

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA

DEPARTAMENTO DE ELECTRÓNICA

VALPARAÍSO - CHILE



**DETECCIÓN AUTOMÁTICA DE ROSTROS EN VIVO
PARA SISTEMAS DE VIDEOCONFERENCIA**

CAMILO DONOSO COLLAO

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELECTRÓNICO

PROFESOR GUÍA: FERNANDO AUAT CHEEIN

PROFESOR CO-REFERENTE: AGUSTÍN GONZÁLEZ VALENZUELA

DICIEMBRE-2022

AGRADECIMIENTOS

Quisiera agradecer primero a mi familia, a mi abuela Ana por darme fuerzas para seguir adelante, a mi madre Jacqueline quien me ha apoyado a lo largo de toda la carrera, a mi padre Daniel por siempre estar presente, a mi hermana María José por levantarme siempre en las épocas mas oscuras, a mis tías y tíos, a mis primas, a mi cuñado, a mis amigos y mi pareja, a todas esas personas que me han acompañado durante este largo proceso, y que sin ellas y ellos nada de esto sería lo mismo.

No tengo palabras para expresar todo el agradecimiento que tengo por el apoyo que me han dado, no ha sido fácil para mi todo este proceso pero espero poder seguir adelante con todo lo que viene y espero poder retribuir todo el cariño recibido.

Detección Automática de Rostros en Vivo para Sistemas de Videoconferencia

Camilo Donoso Collao

Memoria para Optar al Título de Ingeniero Civil Electrónico, Mención en Computadores.

Profesor guía: Dr. Fernando Auat Cheein

Diciembre 2022

Resumen

En el presente trabajo se desarrolló un algoritmo para detección de rostros en interfaces de videoconferencia utilizando redes neuronales con el fin de recopilar vídeos de los rostros presentes en una reunión por videollamada, el algoritmo debe ser capaz de hacer seguimiento a los rostros presentes y almacenarlos por separado. Uno de los objetivos principales es analizar el comportamiento de los recursos computacionales al implementar el algoritmo, evaluando la variación en utilización de CPU, memoria RAM e inclusive GPU en algunos de los casos de prueba, en relación a la cantidad de rostros presentes en pantalla, a la vez de medir el tiempo utilizado por cada iteración del algoritmo.

En vista de que la componente principal del algoritmo, correspondiente al detector de rostros, es desarrollado con redes neuronales, es que se presenta el estudio realizado en la búsqueda de una red que cumpla con las funciones especificadas y que pueda ser entrenada con la cantidad de recursos computacionales disponibles por el autor, las cuales están detalladas dentro de este trabajo. Además se enseña las métricas de evaluación de la red neuronal a utilizar para determinar qué tan bien se desempeña, incluyendo el detalle de las pruebas a realizar con el algoritmo, comparando diferentes redes neuronales, y uso de solo CPU o implementación de GPU como dispositivo de cómputo.

Los resultados finales representan dos aspectos importantes del trabajo realizado, primero el desempeño del algoritmo en relación a qué tan bueno es reconociendo rostros, siendo como parámetros a medir la cantidad de rostros reales reconocidos correctamente, cuantos rostros faltó por detectar, y cuántas falsas detecciones fueron realizadas. El segundo corresponde al uso de recursos computacionales respecto a la cantidad de rostros detectados, para así determinar si una mayor cantidad de rostros implica un mayor consumo de recursos y cómo definir dicha relación.

Palabras claves: Redes neuronales, detección de rostros, recursos computacionales.

Live Automatic Face Detection for Videoconferencing Systems

Camilo Donoso Collao

Final Project Report towards the fulfillment of the Ingeniero Civil Electrónico, Minor in Computers degree.

Advisor: Dr. Fernando Auat Cheein

December 2022

Abstract

In this thesis, a algorithm for face detection in video conference interfaces was developed using neural network with the purpose of collect face videos of the attendants at the meeting. The algorithm it must be capable of made face tracking of the attendants and save this faces separately. One of the main target is analyze the behavior of the computational resources at implement the algorithm, assessing the measures of CPU, RAM memory, and inclusive GPU in some cases of test in connection with the quantity of faces in the screen. At the same time, It's important to measure the execution time per iteration of the algorithm.

Additionally, in view of that the main component in the algorithm for face detection it's made with neuronal network, the study of finding a neuronal network that satisfied the function specifications and that could be trained using the computational resources provide by the author, which are detailed on this report, shall be presented in this work. On top of that, the neuronal network evaluation metrics are showed to determine the performance, including the detail of the algorithm test, comparing between different neuronal network, and use of only CPU or adding GPU as compute device.

Finally, the result show two relevant aspects. First, the performance of the face recognition algorithm, using as parameters the quantity of well recognised faces, and the not detected, and false faces detected. Second, the analysis between the computational resources and the quantity of faces detected, defining if more faces it implied more use of resources and how many would be necessary in the application.

Keywords: Neural networks, face detection, computer resources.

Glosario

- SSD: Single Shot MultiBox Detector.
- SSD 320: SSD MobileNet V2 FPNLite 320x320
- SSD 640: SSD MobileNet V2 FPNLite 640x640+
- CPU: Unidad central de procesamiento, parte del hardware del computador que realiza operaciones específicas.
- RAM: Memoria de almacenamiento a corto plazo del computador, almacena de forma temporal programas y procesos que están en ejecución.
- GPU: Unidad de procesamiento gráfico, parte del hardware del computador encargada de cooperar en el procesamiento de gráficos y se puede utilizar también en cálculos relacionados a redes neuronales.

Índice

0.1	Objetivos	ix
0.2	Trabajos desarrollados por otros autores	ix
1	Descripción del problema	1
2	Desarrollo de la solución	3
2.1	Captura de pantalla	5
2.2	Detector de rostros	6
2.2.1	Redes neuronales	6
2.2.2	Redes neuronales convolucionales	10
2.2.3	Selección de datos de entrenamiento	11
2.2.4	Entrenamiento	13
2.2.5	Evaluación	15
2.3	Seguimiento de rostros	18
2.4	Interfaz de usuario	22
3	Evaluación del sistema	24
4	Resultados	28
4.1	Desempeño del algoritmo	28
4.1.1	SSD MobileNet V2 FPNLite 320x320 CPU	29
4.1.2	SSD MobileNet V2 FPNLite 320x320 GPU	30
4.1.3	SSD MobileNet V2 FPNLite 640x640 CPU	31
4.1.4	SSD MobileNet V2 FPNLite 640x640 GPU	32
4.2	Uso de recursos computacionales	33
4.2.1	Utilización de CPU	33
4.2.2	Utilización de GPU	36
4.2.3	Utilización de memoria RAM	36

4.2.4	Tiempo promedio por iteración	38
4.3	Resultados generales	41

Lista de Figuras

2.1	Flujo de datos general	4
2.2	Ventana mostrando la captura de pantalla	5
2.3	Red neuronal artificial	6
2.4	Neuronas Animales	7
2.5	Red neuronal computacional	7
2.6	Ejemplos de funciones de activación	8
2.7	Ejemplo red neuronal computacional	9
2.8	Optimización por gradiente descendiente	10
2.9	Estructura de una red neuronal convolucional	10
2.10	Estructura de una red neuronal SSD	11
2.11	Ejemplo de resultado de entrenamiento con primer set de datos	12
2.12	Imagen de ejemplo de primer set de datos	12
2.13	Ejemplo de resultado con segundo set de datos	13
2.14	Función de pérdida en entrenamiento	14
2.15	Curva Precisión y Exhaustividad de ejemplo	16
2.16	Ejemplo con red neuronal SSD MobileNet V2 FPNLite 320x320	17
2.17	Ejemplo con red neuronal SSD MobileNet V2 FPNLite 640x640	18
2.18	Sistema de coordenadas en una imagen	19
2.19	Intersección de recuadros	20
2.20	Ajuste de recuadros iniciales para seguidor	21
2.21	Interfaz de usuario	23
3.1	Cambio de tamaño en imágenes con rostro para el set de datos de evaluación .	26
3.2	Selección de rostros para creación set de datos	26
3.3	Imagen de ejemplo del set de datos creado	27
4.1	Desempeño SSD MobileNet V2 FPNLite 320x320 CPU	29

4.2	Desempeño SSD MobileNet V2 FPNLite 320x320 GPU	30
4.3	Desempeño SSD MobileNet V2 FPNLite 640x640 CPU	31
4.4	Desempeño SSD MobileNet V2 FPNLite 640x640 GPU	32
4.5	Utilización de CPU	34
4.6	Utilización de GPU	35
4.7	Utilización de memoria RAM	37
4.8	Tiempo promedio por iteración	40

Introducción

Es parte inherente del ser humano como ser social el lograr comunicarse con otros seres humanos, donde en el proceso se puede lograr identificar las emociones de otra persona y así lograr entender aunque sea superficialmente el estado emocional del otro. Este trabajo nace de la necesidad de identificar las emociones de las personas en interfaces de videoconferencia utilizando solamente la detección de rostros y para conseguirlo en esta memoria se usó redes neuronales artificiales.

0.1 Objetivos

El objetivo principal de esta memoria es detectar rostros en interfaces de videoconferencia, haciendo seguimiento a cada rostros y almacenándolos en vídeos por separado.

Los objetivos secundarios son:

- Definir un límite de rostros posibles a detectar por el programa desarrollado.
- Caracterizar el uso de recursos computacionales en relación al desempeño del detector de rostros. Comparando diferentes redes neuronales, y el uso de solo CPU o complemento de GPU para cálculos computacionales.

0.2 Trabajos desarrollados por otros autores

A continuación se presenta los trabajos realizados recientemente sobre la detección de rostros:

- El primer trabajo a analizar corresponde a TinaFace [1], el cual utiliza una red neuronal para detección de rostros, sosteniendo que para lograr el reconocimiento facial se puede utilizar herramientas orientadas al reconocimiento de objetos en general. Utiliza una red neuronal ResNet50 [2] como base para generar una nueva estructura, la cual logra una precisión del 92.1% en el set de datos Wider Face [3]

- Multilevel Single Stage Network for Face Detection (MSNFD) es un método que propone mejorar la detección de rostros bajo tres aspectos principales. Primero establece una estructura multinivel para la red neuronal a implementar, segundo añade una mejora de funciones para la recolección de información y tercero implemento dos pasos en relación a la función de pérdida de la red neuronal. Este algoritmo también fue probado en el set de datos Wider Face.
- YOLOv3: Face Detection in Complex Environments [4], es un trabajo donde se presenta el uso de una red neuronal YOLOv3 [5] para la detección de rostros en ambientes complejos. Este trabajo, a diferencia de la implementación original de YOLOv3, se realizan cambios en la reagrupación de cajas, adecuando el método para utilizarlo en el set de datos de Wider Face.
- Face Recognition based Attendance System [6] es un método para reconocimiento de rostro e identificación de persona para tomar asistencia. Se centra en dos etapas, primero detectar rostro y luego hacer el reconocimiento utilizando dlib[7].
- En el artículo Human face recognition based on convolutional neural network and augmented dataset [8], se presenta un método para hacer reconocimiento facial utilizando un set de datos pequeño, combinando redes neuronales convolucionales y aumento de datos.

Capítulo 1

Descripción del problema

Este trabajo consiste en detectar múltiples rostros en sistemas de videoconferencia como Zoom, Meet, Microsoft Teams, entre otros, y medir la cantidad de recursos computacionales utilizados durante el proceso. Es importante que la solución tenga la capacidad de utilizarse en las diferentes plataformas de videoconferencia y sistemas operativos, además el programa debe ser capaz de cuantificar los recursos utilizados por el proceso, a la vez que los resultados deben mostrarse y actualizarse durante la videollamada.

Es relevante conocer la información referente a los recursos utilizados debido a que se necesita saber en que equipos es aplicable definiendo los requerimientos básicos de operación. Por otro lado es necesario verificar la precisión que tiene el algoritmo al momento de detectar rostros y hacer seguimiento de ellos para evaluar el rendimiento de este programa y así otorgar claridad respecto a los límites que este tiene.

El algoritmo desarrollado debe ser capaz de almacenar un vídeo de cada rostro detectado, por lo tanto debe lograr definir que rostros pertenecen a una misma persona, también debe comenzar una grabación cuando un rostro nuevo se presenta durante la videoconferencia, y parar la grabación cuando un rostro deje de ser detectado.

Uno de los objetivos de esta memoria es lograr capturar rostros para generar datos que luego serán usados por otro sistema el cual se utilizará para detectar expresiones faciales, por lo tanto la captura y procesamiento de datos debe ser lo suficientemente rápida para lograr captar las variaciones de movimiento del rostro. Según el informe científico Reading facial expressions of emotion "When single emotions occur and there is no reason for them to be modified or concealed, expressions typically last between 0.5 to 4 seconds and involve the entire face"[Cuando una sola emoción ocurre y no hay razón para que sea modificada o disimulada, las expresiones duran típicamente entre 0.5 a 4 segundos e involucrarán el rostro

completo][9], por lo tanto el programa desarrollado deberá procesar en un tiempo menor a los 0.5 segundos por cada imagen.

Finalmente, un aspecto importante de esta memoria consiste en el manejo del costo computacional, se requiere medir la cantidad de recursos utilizados para la detección de rostros, y definir la cantidad máxima de rostros detectados para una cantidad de recursos computacionales específicos.

Como en este caso es relevante conocer el uso de recursos computacionales es necesario conocer las características del computador utilizado en las pruebas, teniendo las siguientes características:

- Procesador: Intel(R) Core(TM) i7-9750H CPU @ 2.60GHz
- RAM: 16,0 GB (15,8 GB utilizable)
- Tarjeta gráfica dedicada: NVIDIA GeForce GTX 1050 4 GB
- Pantalla: Resolución de 1920 x 1080 píxeles

Capítulo 2

Desarrollo de la solución

Para cumplir con el requerimiento de almacenar cada rostro detectado sin superar la cantidad de recursos disponibles, se planteó el supuesto de que la posición de los rostros no suelen variar frecuentemente en la interfaz de videoconferencia, es decir que la posición de cada persona que está presente en la videoconferencia con la cámara encendida va a estar acotada a un rectángulo, y dicho rectángulo se encontrará en una misma posición durante la videoconferencia. El algoritmo debe concluir que en la serie de imágenes procesadas, si detecta un rostro en un mismo rango de posición, entonces este rostro pertenece a una misma persona, si el rostro que está siendo almacenado deja de ser detectado en la posición correspondiente, entonces se debe detener la grabación de dicho rostro. De esta forma cada vez que se muestre un rostro en cierta posición y se mantenga en el tiempo en esa posición se considerará como un rostro a guardar en vídeo, y la grabación del rostro terminará cuando este deje de ser detectado en dicha posición. Se destaca que el detector no reconoce características faciales para definir que cierto rostro pertenece a la misma persona, solo detecta coincidencias en la posición del rostro, y si el rostro de una persona deja de estar en la misma posición de pantalla entonces se perderá la capacidad de rastreo.

El desarrollo del software se realizó en Python, y se separó en 4 etapas:

1. Crear un capturador de pantalla.
2. Crear un detector de rostros.
3. Crear un algoritmo que haga seguimiento de rostros.
4. Crear una interfaz para el usuario e integrar las etapas anteriores.

El software genera tantos vídeos como rostros haya detectado, además genera un vídeo de la captura de pantalla completa donde se muestran los rostros detectados en un marco verde con un número al lado del recuadro asociado a la probabilidad de que la detección sea un rostro, es decir que si el algoritmo determina que una detección tiene un 80% de probabilidad de ser un rostro, entonces se mostrará este número al lado del recuadro que enmarca la detección.

El flujo general de datos se muestra en la figura 2.1, donde se presentan las etapas del algoritmo desarrollado. Primero se tiene la captura de pantalla, donde se obtendrá las imágenes capturadas de la interfaz de videoconferencia, luego estas imágenes son procesadas por un detector de rostros, el cual retornará un arreglo de datos con las coordenadas en píxeles de cada rostro detectado y un número que representa la probabilidad de que una detección sea un rostro. Posteriormente este arreglo de datos pasa al algoritmo detector de rostros, el cual revisa los datos y determina que rostros coinciden en posición dado un historial de posiciones de rostros, es decir que se almacenan las posiciones de los rostros y se comparan con los datos nuevos de las detecciones realizadas, y así se determine cuales coinciden en posición y se asocian a un mismo rostro, este algoritmo también determina cuales son rostros nuevos y deben comenzar a ser almacenados y que rostros se dejan de mostrar y deben dejar de ser almacenados.

