendimiento en Metro

Capitulo 5: Conclusiones.

5.1 Conclusiones.

Los resultados presentados en el capítulo 4, muestran el comportamiento del programa mediante una interfaz amigable al operador cumpliendo con los objetivos del proyecto, ya que el proyecto entrega un *software* en el cual permite al usuario visualizar en un mismo equipo el conteo de personas o autos en distintas áreas (distintos videos) haciendo uso de un módulo computacional neuronal de nombre *Movidius™ Neuronal Compute Stick*.

El programa realiza un proceso de inferencia, el cual hace uso de la mayor parte de los recursos con los que cuenta la *CPU* de los equipos. *Movidius™ NCS* permite que el proceso se centre dentro de su arquitectura, Myriad 2 es una unidad de procesamiento por visión centrado en el proceso de inferencia que permite realizar el proceso de los datos de entrada de manera paralela, teniendo una respuesta más rápida que la obtenida en otros equipos, la ilustración 4.112 muestra como este proceso es incluso 6 veces más rápido que en una computadora con un procesador i5 de 5ta generación al trabajar con 25 videos.

Al realizar la inferencia en la *CPU* de los equipos se observó cómo el proceso consume todos los recursos de este dispositivo, elevando su temperatura. Al usar *Movidius™ NCS* se corrige este problema ya que este dispositivo es de bajo consumo.

Los sistemas C3 y C4 hacen uso de equipo de altas capacidades, así que implementar el proyecto en estas áreas con el uso de *Movidius™ NCS* reduciría los costos del sistema, ya que los equipos donde se ejecuta el programa no requieren de altas especificaciones, basta con el uso del módulo que mejora rendimiento en aplicaciones de visión para la detección de personas y vehículos, con un costo no mayor a cien dólares. La precisión de la detección que entrega como respuesta depende del modelo que se use, ya que en ciertos ángulos o a ciertas distancias de la cámara, el modelo usado no logra identificar los objetos

El proyecto tiene apertura a mejoras, que se especifican en el capítulo 6.

6 Capitulo 6: Proyectos Futuros.

6.1 Introducción.

Este capítulo se plantea algunas de las mejoras que se pueden realizar al proyecto, que pretenden mejorar el rendimiento o bien mejorar el funcionamiento.

6.2 Uso de dos o más Movidius NCS.

Como indican los resultados, el dispositivo externo *Movidius™ NCS* entrega el mejor resultado, en cuestión de rendimiento, *OpenVINO™* ofrece la posibilidad de realizar la inferencia de un modelo dentro de diferentes *Movidius™ NCS*, así que la adquisición de un segundo dispositivo de estos, como muestra la ilustración 6.1, aparentemente entregar mejores resultados, debido a que cada módulo tiene 12 *SHAVEs*, la inclusión de varios módulos pretende acelerar aún más las redes neuronales.

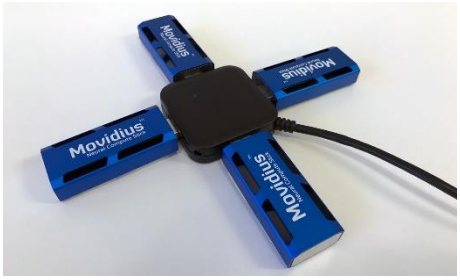


Ilustración 6.1: Inferencia en múltiples dispositivos.

Hacer la ejecución del proyecto con diferentes videos de entrada con el objetivo de corroborar el rendimiento, del uso de más de un *Movidius™ NCS*, para que esto se realice de manera correcta se deben hacer cambios en el código, en librerías y código fuentes.

6.3 Uso de Intel Neuronal Compute Stick 2.

A mediados del presente año, salió a la venta una nueva versión de *Movidius™ NCS* con el nombre de *Intel® NCS 2*, ilustración 6.2, este dispositivo puede ser implementado de la misma manera que su antecesor, sin realizar ninguna modificación en el código.

Ilustración 6.2: NCS 2.

Mediante ilustración 6.3 de la página de MovidiusTM, el fabricante indica que el rendimiento de este dispositivo es hasta 6 veces mejor en el proceso de detección de objeto que el dispositivo *Movidius^{TM} NCS* usado en el proyecto, si se adquiere este dispositivo corroborar su rendimiento mediante la ejecución del programa con múltiples videos de entrada.