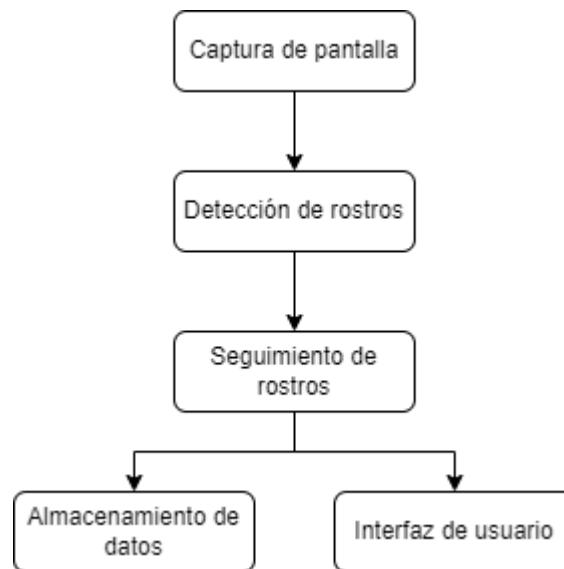


Figura 2.1: Flujo de datos general

2.1 Captura de pantalla

El algoritmo que captura pantalla se desarrolló en Python, y se utilizó el módulo ImageGrab de la librería Pillow [10], este módulo permite capturar la pantalla del computador, opera en Windows, Mac y Linux, y dada su versatilidad fue la opción seleccionada.

El programa desarrollado se puede utilizar con uno o dos monitores conectados al computador, sin embargo, dado las configuraciones realizadas el módulo ImageGrab solo captura imágenes del monitor que esté definido como principal, por lo tanto la interfaz de la videoconferencia debe estar visible en el monitor principal para capturar imágenes útiles para la detección.

Para mostrar las imágenes capturadas se utilizó la librería OpenCV[11], esta librería permite mostrar las imágenes en pantalla y guardar imágenes o videos. En el desarrollo de la captura de pantalla se probó la adquisición exitosa de imágenes en una pantalla y se mostraron dichas capturas, además se midió el tiempo promedio de captura.



Figura 2.2: Ventana mostrando la captura de pantalla

En la figura 2.2 se muestra la interfaz del programa ejecutado, el cual muestra la pantalla del monitor principal. El tiempo promedio de captura de pantalla es de 44 milisegundos, este no incluye el tiempo que demora en mostrar la imagen capturada dado que para esta etapa se requiere solo el tiempo de captura, y se calculó midiendo el tiempo de cada captura durante

500 iteraciones.

Dado que el tiempo de captura es de 44 milisegundos y el tiempo mínimo necesario para formar una expresión facial es de 500 milisegundos[9], se concluye que la captura corresponde a un 8.8% del tiempo máximo de procesamiento, y por lo tanto el algoritmo es válido para las siguientes etapas.

2.2 Detector de rostros

En esta sección se presenta una introducción a las redes neuronales y las etapas desarrolladas para obtener el detector de rostros.

2.2.1 Redes neuronales

El algoritmo detector de rostros se utiliza una red neuronal artificial [12], estas son estructuras basadas en el cerebro humano, pero con la capacidad de ser ejecutadas con algoritmos computacionales. La red consta de diversas unidades llamadas neuronas, unidades o nodos y que se encuentran conectadas entre si, la red recibe datos de entrada y mediante funciones matemáticas y cálculos computacionales retorna datos de salida que pueden corresponder a patrones o valores característicos perteneciente a los datos de entrada [13] como se muestra en la figura 2.3 .

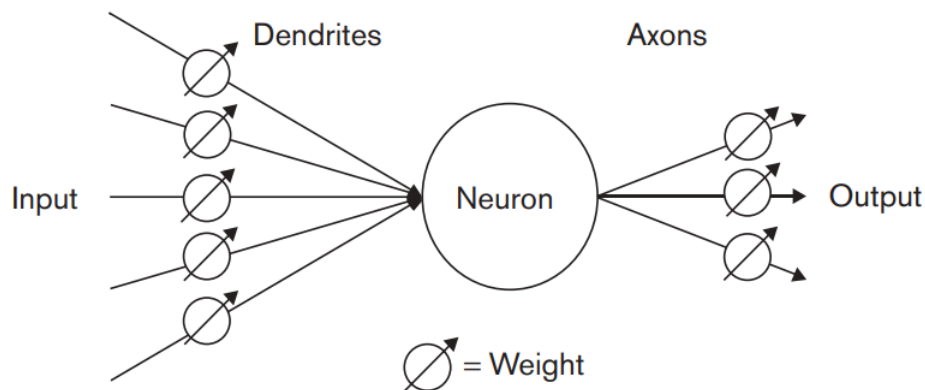


Figura 2.3: Red neuronal artificial

Tal como se muestra en la figura 2.4, el cerebro humano consta de miles de millones de unidades celulares llamadas neuronas, las cuales se conectan entre sí y permiten trasladar señales e información desde y hacia el cerebro.

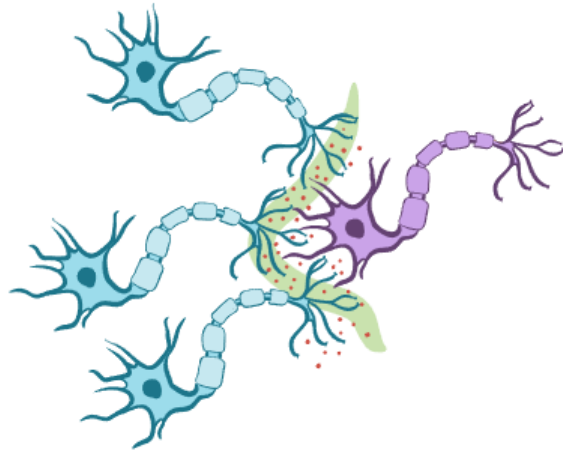


Figura 2.4: Neuronas Animales

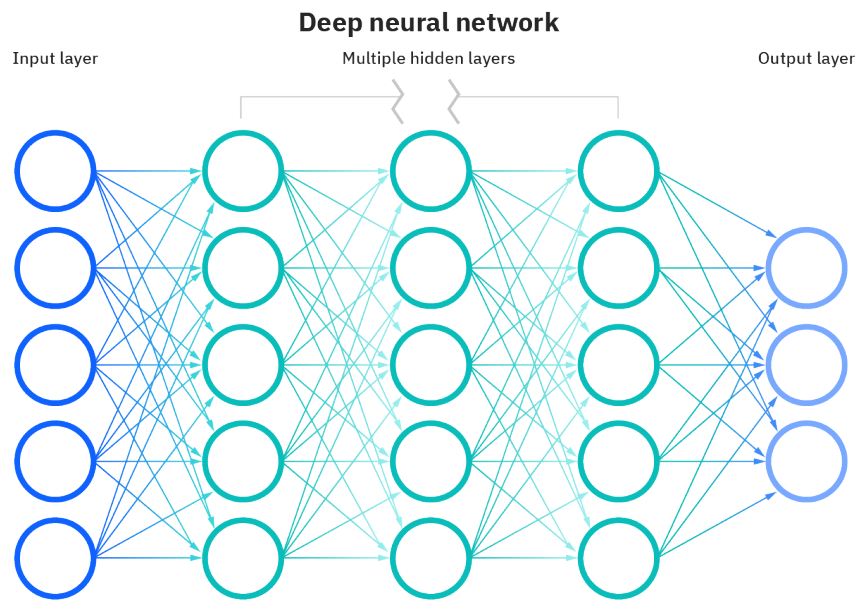


Figura 2.5: Red neuronal computacional

La red neuronal se organiza en capas de una o mas neuronas, la primera es la capa de entrada y recibe los datos que ingresan a la red, luego hay una o mas capas ocultas y finalmente una capa de salida[14], tal como se representa en la figura 2.5. Cada capa tiene neuronas que conectan con las neuronas de la capa siguiente, de izquierda a derecha, no es necesario que todas las capas tengan la misma cantidad de neuronas y no todas las neuronas están conectadas

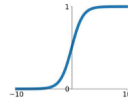
entre sí. Este tipo de organización puede dar origen a estructuras complejas, que pueden estar conformadas por miles de neuronas y decenas de capas.

Cada nodo o neurona de la red tiene información, a los cuales se les denomina pesos y bias, y dada las entradas de la neurona, sus pesos y bias se determina un valor correspondiente a la sumatoria de la multiplicación entre cada entrada y peso la neurona, mas el bias, dicho valor ingresa a una función de activación [15], la cual retorna la salida de la neurona. Existen diferentes tipos de función de activación, algunos se muestran en la figura 2.6, y otorgan elementos no lineales al calculo de la salida de cada neurona.

Activation Functions

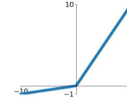
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



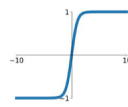
Leaky ReLU

$$\max(0.1x, x)$$



tanh

$$\tanh(x)$$

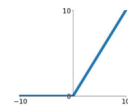


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ReLU

$$\max(0, x)$$



ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

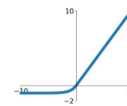


Figura 2.6: Ejemplos de funciones de activación

Durante el procesamiento se reciben datos en la capa de entrada, esta capa utiliza sus neuronas y en relación a los pesos y la función de activación de cada una retornan datos de salida que se dirigen hacia las capas ocultas, estas capas procesan los datos y los propagan hasta la salida de la red, donde se entrega el resultado del proceso, este flujo de datos se denomina Forward Propagation, o propagación hacia adelante.

En la figura 2.7 se muestra una red neuronal que consta de una capa de entrada con dos neuronas, una capa oculta con tres neuronas y una capa de salida con dos neuronas, y en cada neurona se encuentra un arreglo de pesos y bias. Por ejemplo, si se utiliza una función de activación sigmoide en cada neurona, se obtiene la siguiente salida v en la primera neurona de la capa de entrada.

$$v_{11} = \text{sigmoid}(IN_1 \cdot w_{11} + b_{11}) \quad (2.1)$$

Luego, los resultados de las capas de entrada se propagan a la capa oculta, hasta llegar a la capa de salida, donde se retorna el resultado de la red neuronal. Si una neurona tiene N entradas u y por consiguiente tiene N pesos, y una función de activación sigmoide entonces

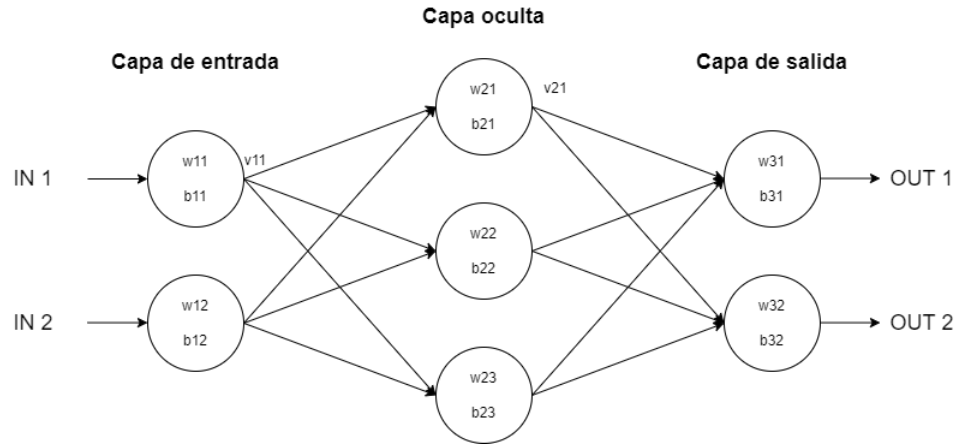


Figura 2.7: Ejemplo red neuronal computacional

la salida v se calcula como:

$$v_j = \text{sigmoid} \left(\sum_{i \in N} (u_i \cdot w_{ji}) + b_j \right) \quad (2.2)$$

Los pesos y bias de la red neuronal se ajustan mediante entrenamiento, en esta etapa la red neuronal recibe un dato, lo procesa en sus capas y entrega un resultado, la salida es comparada con el resultado esperado asociado al dato de entrada y ajusta los pesos y bias, este proceso se denomina "backpropagation"[16] o propagación hacia atrás. La diferencia entre el resultado esperado y la salida de la neurona se calcula mediante la función de pérdida, este valor es utilizado por un optimizador que se encarga de ajustar los parámetros de cada neurona, con el fin de minimizar el error generalizado entre los resultados calculados por la red neuronal y los resultados esperados. Existen diversos optimizadores, uno de los mas conocido es el gradiente descendiente[17], en la figura 2.8 se observa un ejemplo de minimización de una función utilizando este optimizador.

Al utilizar el método de propagación hacia atrás la salida calculada por la red dada una entrada es utilizada para calcular el error entre el resultado esperado y el calculado. La actualización de los pesos de las capas de la red neuronal depende de principalmente de cuatro parámetros, la tasa de aprendizaje, la derivada de la función de activación, el error y la entrada de la neurona, en base a estos datos se calcula los pesos de cada capa desde la salida de la red neuronal y los resultados de estas actualizaciones se propaga hasta la capa de entrada de la red neuronal [16].

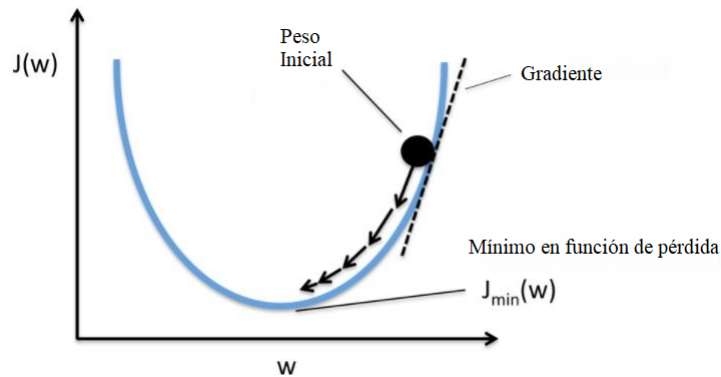


Figura 2.8: Optimización por gradiente descendiente

2.2.2 Redes neuronales convolucionales

Las redes neuronales se pueden especializar para ciertos tipos de datos, una red neuronal convolucional[18] se especializa en reconocer patrones en imágenes, y su estructura está diseñada para procesar matrices de datos. Algunas capas de este tipo de red tienen tres dimensiones, correspondientes al ancho, largo y cantidad de canales de la capa.

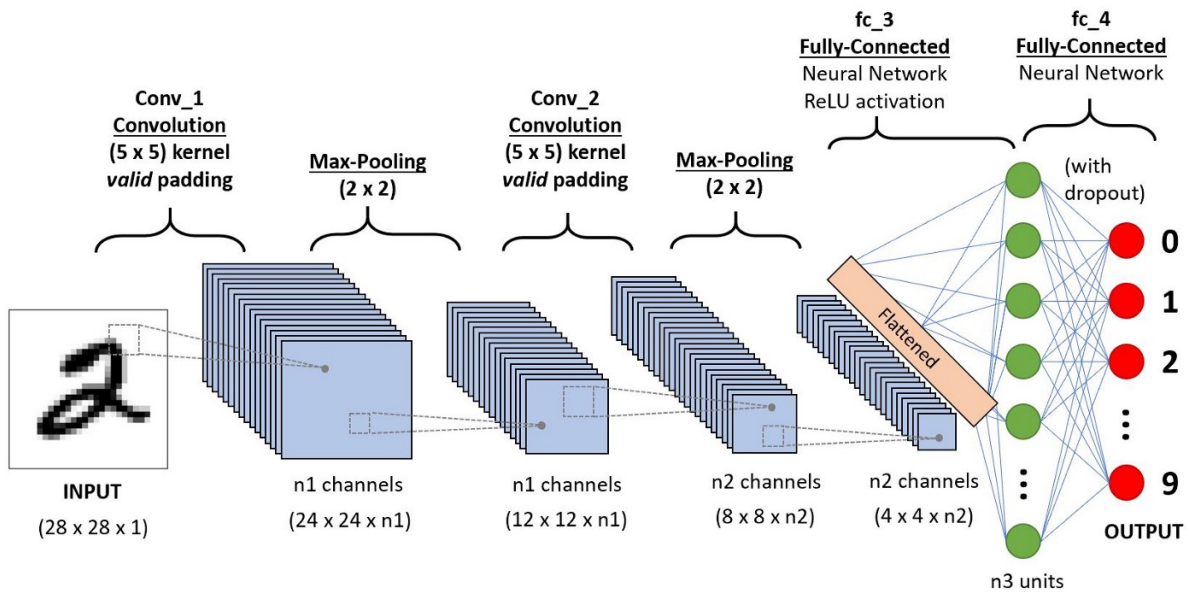


Figura 2.9: Estructura de una red neuronal convolucional

En la figura 2.9 se observa una imagen como entrada a la red neuronal, la cual pasa primero a través de una serie de capas convolucionales, luego pasa a las capas ocultas llamadas "Fully-Connected"[18] o capa totalmente conectada, y finalmente a la capa de salida, la cual no es

una imagen, sino que son números enteros del 0 al 9.

En las redes neuronales dedicadas a clasificación de imagen, la salida usualmente es la probabilidad de que la imagen procesada corresponda a cierta clase, la figura 2.9 corresponde a la clasificación de números, donde la entrada es una imagen con el número dos y a la salida entrega la probabilidad de que la imagen de entrada corresponda a un número del 0 al 9.

Una red neuronal SSD[19] se especializa en la detección y clasificación de objetos, reciben como entrada una imagen y entregan como la salida recuadros donde se encontraron objetos, y la probabilidad de que dicho objeto pertenezca a una clase. En la figura 2.10 se presenta la estructura de una red neuronal SSD, en esta se observan una serie de capas convolucionales, seguidas de una serie de capas ocultas encargadas de la detección de objetos por clase, y la etapa de salida encargada de filtrar dichas detecciones un algoritmo que utiliza el método "Non-Maximum Supresion"[20], el cual elimina recuadros superpuestos redundantes.

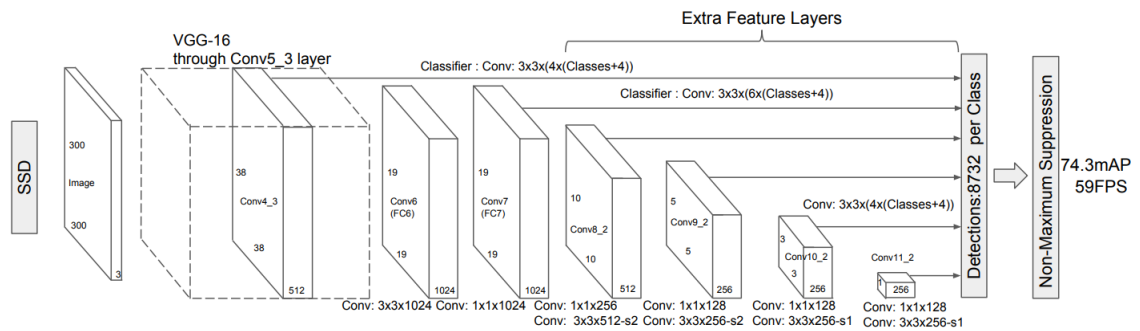


Figura 2.10: Estructura de una red neuronal SSD

2.2.3 Selección de datos de entrenamiento

Una red neuronal requiere de datos para su entrenamiento, dichos datos deben contener la entrada a la red neuronal y la salida esperada. Para el detector de rostros se utilizó una serie de imágenes con detecciones registradas pertenecientes a WIDER FACE dataset[3], este set de datos consta de 32203 imágenes con 393703 caras detectadas, donde se presentan rostros en diferentes contextos, además este set es ampliamente usado como punto de referencia para detección de rostros y por lo tanto se comprueba su importancia en este ámbito. Cabe destacar que durante todas las pruebas realizadas en esta memoria no se utilizó censura en los rostros, es decir que estos no fueron distorsionados, sin embargo en la escritura de este texto se decidió distorsionar los rostros para mantener la privacidad de las personas en las imágenes.

Durante las pruebas de entrenamiento realizadas con este set de datos se determinó que

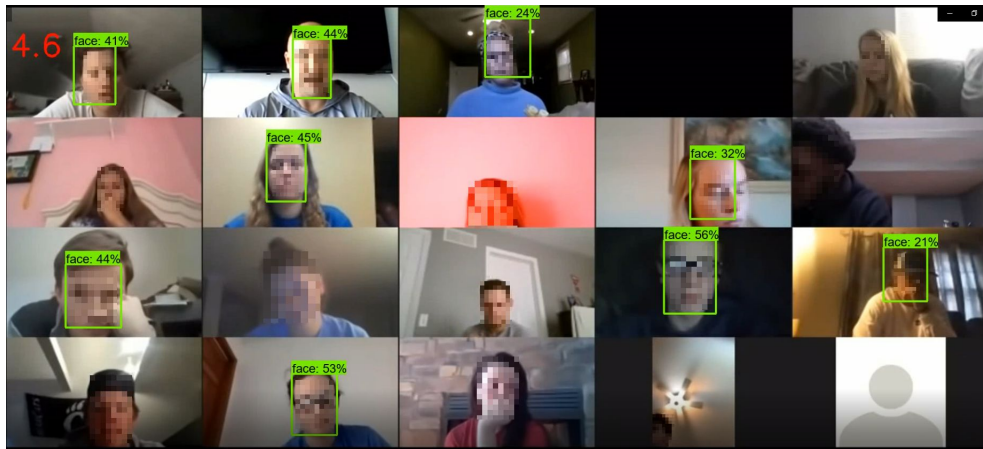


Figura 2.11: Ejemplo de resultado de entrenamiento con primer set de datos

no todas las imágenes eran útiles para el propósito de esta memoria, si bien el entrenamiento convergió a funciones de pérdidas minimizadas, en el rango de 0.5 a 2 (no tiene unidad), al evaluar el detector con imágenes de interfaces de videoconferencias se notó que no todos los rostros eran detectados, y las probabilidades de que cada rostro detectado rondaba el 50% o menos como se observa en la figura 2.11



Figura 2.12: Imagen de ejemplo de primer set de datos

Se estudió el set de datos utilizado para el entrenamiento se observó que algunas imágenes presentaban rostros muy pequeños tal como se muestra en la figura 2.12, por lo tanto para las siguientes pruebas se descartó imágenes que tuvieran una cantidad de detecciones mayores a cierto límite, dicho límite se ajustó hasta lograr una probabilidad superior al 60% para la

detección de cada rostro, y se estableció un máximo de 10 rostros detectados por cada imagen de entrenamiento. En la figura 2.13 se presenta el resultado obtenido con el último set de datos utilizado.

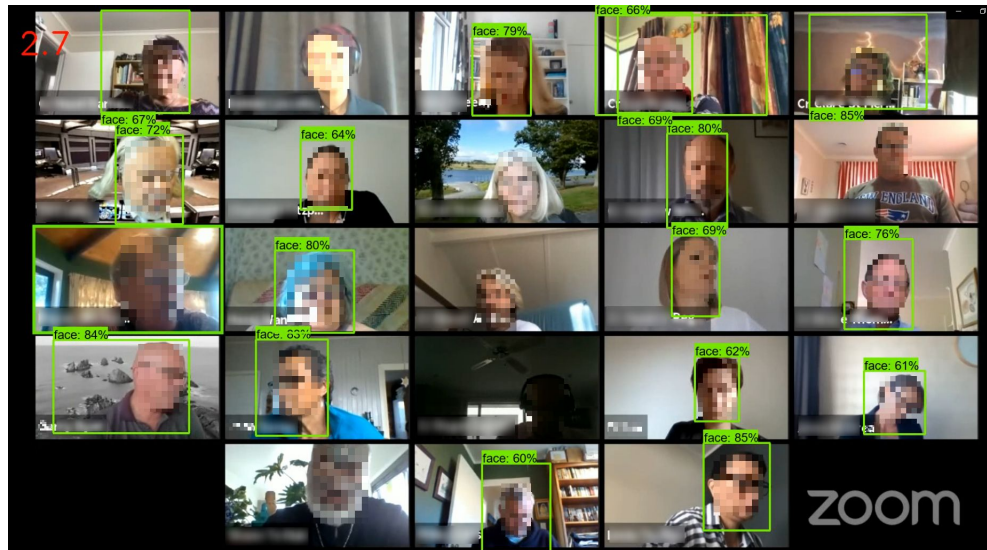


Figura 2.13: Ejemplo de resultado con segundo set de datos

Del total de imágenes disponibles en WIDER FACE se utilizó 10534 para entrenamiento y 3222 para validación.

2.2.4 Entrenamiento

Para el entrenamiento e implementación de la red neuronal se utilizó la librería TensorFlow en Python, y el desarrollo se basó principalmente en el repositorio TensorFlow Model Garden[21]. En dicho repositorio se encuentran diversos ejemplos relacionados a redes neuronales, entre los cuales se encontró una guía para entrenar una red neuronal orientada a detección de objetos.

El entrenamiento comienza con el acondicionamiento de datos, estos deben tener formato TFRecord, correspondiente a una serie de registros que se lee secuencialmente. El siguiente paso consiste en seleccionar y configurar un tipo de red neuronal, donde la colección de redes neuronales se encuentra en la sección TensorFlow 2 Detection Model Zoo[22] del repositorio, y dentro de las cuales se seleccionó 2 redes, SSD MobileNet V2 FPNLite 320x320 desde ahora denominada SSD 320, y SSD MobileNet V2 FPNLite 640x640 desde ahora llamada SSD 640, dichas redes tienen una estructura similar pero trabajan con diferente resolución de imagen.

Durante las pruebas realizadas se analizó el rendimiento del computador y la variable total loss, correspondiente al valor de la función de pérdida de la red neuronal. A continuación se muestra en la tabla 2.1 la comparación de la utilización de recursos de cada red neuronal durante el entrenamiento, se considera el porcentaje de uso de CPU, la cantidad de memoria RAM utilizada en GB y el porcentaje de uso de la GPU.

Red neuronal	SSD MobileNet V2 FPN Lite 320x320	SSD MobileNet V2 FPN Lite 640x640
CPU	33%	30%
GPU	19%	32%
RAM	2.9GB	4.8GB

Tabla 2.1: Evaluación de precisión promedio de la media y Exhaustividad en redes neuronales entrenadas

Se observó un aumento del uso de recursos computacionales al utilizar una red neuronal que procesa imágenes con mayor resolución, esto se debe a que las matrices de datos que debe alojar en memoria tienen mayor tamaño.

A continuación se muestra el valor de la función de pérdida de cada entrenamiento en relación a la cantidad de iteraciones o steps.

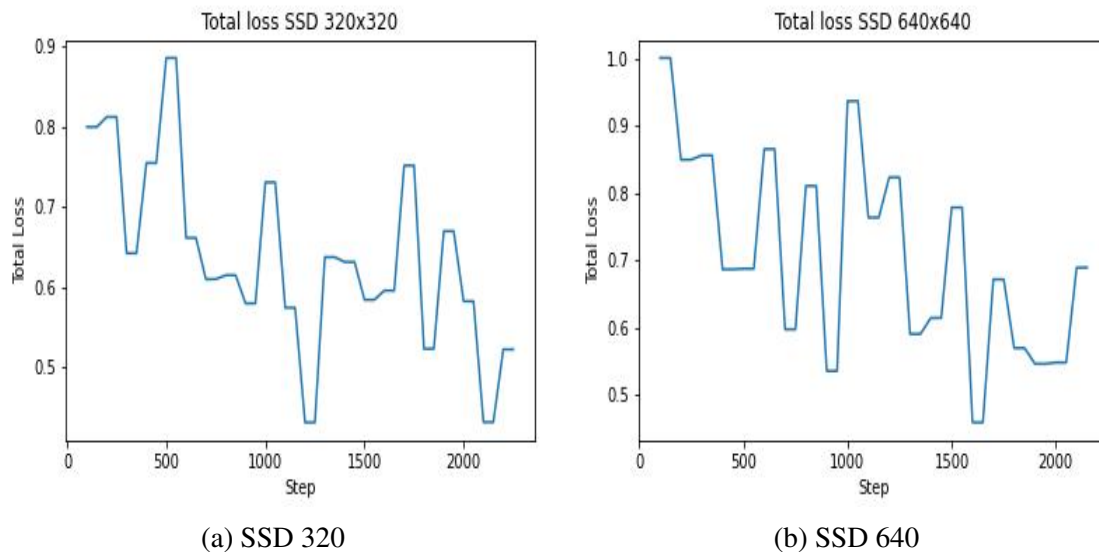


Figura 2.14: Función de pérdida en entrenamiento

En la figura 2.14 como el valor de la función de pérdida decrece durante el entrenamiento, obteniendo un mínimo de 0.43 en la iteración 1250 para la red neuronal con resolución

320x320, y un mínimo de 0.46 en la iteración 1650 para la red neuronal con resolución 640x640. Por otro lado, el entrenamiento de la red neuronal SSD 320 tardó 15 minutos en completar 2250 iteraciones, mientras que la red neuronal SSD 640 demoró 51 minutos en completar 2150 iteraciones, esto da cuenta de que utilizar imágenes de mayor resolución implica un mayor uso de recursos computacionales y de tiempo.

2.2.5 Evaluación

Para la evaluación se utilizó imágenes que no fueron usadas para el entrenamiento, como se mencionó anteriormente, el conjunto de imágenes disponibles se dividió en imágenes de entrenamiento y de validación, para lograr evaluar el desempeño de la red neuronal en imágenes que no haya visto antes.

Por otro lado, se utilizó las métricas de evaluación COCO[23], el cual mide la superposición entre los recuadros con detecciones reales y los recuadros calculados por la red neuronal. La superposición se mide calculando el área de intersección entre un recuadro con una detección real previamente establecido y el recuadro calculado por la red neuronal, y se divide por el área de ambos recuadros, a este cálculo se le denomina intersección sobre unión o IoU por sus siglas en inglés .

Existen cuatro casos posibles para las detecciones, un verdadero positivo (TP) es un rostro detectado correctamente, un falso positivo (FP) es una detección donde no hay un rostro, un verdadero negativo es no detectar donde no hay rostro, y un falso negativo es no detectar donde si hay rostro. Para el sistema de evaluación COCO un verdadero positivo ocurre cuando hay una superposición mayor a 0.5.