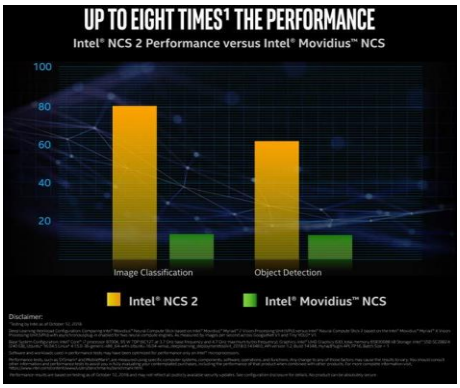


Ilustración 6.3: Comparación de rendimiento de ambos módulos de aceleración neuronal.

6.4 Detección con dos modelos, peatones y vehículos.

Realizar modificaciones en el código principal (*main*) del programa, el cual permita la lectura de dos modelos de representación intermedia distintos, la idea es poder realizar la detección de autos y personas dentro de un *frame* para posteriormente presentar el conteo de estos objetos.

La ilustración 6.4 presenta la idea del programa, que es la generación de las *bounding boxes* en ambos objetos.

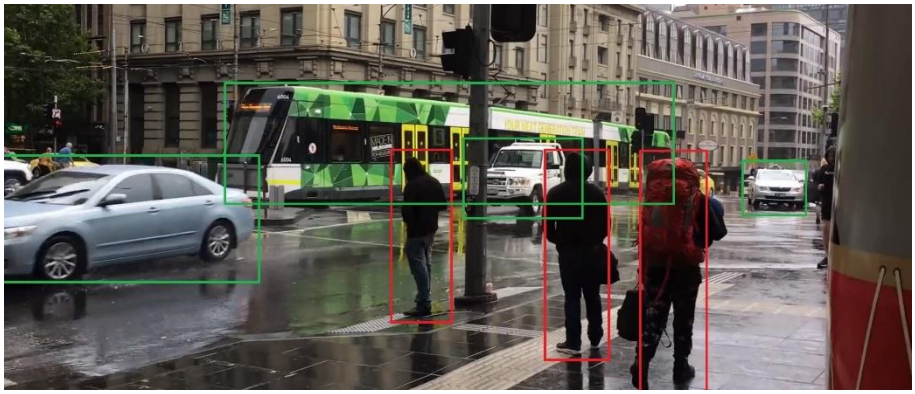


Ilustración 6.4: Detección de Peatones y Vehículos.

Al usar dos modelos se podrá realizar el proceso de inferencia en dos dispositivos, o el mismo dispositivo, ya sea la detección de peatones en el *VPU Myriad*, y la detección de peatones en el *CPU* o los dos modelos de detección se realiza la inferencia dentro de *Myriad*, realizar las pruebas para verificar cual combinación ofrece el mejor desempeño.

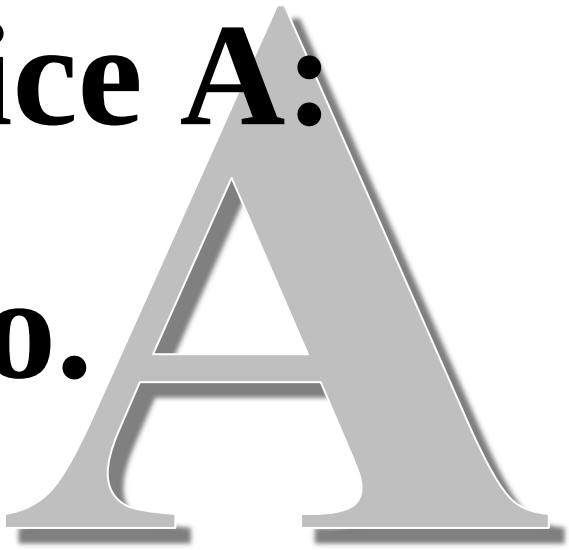
6.5 Creación de modelos y aplicación del Model Optimizer.

Al observar las ilustraciones de los resultados se encuentran frames donde los peatones o vehículos no son detectados, debido a la posición o la distancia a la que se encuentra, esto puede mejorado mediante el entrenamiento de modelos con *frameworks* populares como caffe model, *TensorFlow* para la creación de modelos de *Deep Learning*. Esto permitiría una mejor precisión.

Una vez creado el modelo, hacer uso del *Model Optimizer*, tema 2.2.1, para que sea compatible con en la implementación del programa y tener una interfaz con una detección de mayor precisión.

Apendice A:

Glosario.



Glosario.