La precisión[24] se calcula como la cantidad de verdaderos positivos entre la cantidad total de detecciones 2.3, y el sistema COCO mide la precisión para diferentes grados de superposición, que van desde 0.5 a 0.95 con intervalos de 0.05.

$$Precision = \frac{TP}{TP + FP} \quad (2.3)$$

Otro término importante es el exhaustividad[24], el cual corresponde a la proporción entre verdaderos positivos y la suma de verdaderos positivos mas falsos negativos 2.4. Al igual que la precisión, para el sistema de evaluación COCO se mide la exhaustividad para intervalos de superposición que van desde 0.5 a 0.95.

$$Exhaustividad = \frac{TP}{TP + FN} \quad (2.4)$$

Por último, es importante poder representar los casos donde la detección presente buena precisión y mala exhaustividad y viceversa, para estos casos se utiliza valor F1, el cual utiliza los parámetros mencionados en su cálculo tal como se presenta en la ecuación 2.5.

$$F1Score = \frac{2 \cdot Exhaustividad \cdot Precisión}{Exhaustividad + Precisión} \quad (2.5)$$

Dada la naturaleza del precisión y del exhaustividad, y del cálculo realizado con los grados de superposición IoU, se presenta una relación donde a bajos niveles de IoU, por ejemplo del 25%, se obtiene una mayor cantidad de detecciones de las cuales es probable que la mayoría sean verdaderos positivos, y a la vez una menor cantidad de falsos negativos, provocando que el exhaustividad calculado tenga valores elevados, y también una menor medición en precisión. En cambio al tener una nivel de confianza IoU elevado, como un 80%, se presenta una menor cantidad de detecciones y es menos probable obtener verdaderos positivos, aumentando la cantidad de falsos negativos y también una menor cantidad de falsos positivos, resultando un exhaustividad menor y un precisión mayor. Esta relación se presenta en la curva Precision-Exhaustividad[25] como se muestra en la figura 2.15

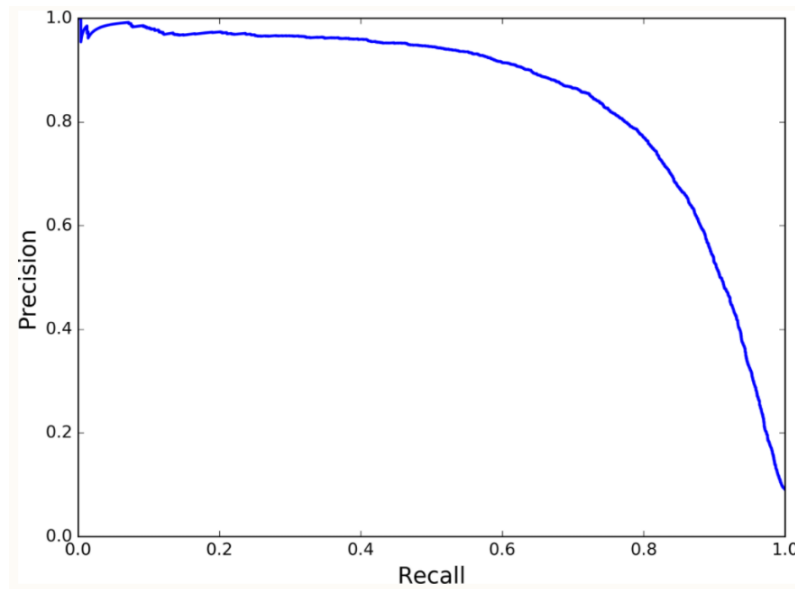


Figura 2.15: Curva Precisión y Exhaustividad de ejemplo

Con el fin de estandarizar la medición y de representar los resultados en los diferentes intervalos de confianza IoU es que se genera una medición llamada Precisión promedio [24] o AP, la cual se define como el área bajo la curva Precisión-Exhaustividad, y dado que los detectores de objetos pueden identificar diferentes clases de objetos se define la Precisión

promedio de la media [24] o mAP, la cual representa la Precisión promedio para todas las clases.

Red neuronal	SSD 320	SSD 640
mAP [IoU=0.50:0.95]	0.066	0.090
Exhaustividad [IoU=0.50:0.95]	0.106	0.148

Tabla 2.2: Evaluación de precisión promedio de la media y Exhaustividad en redes neuronales entrenadas

De acuerdo a los resultados obtenidos en la evaluación presentes en la tabla 2.2, la red neuronal SSD 640 tiene mejores resultados ya que tiene una mayor mAP y mayor exhaustividad, lo que implica que tiene mejor precisión y detecta mejor los verdaderos positivos respecto a la matriz de confusión.

Se realizó pruebas con imágenes de vídeo conferencia y las redes neuronales entrenadas para corroborar los datos de la evaluación, a continuación se muestra un ejemplo de los resultados obtenidos:

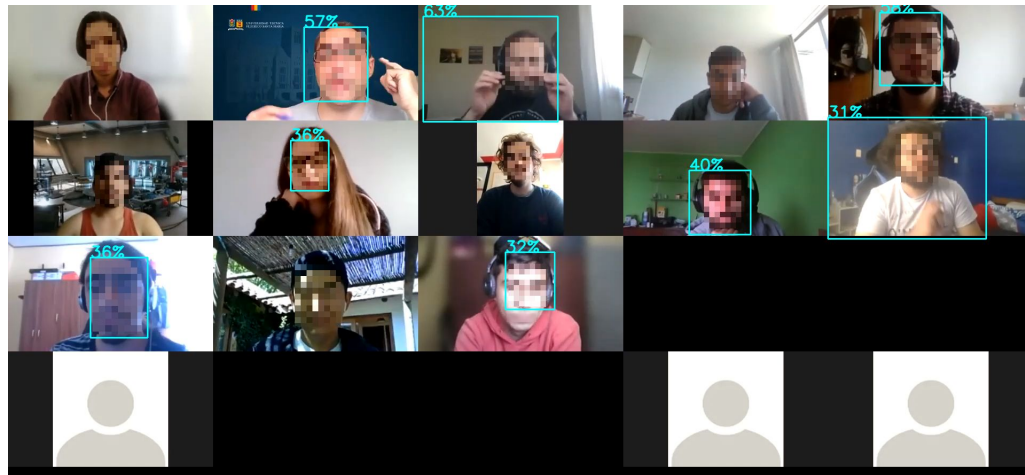


Figura 2.16: Ejemplo con red neuronal SSD MobileNet V2 FPNLite 320x320

En la figura 2.16 donde se utilizó la red neuronal SSD 320 se logró detectar 8 de 13 rostros, con una probabilidad de acierto de verdaderos positivos promedio del 44%, y un tiempo promedio de procesamiento de imagen de 0.25 segundos. Por otro lado, en la figura 2.17 donde se utilizó la red neuronal SSD 640 se detectó 12 de 13 rostros, con una probabilidad de acierto de verdaderos positivos promedio del 74%, y el tiempo de procesamiento de imagen

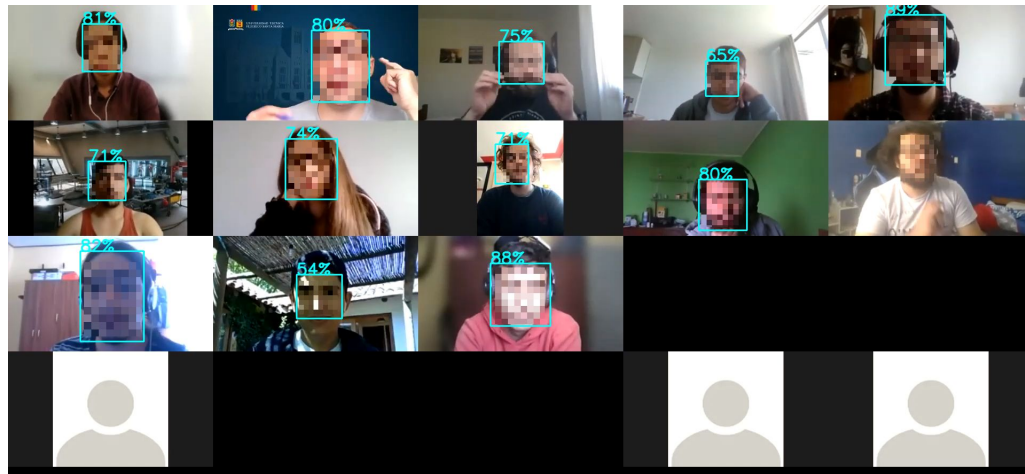


Figura 2.17: Ejemplo con red neuronal SSD MobileNet V2 FPNLite 640x640

promedio es de 0.35 segundos. En comparativa, la red neuronal SSD 640 obtiene una mayor cantidad de verdaderos positivos, con una probabilidad mayor de detección y en promedio tarda 0.1 segundos mas que la otra red, sin embargo se mantiene por debajo de los 0.5 segundos que se definieron en un comienzo para el tiempo mínimo utilizado en generar una expresión facial, estableciendo así a esta red neuronal como la mejor opción a utilizar.

2.3 Seguimiento de rostros

Para el desarrollo de esta memoria se consideró que las personas en la videoconferencia están en movimiento en un rango de posición en la pantalla, esto quiere decir que el rostro de una persona no siempre va a estar en la misma posición en instantes de tiempo diferentes, sin embargo durante la videoconferencia el rostro de cada persona va a estar dentro de un rango de posiciones definido.

En esta sección se desarrolló el seguidor de rostros, si se detecta un rostro en una determinada posición y el rostro sigue siendo detectado en una misma posición o en una posición cercana dentro de un rango definido, entonces las detecciones realizadas corresponden a un mismo rostro, de manera que durante la ejecución del programa se obtiene un vídeo de cada rostro dentro de la videoconferencia.

Para definir la posición de un rostro se estableció cada imagen como un sistema de coordenadas x, y, donde cada píxel de la imagen es un punto dentro de una matriz de datos. La detección del rostro retorna las coordenadas del recuadro que contiene el rostro, definido por el píxel de la esquina superior izquierda (x1,y1) y el píxel de la esquina inferior derecha

(x_2, y_2) . En la figura 2.18 se muestra un ejemplo de la definición posición de un píxel y la definición de un recuadro dentro de una imagen.

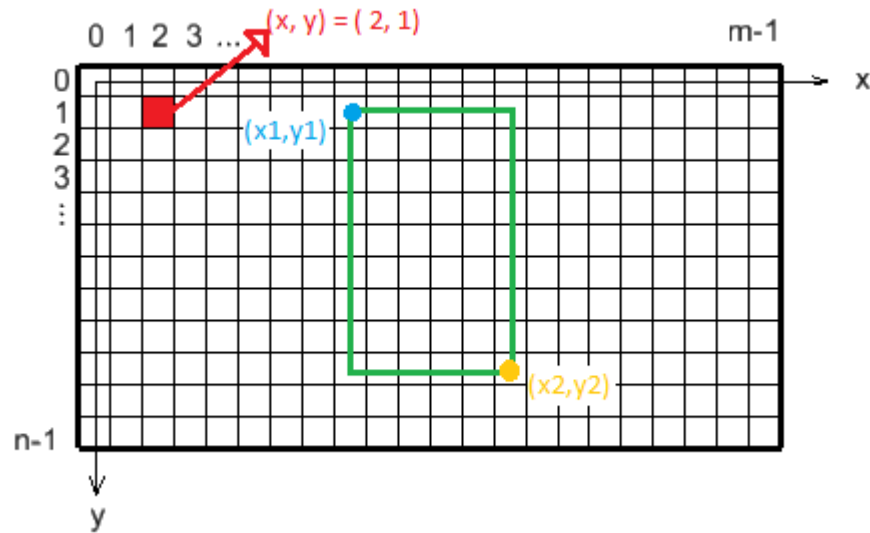


Figura 2.18: Sistema de coordenadas en una imagen

Con el objetivo de saber si dos detecciones corresponden a un mismo rostro se estableció que deben estar en una misma posición o cercanas, para esto se utiliza el término de intersección sobre unión o IoU por sus siglas en inglés. Este término determina que tan superpuestos están dos recuadros en un mismo marco de referencia, para esto se calcula el área de la intersección y se divide por el área total de la superficie compuesta por ambos recuadros, obteniendo el porcentaje de superposición.

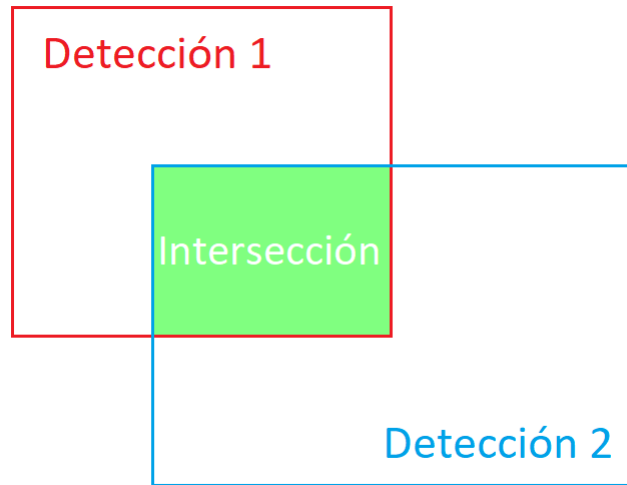


Figura 2.19: Intersección de recuadros

En la figura 2.19 se presenta la intersección de dos cuadrados, y la superposición se calcula como:

$$superposicion = \frac{A_{Interseccion}}{A_{Deteccion1} + A_{Deteccion2} - A_{Interseccion}} \quad (2.6)$$

Esta función se programó en Python y se agregó una condición de tal forma que si un recuadro está dentro de otro recuadro la superposición sea del 100%.

Finalmente se programó una función que itera sobre los rostros detectados y por sobre cada imagen capturada durante la ejecución del programa. El procedimiento de esta función se divide en dos etapas, primero una etapa de ajuste, donde se calcula el área donde un rostro puede estar presente, para cada rostro detectado se toman las primeras 40 imágenes capturadas desde que se realizó la primera detección del rostro, en cada iteración se genera un recuadro donde se encuentra el rostro, considerando que los recuadros deben tener un mínimo de superposición del 20% para determinar que corresponden al mismo rostro, y se une cada recuadro para obtener un área y posición representativa de las posibles posiciones del rostro, estableciendo los recuadros iniciales para cada rostro capturado.

En la figura 2.20 se observa que en cada captura realizada se obtiene la detección del rostro en un marco verde y se genera un recuadro en rojo que abarca las posibles posiciones del rostro dado el algoritmo descrito anteriormente, este recuadro rojo corresponde al recuadro inicial del rostro y es utilizado como base para comparar las detecciones en las siguientes capturas, además el cuadro inicial se utilizó para definir el tamaño del vídeo que contendrá

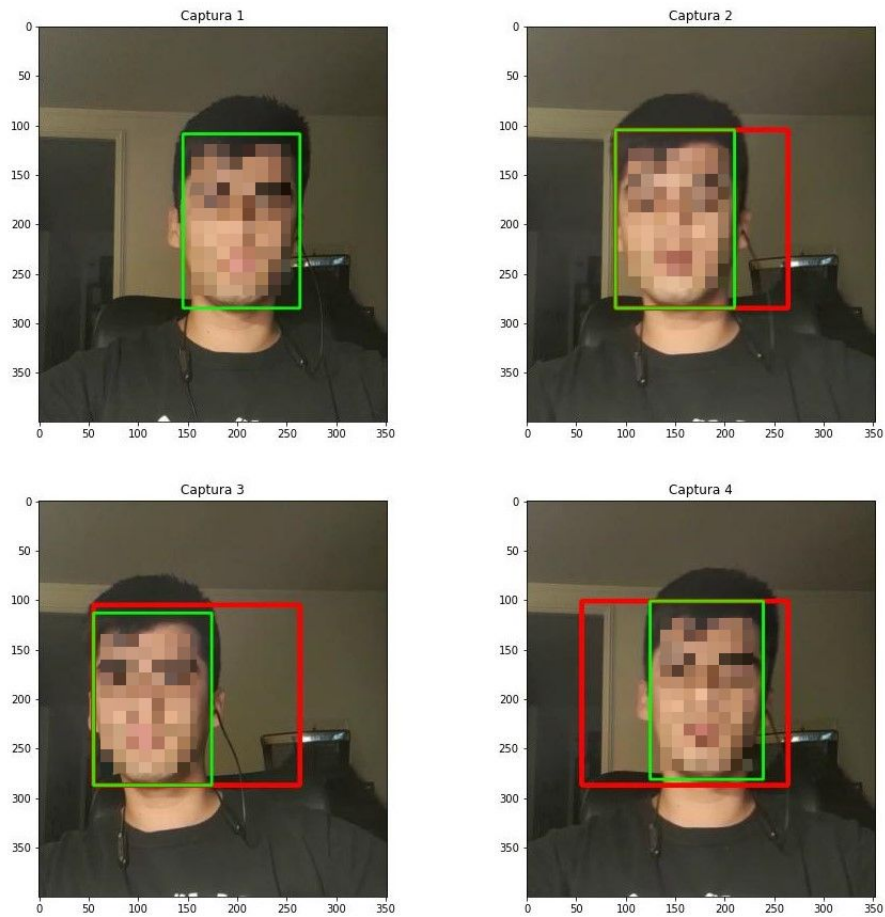


Figura 2.20: Ajuste de recuadros iniciales para seguidor

las capturas realizadas del rostro.

En la segunda etapa se compara los recuadros iniciales con las detecciones realizadas en las nuevas imágenes capturadas, si el algoritmo determina que la posición del rostro detectado coincide con un cuadro inicial entonces se asocian al mismo rostro y se almacena la imagen del rostro capturado, esto se realiza para cada rostro detectado en cada imagen capturada. Cada vez que se detecte un rostro que no se encuentre dentro de los recuadros iniciales se se ejecutará la primera etapa de ajuste y luego la segunda etapa, por otro lado, si un rostro deja de ser detectado después de 40 imágenes capturadas entonces el programa finalizará la captura de datos de ese rostro.

2.4 Interfaz de usuario

En esta sección se desarrolló la interfaz que utilizará el usuario para la ejecución del programa, esta interfaz permite iniciar y detener la captura de imágenes y detección de rostros, como también definir el límite porcentual de confianza mínimo que debe tener una detección para ser considerada un rostro y muestra la imagen capturada con los rostros detectados.

La interfaz se diseñó con la librería *tkinter*, la cual está incluida en el paquete inicial de librerías de Python, y en el desarrollo se agregaron los siguientes elementos:

- Botón Iniciar: Comienza la captura de imágenes desde la pantalla definida como monitor principal e inicia la detección de rostros.
- Botón Detener: Cesa la captura de imágenes y la detección de rostros.
- Spin box: Es una caja de texto numérica que permite ingresar el valor del porcentaje mínimo de probabilidad para detección.
- Botón Set: Define el porcentaje mínimo de probabilidad para que una detección sea calificada como rostro.
- Botón Salir: Detiene el programa y cierra la ventana.
- Recuadro imagen: Sección donde se muestra la imagen capturada y los resultados del algoritmo.

El desarrollo de la interfaz culminó en la integración de todos los pasos realizados anteriormente, se agregó la captura de pantalla, la detección de rostros mediante la red neuronal entrenada, se utilizó el algoritmo seguidor de rostros y se habilitó un espacio para que el usuario interactúe con el programa. Además, durante la ejecución del programa se almacenan los vídeos con cada rostro detectado y un vídeo con la captura de imagen completa con las detecciones realizadas.

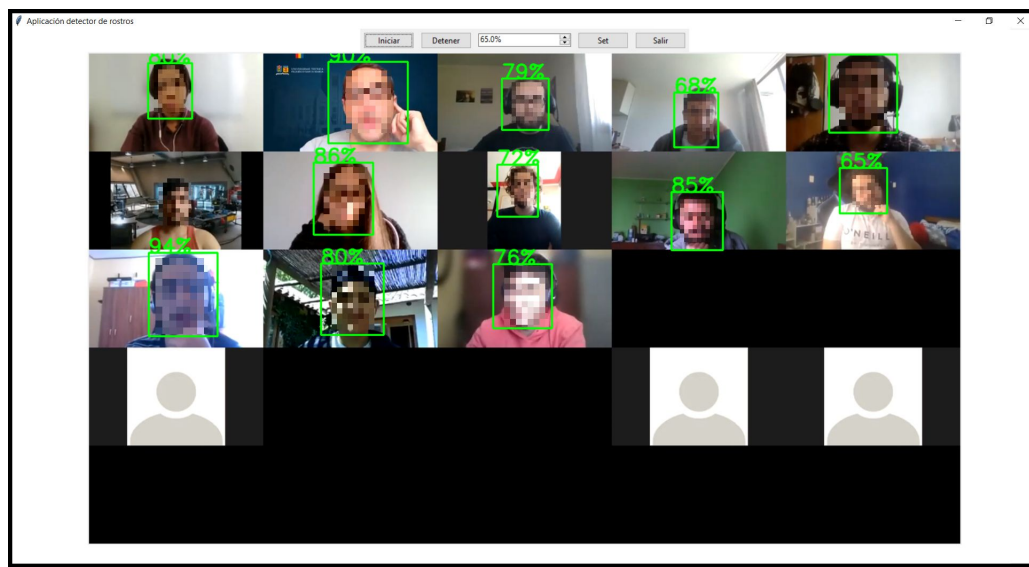


Figura 2.21: Interfaz de usuario

En la figura 2.21 se observa la interfaz desarrollada, con sus botones y la imagen con los resultados de las detecciones. Para este ejemplo se utilizó la red neuronal SSD 640 ya que obtiene los mejores resultados durante la detección.

Capítulo 3

Evaluación del sistema

Para la evaluación del sistema se consideró dos aspectos importantes, primero el desempeño y precisión en la detección de rostros, es decir que tan bien el programa es capaz de detectar rostros, lo cual se relaciona estrechamente con la red neuronal utilizada para la detección. El segundo punto corresponde a la cantidad de recursos computacionales utilizado durante la ejecución del algoritmo, objetivo principal de esta memoria, y a la vez se calculará el tiempo de ejecución de cada iteración, o sea el tiempo que transcurre entre cada captura y procesamiento, dato que definirá si este programa es útil al momento de almacenar los datos con la rapidez adecuada dada las características del equipo donde se ejecute.

Por otro lado, durante el desarrollo de esta memoria se entrenaron principalmente 2 redes neuronales, denominadas SSD MobileNet V2 FPNLite 320x320 (SSD 320) y SSD MobileNet V2 FPNLite 640x640 (SSD 640), las cuales se encuentran en el repositorio TensorFlow 2 Detection Model Zoo [22], por lo tanto se realizarán pruebas con ambas redes entrenadas para representar las diferencias en precisión en detecciones y uso de recursos computacionales, como de tiempo por cada iteración.

Así se definió hacer 4 pruebas principales, donde se evaluarán los puntos mencionados anteriormente:

- **Prueba 1:** Red SSD MobileNet V2 FPNLite 320x320 sólo con CPU.
- **Prueba 2:** Red SSD MobileNet V2 FPNLite 320x320 con CPU y GPU.
- **Prueba 3:** Red SSD MobileNet V2 FPNLite 640x640 sólo con CPU.
- **Prueba 4:** Red SSD MobileNet V2 FPNLite 640x640 con CPU y GPU.

En cada una de estas pruebas se busca cuantificar el desempeño en la detección de rostros y el uso de recursos computacionales, para lograr este objetivo se realizaron iteraciones con imágenes que contengan desde 1 rostro hasta un máximo de 150 rostros, se definió este número ya que en las plataformas de Zoom y Google Meets se tiene un máximo de 100 participantes en las reuniones, por otro lado se quiere llevar las pruebas a un punto donde ya no se detecte ningún rostro. Con el fin de que las pruebas realizadas sean equivalentes para cada uno de los experimentos realizados se utilizó 100 muestras por cada iteración en la cantidad de rostros, es decir que se comenzó con 100 imágenes con 1 rostro hasta 100 imágenes con 150 rostros.

Dado que no es fácil encontrar capturas de videoconferencia con la cantidad de rostros requeridos para cada prueba se optó por crear dichas imágenes, de esta manera se puede controlar la cantidad de rostros en cada caso y determinar la posición de estos para luego corroborar que tan bien el programa es capaz de detectar rostros. Para crear este set de datos se requiere de imágenes que contengan rostros aislados y saber la posición del rostro dentro de cada captura, dado que el set de datos WiderFace[3] utilizado en el entrenamiento contiene varias imágenes con rostros no aislados se decidió buscar otro que satisfaga los requerimientos, y se concluyó que el set de datos Labeled Faces in the Wild[26] es una opción viable, el cual contiene más de 13 mil imágenes con rostros aislados de personas, pero no la posición dentro de la imagen. Cabe recordar que para la evaluación del sistema se utilizó imágenes con rostros sin censura, pero que en el desarrollo de la escritura de la memoria se distorsionó los rostros.

Las imágenes del set de datos seleccionado son de 250 píxeles de ancho por 250 píxeles de alto, manteniendo una relación 1:1, mientras que la mayoría de los monitores, y el monitor utilizado en este experimento tiene una resolución de 1920 por 1080 píxeles, con una relación de 16:9, y por lo tanto las interfaces utilizadas mantienen una relación similar cuando ocupan toda la pantalla. Por lo tanto, para lograr que las imágenes del set de datos se distribuyan de la mejor forma en la simulación de la interfaz se adecuó el tamaño a una relación 16:9, sin embargo, si solo se recorta la imagen en la mayoría de los casos se recorta el rostro, por lo tanto se optó por recortar la imagen y añadir unas barras negras a los lados con el fin de rellenar y lograr el formato deseado tal como se muestra en la figura 3.1.

Por otro lado, para los fines de esta evaluación se requiere saber la posición de los rostros en las capturas, datos que no son entregados en el set de datos y se desarrolló como solución un generador de coordenadas de posición, este programa muestra al usuario las imágenes del set de datos con la resolución obtenida anteriormente, posteriormente el usuario utilizando el cursor del computador puede hacer click y arrastrar el cursor del computador en la imagen

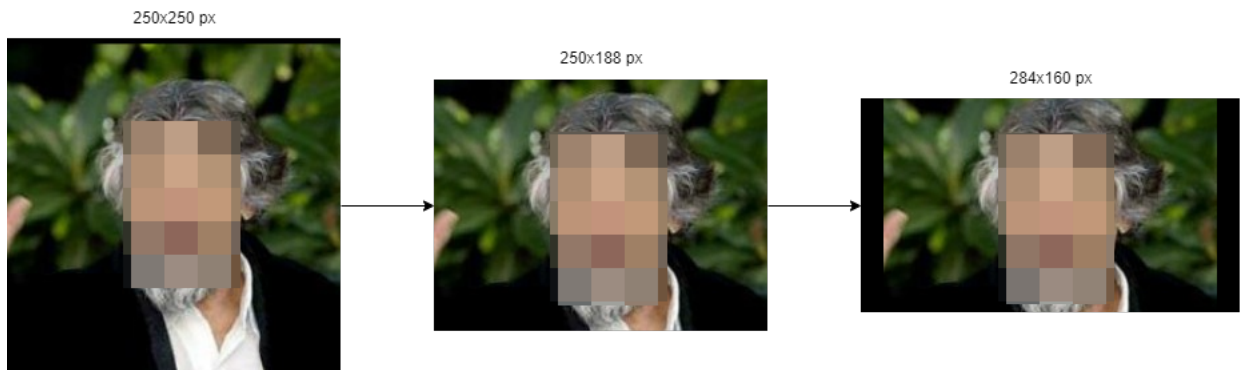


Figura 3.1: Cambio de tamaño en imágenes con rostro para el set de datos de evaluación

desplegada para enmarcar el rostro como se muestra en la figura 3.2.

Se almacenó los datos de posición de rostros en 400 imágenes, donde se guardó la imagen y un archivo de texto con la posición.

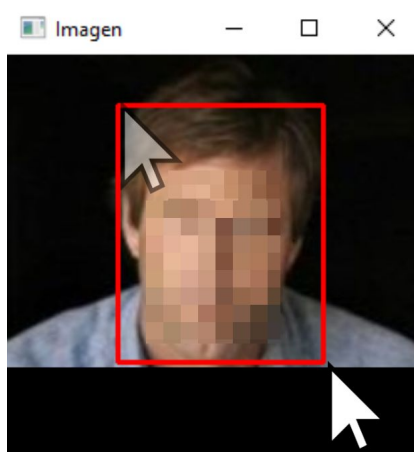


Figura 3.2: Selección de rostros para creación set de datos

Posteriormente, se creó un programa que utilizó todas las imágenes para generar el set de datos deseado, donde seleccionó fotos de forma aleatoria y se agregó condiciones para que los rostros no se repitan dentro de una misma imagen, formando 100 imágenes para los casos de 1 a 150 rostros como se muestra en la figura 3.3, a la vez que se generó archivos de texto con las posiciones de los rostros dentro de la imagen. De esta manera se obtienen capturas iguales para todos los experimentos y se obtienen los datos necesarios para calcular la precisión del algoritmo.

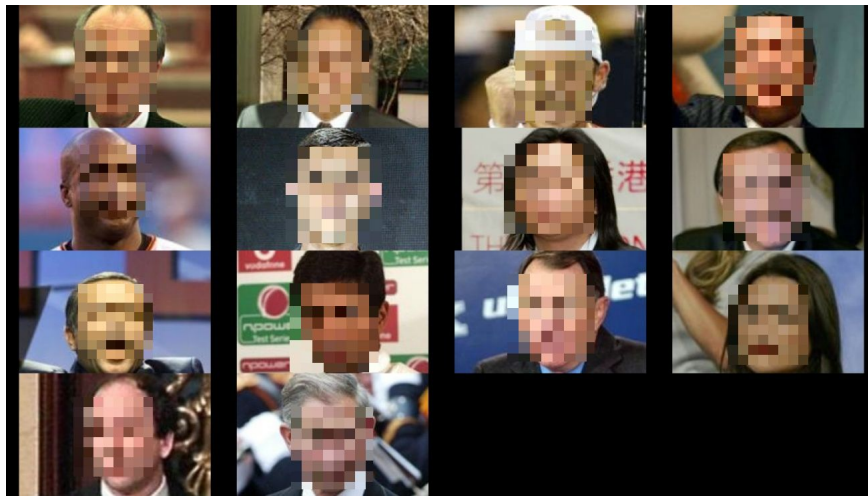


Figura 3.3: Imagen de ejemplo del set de datos creado

Se modificó el algoritmo creado para que el ingreso de las imágenes a procesar no sea realizado por el capturador de pantalla, si no que sea desde los datos almacenados. Además se agregó una función que compara los datos de rostros almacenados y los rostros detectados, se definió que los rostros almacenados y los detectados deben estar un 50% superpuestos para ser considerados un verdadero positivo, también se calculó los falsos positivos y los falsos negativos, y se definió para todas las pruebas que los rostros detectados por la red deben tener al menos un 80% de probabilidad de ser rostros. El seguidor de rostros del algoritmo no fue modificado, dado que la posición de los rostros en el set de datos no varía significativamente y cada rostro se encuentra acotado a un recuadro específico en la imagen es posible determinar si el detector está siguiendo correctamente los rostros basándose en la cantidad de rostros detectados correctamente, es decir que mientras el detector siga encontrando rostros en las posiciones iniciales entonces el seguimiento de rostros será correcto.

Al algoritmo creado para la evaluación se agregó la librería *py3nvm* para medir la utilización de GPU y la librería *psutil* para obtener la utilización de CPU y memoria RAM, además se midió el tiempo utilizado en cada iteración sin considerar el tiempo utilizado por la función que compara los rostros detectados con los rostros reales.

Los datos obtenidos tanto en los resultados de precisión de algoritmo, como de uso de recursos computacionales fueron almacenados en archivos csv para su posterior análisis. Cabe destacar, de los recursos computacionales que son ocupados durante la evaluación se midió solamente los que son utilizados por el programa que procesa las imágenes, así se evitó tener errores al tener otros procesos ejecutándose en paralelo.

Capítulo 4

Resultados

Este capítulo de resultados se divide en 2 secciones principales, la primera se orienta a presentar los resultados de desempeño en reconocimiento de rostros, es decir que tan bien el algoritmo está identificando los rostros dadas las condiciones de evaluación, mientras que la segunda sección presenta el uso de recursos computacionales durante la ejecución del algoritmo dada las pruebas efectuadas en la evaluación. Dentro de cada sección se presenta los resultados de las diferentes pruebas planteadas, separando en la red neuronal utilizada y los casos donde se utilizó GPU o solo CPU.

4.1 Desempeño del algoritmo

El desempeño se evaluó en 3 parámetros, precisión, exhaustividad y valor F1, en los gráficos se muestra la evolución de estos parámetros en relación a la cantidad de rostros mostrados en pantalla, cabe recordar que se realizó pruebas desde 1 rostro hasta 150 y se utilizaron 100 muestras para cada caso.

Dadas las ecuaciones vistas en la sección 2.2.5, estos valores mientras mas cercanos sean a uno indican un mejor desempeño, el valor precisión relaciona la cantidad de detecciones correctas en relación a la cantidad de detecciones totales, exhaustividad representa la relación entre la cantidad de detecciones correctas y la cantidad de rostros que no fueron detectados, y el valor F1 relaciona los dos parámetros mencionados, es la conjunción de estos resultados y nos dan una mejor perspectiva del comportamiento del algoritmo detector.