Inferencia	Se refiere al proceso de hacer predicciones mediante la aplicación del modelo ya entrenado a ejemplos no etiquetados. [1]
Entrenamiento	Se refiere a cuando el sistema aprende o se ajusta de los datos existentes de entrenamiento, mediante la aplicación del modelo ya entrenado. [1]
Frameworks	Herramienta para el desarrollo de modelos en el área de visión como *Caffe* o *Tensor Flow*.
Workloads	Carga de trabajo
Workflow	Flujo de trabajo representa tanto el modelado como la automatización de un proceso.
Modelo	Define la relación entre características y etiquetas, usado en dos fases: Entrenamiento e Inferencia [1]
Deep Learning	Es el campo que permite a la máquina aprender por sí sola de los errores y la información que recibe
Machine Learning	Aprendizaje Automático, subcampo de la inteligencia artificial que proporciona a los ordenadores la capacidad de aprender sin ser explícitamente programados, es decir, sin que necesiten que el programador indique las reglas que debe seguir para lograr su tarea, sino que las hace automáticamente. [1]
Inference Engine	Máquina de Inferencia, etapa donde se realiza el proceso de inferencia
On/At The Edge	Proceso realizado en tiempo real.
Big Data	Análisis de grandes volúmenes de datos.
Inteligencia Artificial	Campo muy amplio que abarca muchas áreas del conocimiento relacionadas con el aprendizaje automático; incluso se incluyen muchos más enfoques no siempre catalogados como aprendizaje automático. Subdivida en diversos campos para un mejor enfoque [1]

Apendice B: Nomenclatura.

Nomenclatura.

SDK	*Software Development Kit* (Kit de Desarrollo de Software)
NC	*Neuronal Compute* (Computacional Neuronal)
NCS	*Neuronal Compute Stick* (Módulo de Computacional Neuronal)
NCSDK	*Neuronal Compute Software Development Kit* (Paquete de Desarrollo de Software Computacional Neuronal)
VPU	*Vision Processing Unit* (Unidad de Procesamiento de Visión)
GPU	*Graphics Processing Unit* (Unidad de Procesamiento Gráfico)
CPU	*Central Processing Unit* (Unidad Central de Procesamiento)
VINO	*Visual Inferencing and Neuronal Network Optimization* (Inferencia Visual y Optimizador de Redes Neuronales)
DL	*Deep Learning* (Aprendizaje Profundo)
NCS	*Neuronal Compute Stick* (Módulo de Computación Neuronal)
API	*Application Programming Interface* (Interfaz de Programación de Aplicaciones)
SDD	*Single Shoot MultiBox Detector* (Detector de Múltiples Cajas en un solo disparo)
CNN	*Convolution Neuronal Networks* (Redes Neuronales Convolucionales)
R-CNN	*Regions with CNN features*
IR	*Intermediate Representation* (Representación Intermedia)
LPDDR3	*Low Power Double Data Rate 3*
DRAM	*Dynamic RAM* (RAM Dinámico)
RAM	*Random Access Memory* (Memoria de Accesso Aleatorio)
SHAVE	*Streaming Hybrid Architecture Vector Engine*
VLIW	*Very long instruction word*
ROI	*Region of Interest* (Región de Interés)

Referencias Bibliografía

[1] Torres J. (2018). *Deep Learning Introducción prácticas con Keras Primera Parte*. Barcelona, España. WATCH THIS SPACE

[2] https://www.youtube.com/watch?v=sRYs0dZLXkw&feature=youtu.be 24 de Junio

[3] https://movidius.github.io/blog/ 25 de Junio

[4] http://docs.openvinotoolkit.org/latest/_docs_Pre_Trained_Models.html 26 de Junio

[5] https://software.intel.com/en-us/openvino-toolkit/documentation/pretrained-models 26 de Junio

[6] http://docs.openvinotoolkit.org/latest/_docs_Pre_Trained_Models.html 26 de Junio

[7] https://master-deeplearning.com/origen-deep-learning/ 23 de Septiembre

[8] https://docs.openvinotoolkit.org/latest/index.html 23 de Septiembre

[9] https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html 25 de Septiembre

[10] https://docs.openvinotoolkit.org/latest/_docs_optimization_guide_dldt_optimization_guide.html 1 de Octubre

[11] https://docs.openvinotoolkit.org/latest/_docs_IE_DG_supported_plugins_Supported_Devices.html 1 de Octubre

[12] https://movidius.github.io/ncsdk/ncs.html 1 de Octubre

[13] https://uploads.movidius.com/1463156689-2016-04-29_VPU_ProductBrief.pdf 1 de Octubre

[14] http://grupodesta.com/sistemas-de-monitoreo-c3-y-c4/ 10 de Octubre

[15] https://software.intel.com/en-us/iot/reference-implementations/restricted-zone-notifier 10 de Octubre

[16] https://software.intel.com/en-us/iot/reference-implementations/people-counter-system 10 de Octubre

[17] https://software.intel.com/en-us/iot/reference-implementations/parking-lot-tracker 10 de Octubre

[18] https://docs.openvinotoolkit.org/latest/_docs_IE_DG_inference_engine_intro.html 10 de Octubre