4.1.1 SSD MobileNet V2 FPNLite 320x320 CPU

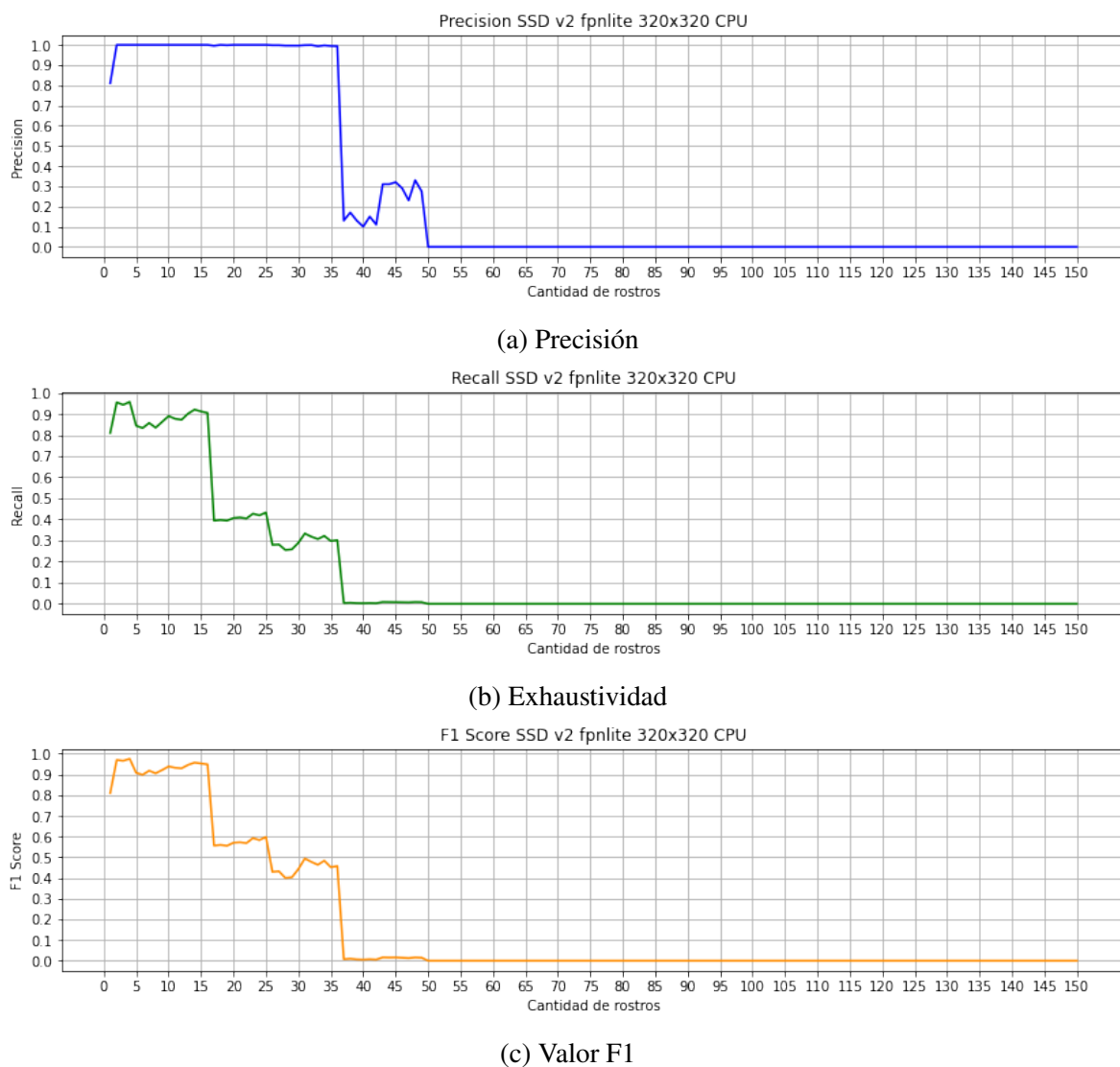


Figura 4.1: Desempeño SSD MobileNet V2 FPNLite 320x320 CPU

En las figuras 4.1 se presentan los gráficos de precisión, exhaustividad y valor F1, en este caso la red neuronal tuvo buena precisión hasta los 35 rostros, también se observó que hasta los 15 rostros la mayoría de ellos eran detectados correctamente, y al analizar el valor F1 se obtiene el desempeño de los dos parámetros mencionados anteriormente, concluyendo que el mejor desempeño se realizó hasta antes de los 15 rostros.

4.1.2 SSD MobileNet V2 FPNLite 320x320 GPU

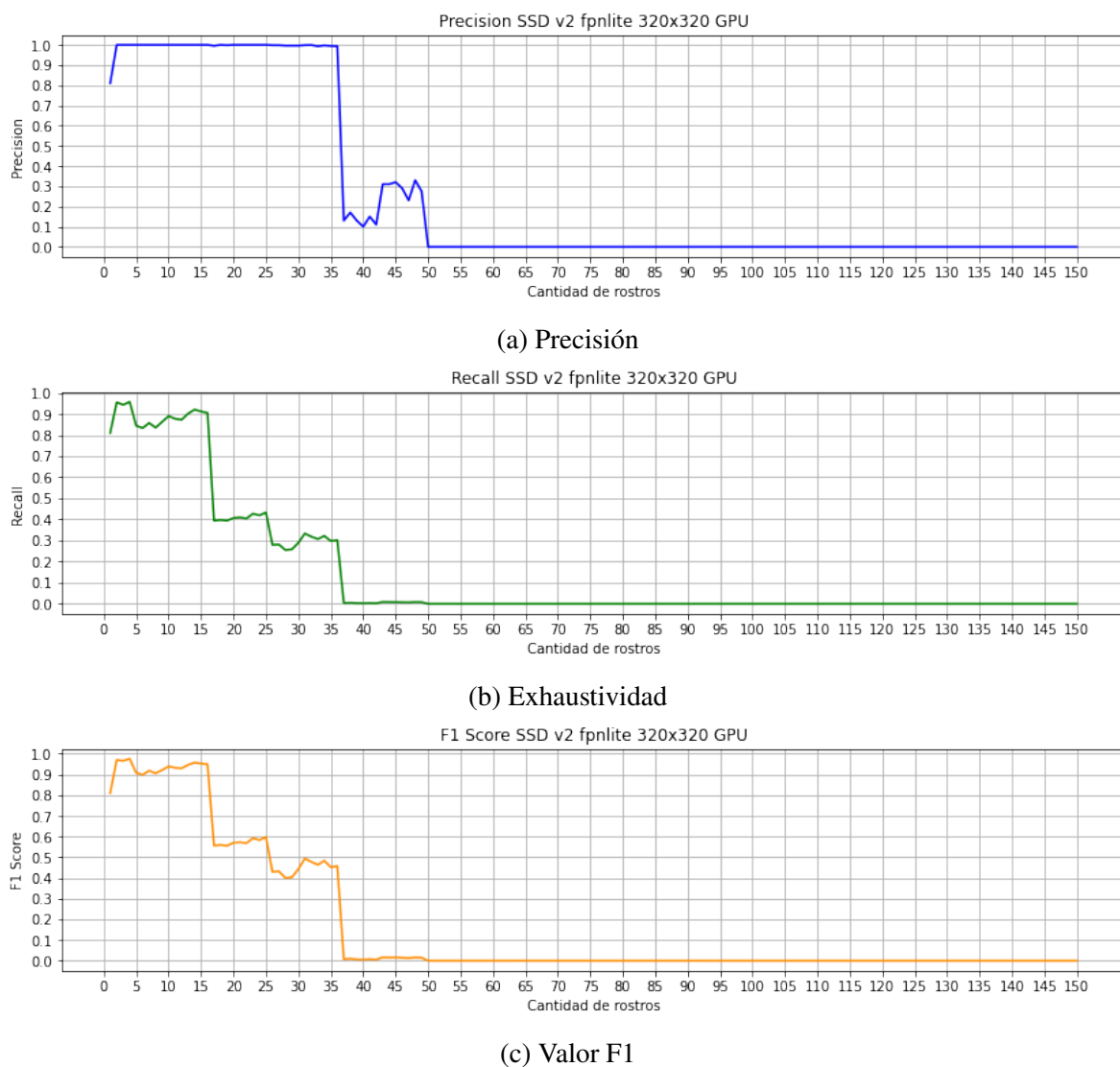


Figura 4.2: Desempeño SSD MobileNet V2 FPNLite 320x320 GPU

El caso presentado en la figura 4.2 es idéntico al anterior en la figura 4.1, esto se debe a que se trata de la misma red neuronal, los cálculos realizados al analizar una imagen son los mismos que en el caso anterior y por lo tanto se obtienen los mismos resultados, el único cambio presente en esta prueba es que los cálculos principales realizados por la red neuronal se ejecutan en la GPU, mientras que en el caso anterior los cálculos se hacen exclusivamente en la CPU.

4.1.3 SSD MobileNet V2 FPNLite 640x640 CPU

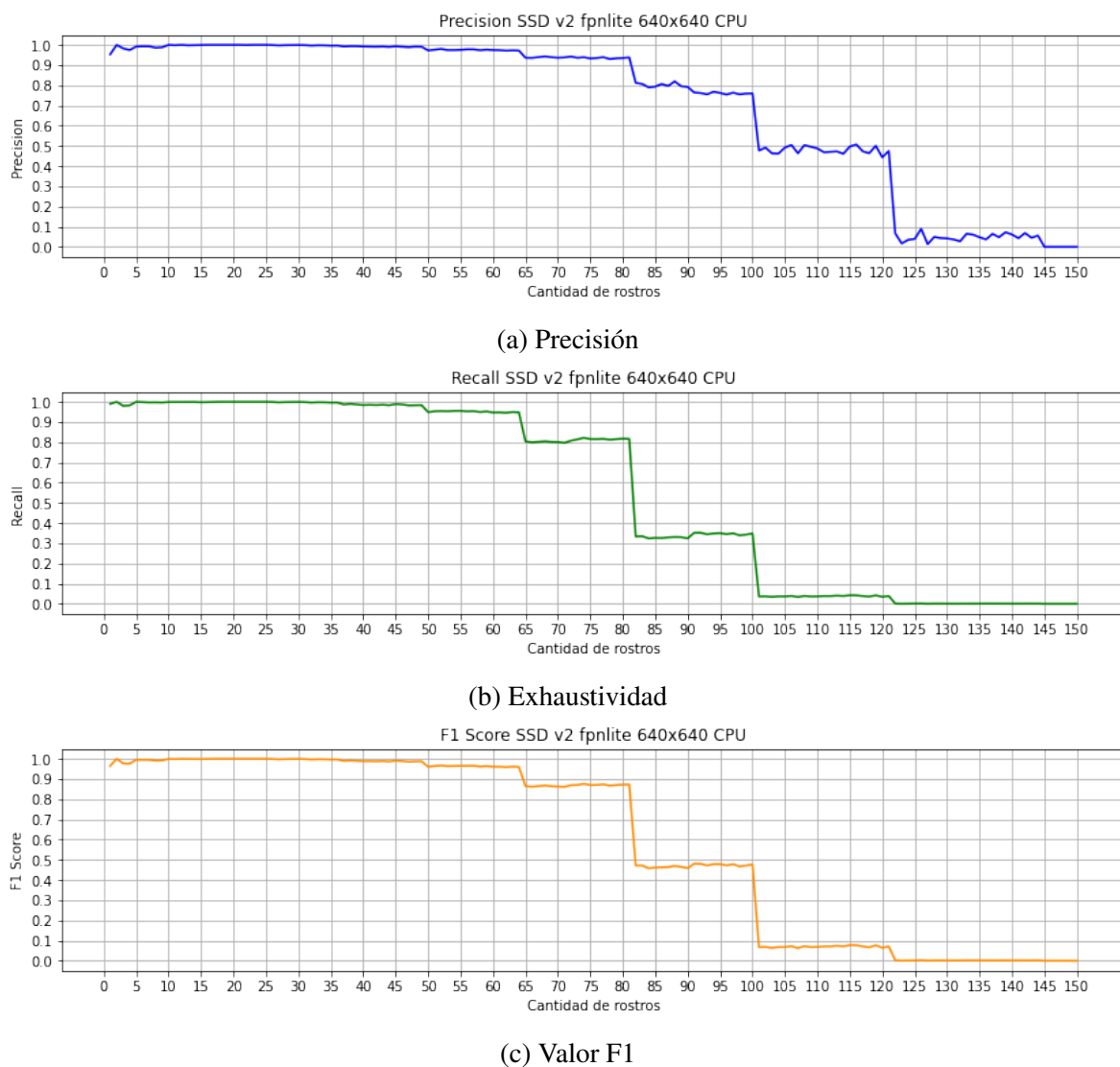
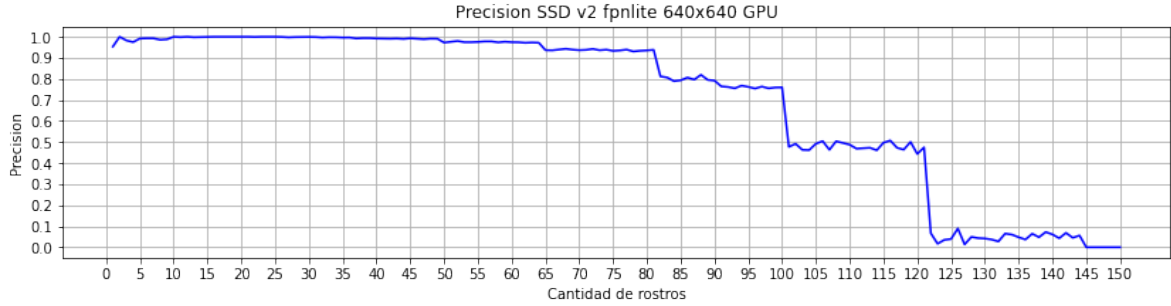


Figura 4.3: Desempeño SSD MobileNet V2 FPNLite 640x640 CPU

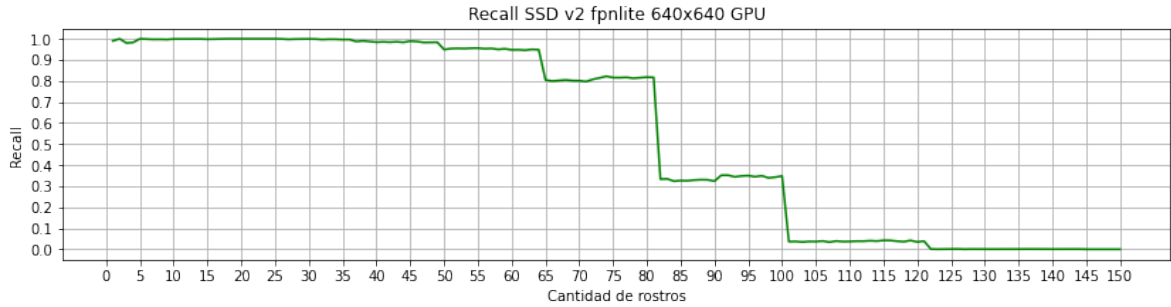
En los resultados de la prueba presentada en la figura 4.3 se muestra una diferencia respecto a los casos anteriores, esto se debe a que es otra red neuronal y por lo tanto los cálculos realizados son diferentes. En este caso el valor precisión se mantiene en un rango 0.9-1.0 hasta los 80 rostros presentes en la imagen, mientras que el valor exhaustividad se mantiene en un rango 0.9-1.0 hasta casi los 65 rostros, dando como resultado un valor F1 que se mantienen en un rango 0.9-1.0 hasta casi los 65 rostros, resultados que indican que la red neuronal SSD 640 detecta hasta 4 veces más rostros que la SSD 320 donde el valor F1 se

mantuvo en el rango 0.9-1.0 hasta los 15 rostros.

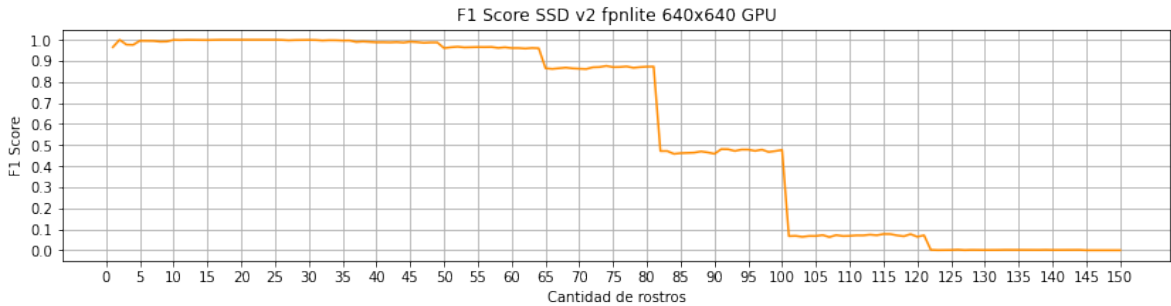
4.1.4 SSD MobileNet V2 FPNLite 640x640 GPU



(a) Precisión



(b) Exhaustividad



(c) Valor F1

Figura 4.4: Desempeño SSD MobileNet V2 FPNLite 640x640 GPU

Al igual que en el caso donde se utilizó la red neuronal SSD 320 en CPU y GPU, en la prueba en GPU de la red SSD 640 que se presenta en la figura 4.4 no se encuentra diferencia respecto a la prueba de la figura 4.3 dado que es la misma red y los cálculos son los mismo, la diferencia es que unos se realizan principalmente en la GPU y en el otro caso se realizan exclusivamente en la CPU.

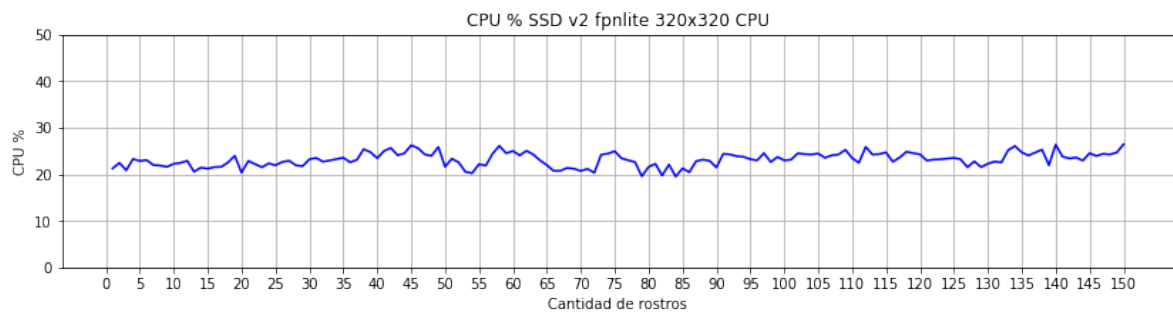
De los resultados de esta primera parte de las pruebas realizadas se destaca el hecho de que los cálculos realizados en una misma red, pero utilizando diferentes dispositivos de cómputo, no presentan una mayor diferencia en los resultados obtenidos, en esta prueba no se comprobó si los resultados eran exactamente iguales respecto a si los recuadros obtenidos en el detector de rostros, sin embargo se calculó diferencia al utilizar CPU y GPU de los parámetros precisión, exhaustividad y valor F1, y se obtuvo una diferencia de 0, es decir que esos resultados son exactamente iguales en todas las imágenes utilizadas para la prueba. Por lo tanto esta prueba da cuenta de que el cambio en el hardware no modificó los resultados de desempeño en reconocimiento de rostros en la pruebas realizadas al utilizar la misma red neuronal.

4.2 Uso de recursos computacionales

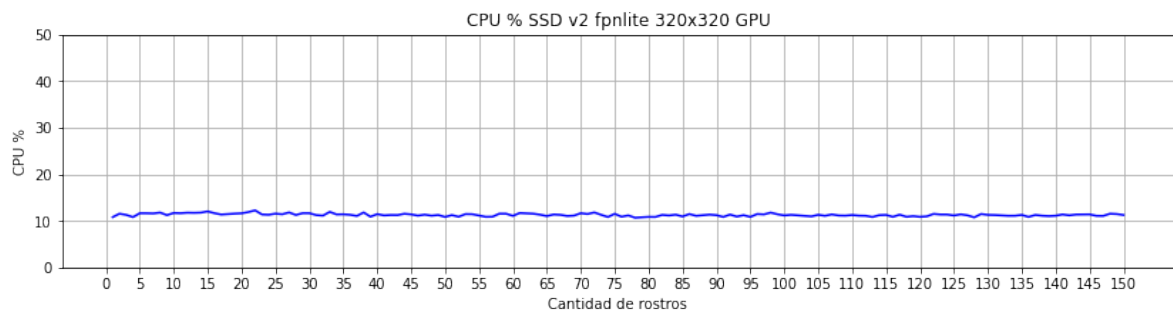
En esta segunda parte de los resultados se presenta las diferencias realizar los cálculos del detector de rostros utilizando GPU y utilizando exclusivamente CPU, donde los factores a comparar son el porcentaje de utilización de la CPU, el porcentaje de utilización de la GPU, la cantidad de memoria RAM en GB y el tiempo promedio por iteración. Con el fin de entregar una mejor comparación se presentaran los resultados de cada factor por separado para así comparar los casos donde se utilizó CPU, GPU y las redes neuronales SSD 320 y SSD 640.

4.2.1 Utilización de CPU

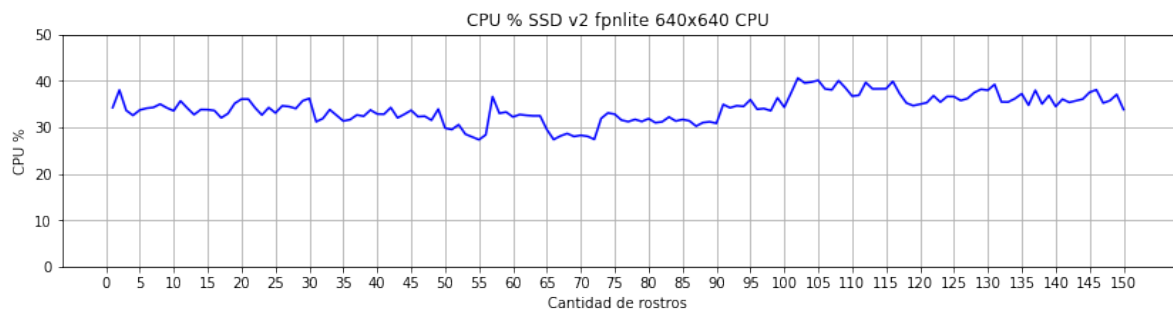
En la figura 4.5 se compara la utilización de CPU al utilizar las redes neuronales SSD 320 y SSD 640 con CPU y GPU, en la figura 4.5a donde se implementa la red SSD 320 con CPU se observa una utilización en un rango entre el 20%-30%, mientras que en la red neuronal SSD 640 con CPU observa un aumento dando un rango entre 30% y 40%. Por otro lado, al utilizar GPU en el computo se observa que tanto en la red SSD 320 como en la SSD 640 la CPU se mantiene aproximadamente en el 10% de utilización durante las iteraciones de las pruebas. Finalmente, un aspecto importante a mencionar es que la utilización de CPU no se vio afectada significativamente por la cantidad de rostros detectados, los rangos de utilización se mantuvieron relativamente similares para todos los casos al reconocer entre 1 y 150 rostros.



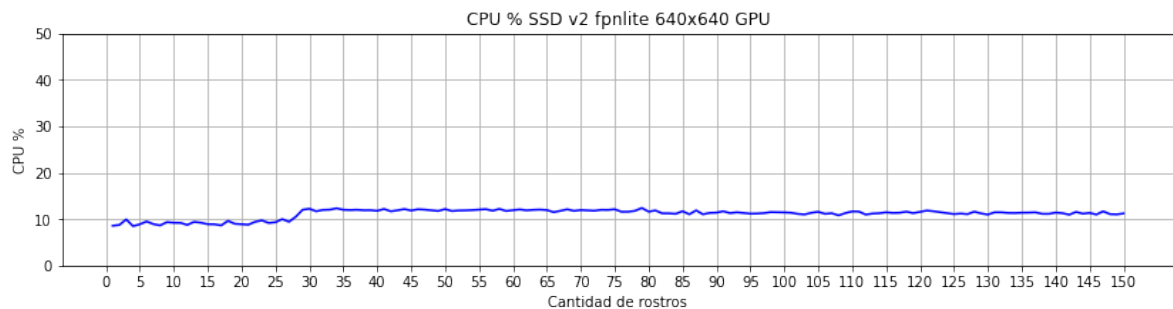
(a) SSD MobileNet V2 FPNLite 320x320 con CPU



(b) SSD MobileNet V2 FPNLite 320x320 con GPU

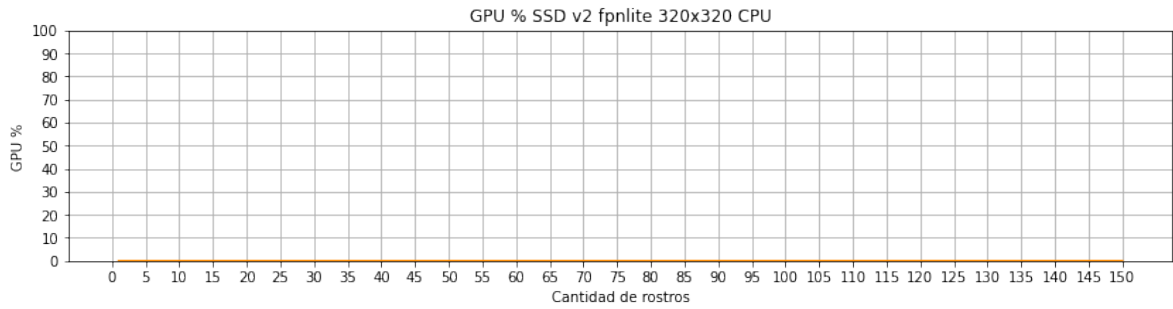


(c) SSD MobileNet V2 FPNLite 640x640 con CPU

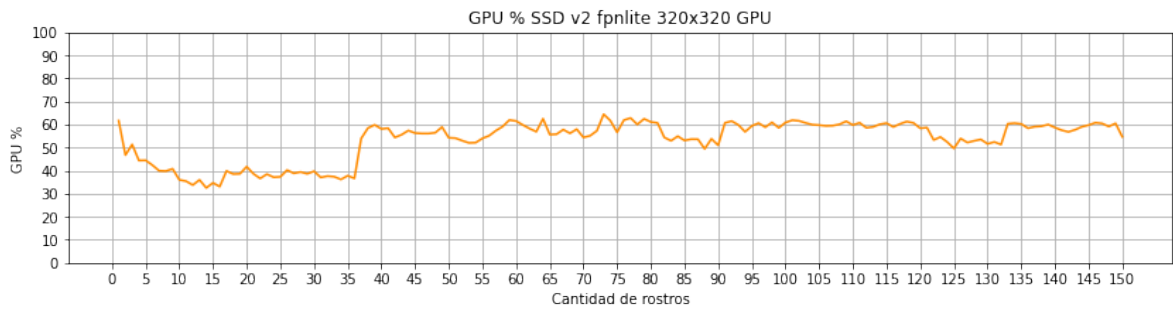


(d) SSD MobileNet V2 FPNLite 640x640 con GPU

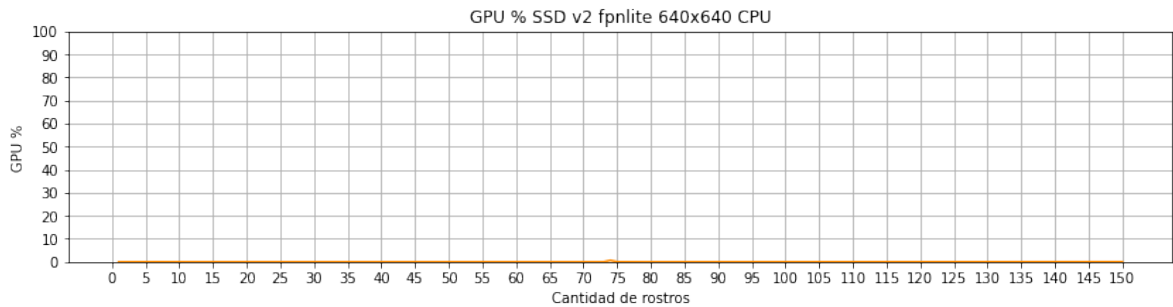
Figura 4.5: Utilización de CPU



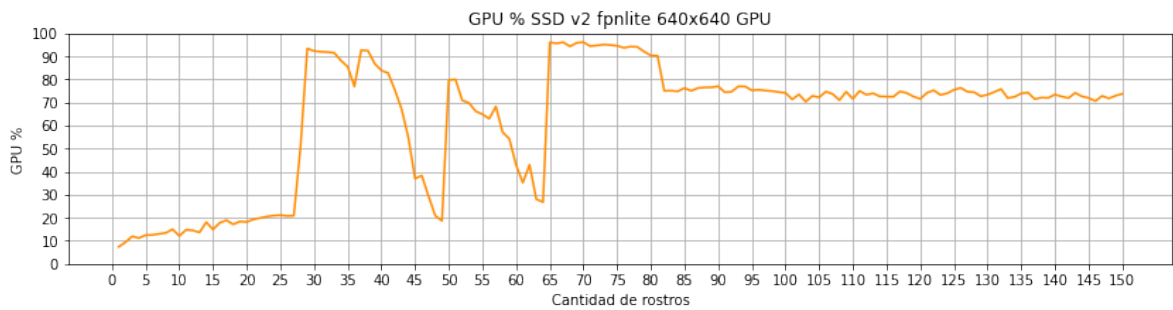
(a) SSD MobileNet V2 FPNLite 320x320 con CPU



(b) SSD MobileNet V2 FPNLite 320x320 con GPU



(c) SSD MobileNet V2 FPNLite 640x640 con CPU



(d) SSD MobileNet V2 FPNLite 640x640 con GPU

Figura 4.6: Utilización de GPU

4.2.2 Utilización de GPU

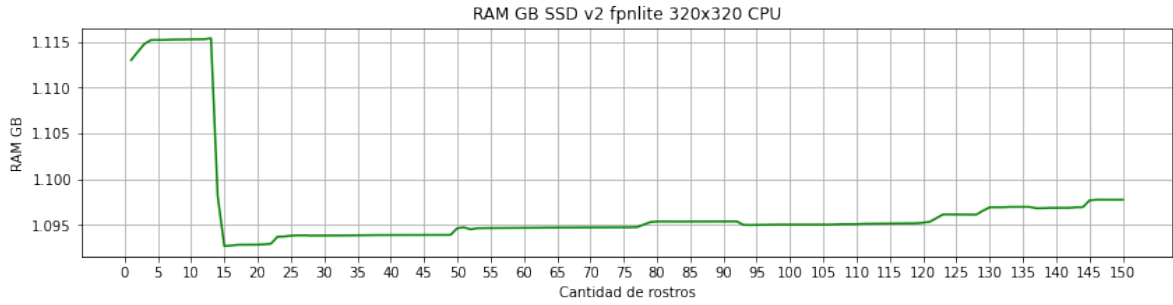
En la figura 4.6 se presenta los resultados de utilización de GPU en las pruebas realizadas, se observa en particular en las figuras 4.6a y 4.6c una utilización del 0% de GPU dado que el algoritmo se configuró para no utilizar este dispositivo durante la prueba y este factor corrobora que las configuraciones fueron realizadas correctamente.

En la figura 4.6b se observan diferencias en utilización a medida que se muestra una menor o una mayor cantidad de rostros en las imágenes, comienza con una utilización al 60% con 1 rostros, luego desciende hasta un 35% aproximado de utilización en los 15 rostros presentados que es hasta este límite de rostros donde el valor F1 se mantuvo en un rango 0.9-1.0 para la red SSD 320, posteriormente la red se mantienen en un rango de utilización aproximado del 40% entre los 15 y 35 rostros, y finalmente se mantiene en un rango entre el 50% y 60% de utilización desde los 40 a los 150 rostros presentes en las imágenes. Cabe destacar que en la figura 4.1 y 4.2 donde se muestra el desempeño en reconocer los rostros de la red neuronal SSD 320 se obtuvo un valor F1 distinto de 0 hasta los 35 rostros detectados, esto calza con el aumento en utilización de la red neuronal hasta el 60%.

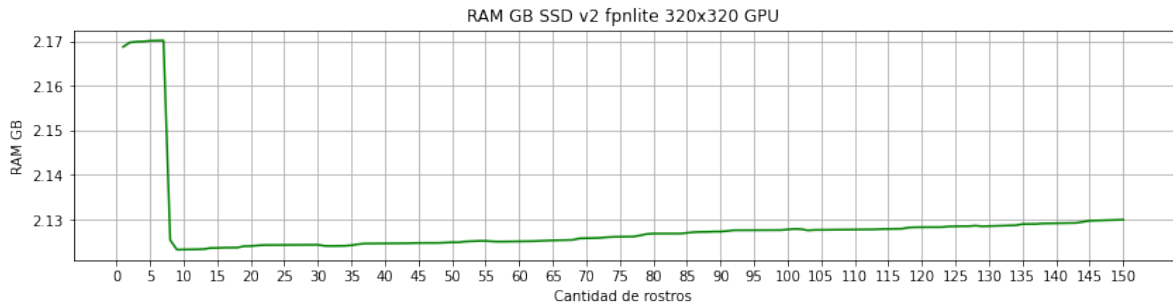
Finalmente, en la figura 4.6d se observan variaciones en el uso de GPU al implementar la red SSD 640, específicamente la utilización alcanza valores sobre el 90% entre los 25 y los 40 rostros, y este umbral también se alcanza entre los 65 a 80 rostros presentes en una imagen. El mínimo de utilización se alcanza al comienzo del gráfico, es decir cuando se encuentra solo 1 rostro en las imágenes de prueba donde se presenta aproximadamente un 10% de utilización. Por otro lado, desde los 80 rostros en adelante la GPU se estabiliza en aproximadamente un 75% de utilización, este corte calza en donde el valor F1 baja de los 0.9 tal como se presenta en la figura 4.4c

4.2.3 Utilización de memoria RAM

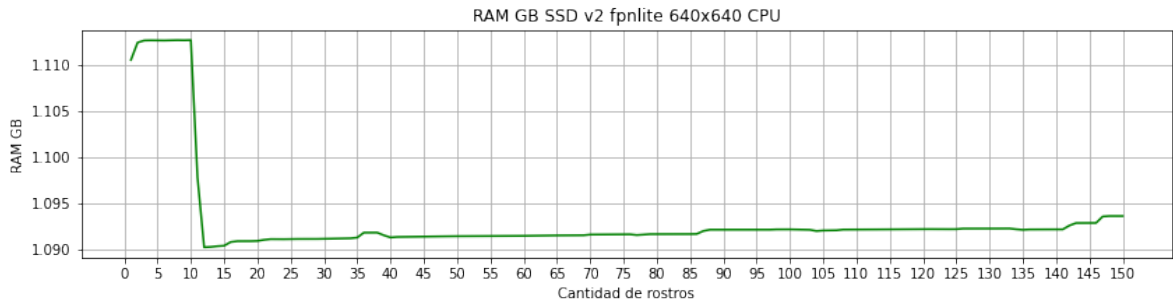
En los resultados presentados en la figura 4.7 se muestra la cantidad de memoria RAM utilizada en GB para los diferentes casos, donde a rasgos generales se presenta un máximo en el uso de memoria al comienzo de las iteraciones, luego la cantidad de memoria disminuye considerablemente, y a partir de ese punto el uso continua creciente aproximadamente lineal.



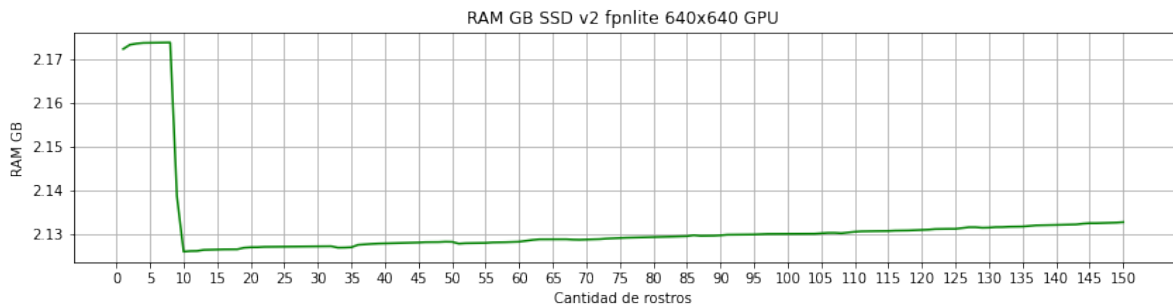
(a) SSD MobileNet V2 FPNLite 320x320 con CPU



(b) SSD MobileNet V2 FPNLite 320x320 con GPU



(c) SSD MobileNet V2 FPNLite 640x640 con CPU



(d) SSD MobileNet V2 FPNLite 640x640 con GPU

Figura 4.7: Utilización de memoria RAM

Al analizar la documentación de Tensorflow [27] se encontró que en algunos casos las redes neuronales implementadas en Tensorflow pueden llegar a optimizar el uso de memoria, esto aplica a los casos donde el flujo de datos en las redes sigue un curso lineal y no se utilizan todas las capas de la red al mismo tiempo, logrando que no toda la red esté alocada en memoria al mismo tiempo, permitiendo disminuir la cantidad de memoria RAM utilizada. En la figura 2.10 se observa que las redes neuronales SSD utilizadas siguen una estructura lineal en el flujo de datos desde la entrada de la red hasta la salida, además en el documento estudiado se menciona la presencia de uso de optimización en las redes MobileNet, correspondiente a la clase de red que se utilizó en este trabajo, por lo tanto se puede concluir que en las pruebas realizadas se realiza una optimización en el uso de memoria a medida que se realizan iteraciones.

Al observar los casos por separados en la figura 4.7 se tiene que al utilizar GPU la optimización de memoria RAM ocurre entre los 5 y 10 rostros, mientras que al utilizar solo CPU la optimización ocurre entre los 10 y 15 rostros, dando a concluir que al utilizar GPU se optimiza antes el uso de memoria. En otro aspecto, al usar GPU en el cómputo se utiliza aproximadamente 2GB de memoria RAM, mientras que al usar solo CPU se utiliza aproximadamente 1GB de RAM, siendo la de memoria utilizada en comparación a las pruebas donde se utilizó GPU. Y por último, una observación importante corresponde al crecimiento aproximadamente lineal de uso de memoria una vez ocurre la optimización, esto se puede deber al aumento de rostros detectados y que tienen que pasar el seguidor de rostros y a la estructuras de datos dentro del programa que aumentan en tamaño en relación a la cantidad de rostros presentes en las imágenes, este aumento es aproximadamente de 100MB en el caso con 150 rostros en comparación al punto de inicio después de la optimización de memoria.

4.2.4 Tiempo promedio por iteración

En este último parámetro de evaluación se mide el tiempo promedio por cada iteración realizada por el algoritmo, considerando 100 imágenes por cada prueba las cuales van desde 1 hasta 150 rostros, es decir que para 1 rostro se realizan 100 iteraciones y luego se aumentó la cantidad de rostros en 1 hasta llegar a los 150 rostros por imagen.

En la figura 4.8a se observa el tiempo promedio por iteración al usar la red neuroanl SSD 320 y solo CPU para el cómputo, en este caso entre 1 y 5 rostros el tiempo se acota al rango de 0.1 y 0.105 segundos, los tiempos promedios máximos se observan entre los 10 y los 37 rostros llegando a un valor 0.125 segundos, y a partir de los 40 rostros el tiempo promedio se acota al rango 0.095 y 0.11 segundos. Cabe destacar que para esta red el valor F1 disminuye

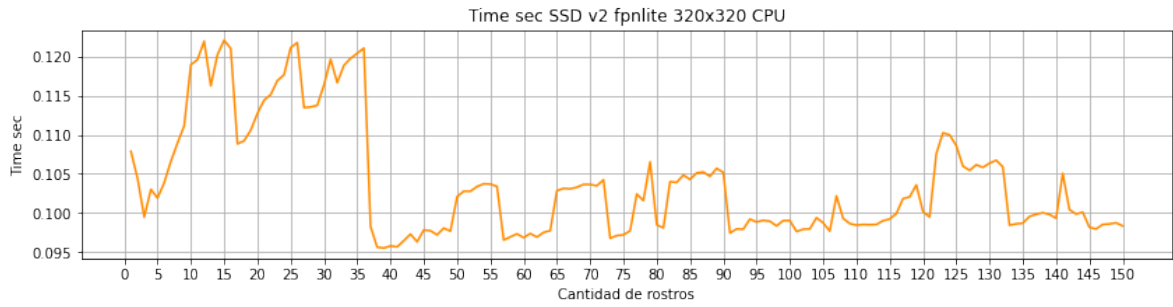
menos de 0.1 desde los 37 rostros según la figura 4.1a, lo que implica que desde este punto la red neuronal deja de detectar correctamente los rostros y lo que indica porque el tiempo promedio de iteraciones disminuye desde este punto, donde el algoritmo deja de procesar una gran cantidad de rostros ya que la mayoría de estos no son detectados y por lo tanto el tiempo de procesamiento disminuye. Por último, los tiempos promedio de iteración se mantuvieron siempre entre los 0.95 y 0.125 segundos para todas las pruebas variando la cantidad de rostros.

En el segundo caso, en la figura 4.8b se observa los resultados de la misma red SSD 320 pero ahora utilizando GPU en los cálculos computacionales. Al igual que el caso anterior, los tiempos promedios obtuvieron máximos hasta antes de las pruebas realizadas con 37 rostros obteniendo valores de 0.28 segundos, después de aumentar la cantidad de rostros por sobre los 37 el tiempo disminuye a valores entre 0.21 y 0.22 segundos, repitiéndose el patrón observado en el caso anterior. En esta prueba donde se usó GPU se obtuvo rangos de tiempo entre los 0.21 y 0.28 segundos, siendo mas de el doble que al solo utilizar CPU para la misma red.

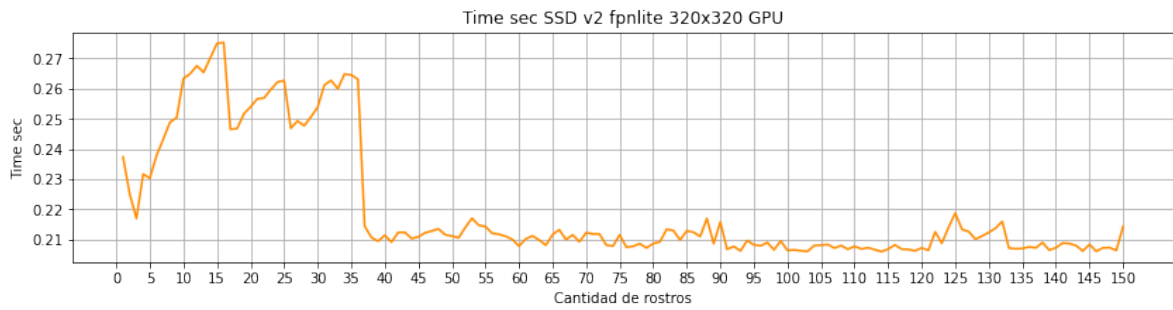
La figura 4.8c se presenta los resultados del tiempo promedio por iteración para la red neuronal SSD 640 utilizando solo CPU para los cálculos, en este caso se presenta un aumento del tiempo desde aproximadamente los 0.14 segundos con 1 rostro por imagen, hasta los 0.25 segundos con cerca de 80 rostros por imagen. A partir de los 80 rostros el tiempo promedio disminuye a un rango entre 0.2 y 0.22 segundos hasta los 100 rostros, y a partir de este punto el tiempo disminuye al rango 0.14 a 0.16 segundos. Al contrastar estos resultados con el desempeño de esta red en la figura 4.3 se tiene que hasta los 80 rostros el valor F1 se mantiene por sobre 0.8, luego entre los 80 y 100 rostros el valor F1 baja a 0.5 y finalmente desde los 100 rostros el valor F1 disminuye bajo 0.1, coincidiendo estos puntos con los del tiempo promedio por iteración, esto quiere decir que el algoritmo llega a un límite de desempeño en la cantidad de rostros detectados correctamente, provocando una menor cantidad de rostros a procesar y por lo tanto disminuyendo los tiempos de procesamiento. Para este caso, en todas las pruebas realizadas, se obtuvo un tiempo de procesamiento promedio por iteración en el rango 0.14 a 0.24 segundos, alcanzado el máximo en los 80 rostros.

En el último caso utilizando la red neuronal SSD 640 con GPU en la figura 4.8d el promedio de tiempo tiene un mínimo aproximado de 0.15 segundos en imágenes con 2 o 3 rostros, el máximo ocurre cerca de los 30 rostros donde se alcanza valores de tiempo por sobre los 0.35 segundos, los cuales se mantienen hasta los 80 rostros donde el tiempo disminuye al rango 0.25 a 0.30 segundos, donde calza la disminución del valor F1 para esta prueba realizada según la figura 4.4. En todas las pruebas con este caso el tiempo promedio por iteración se mantuvo por debajo de los 0.4 segundos.

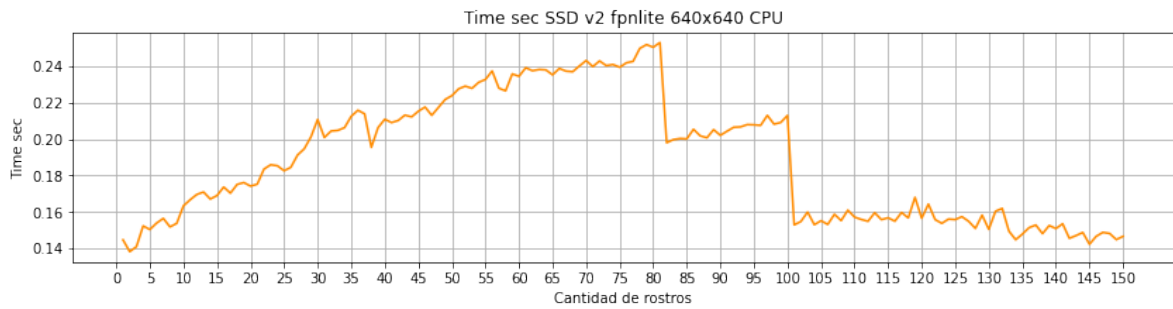
Para los resultados del tiempo de procesamiento por iteración se observó una dependencia



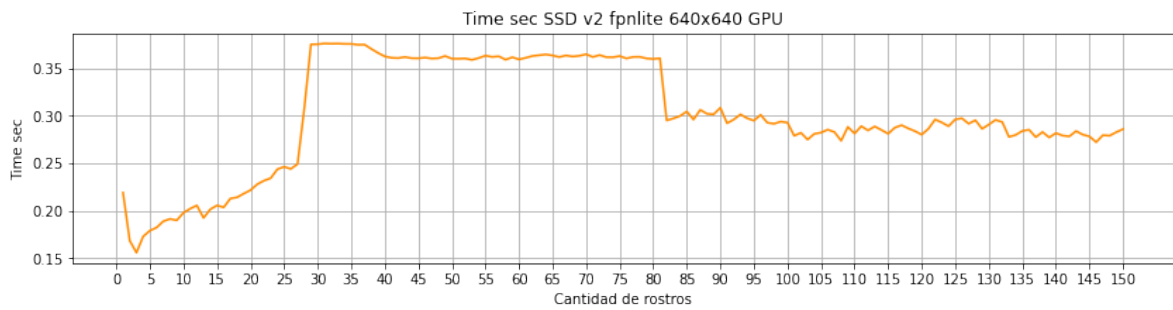
(a) SSD MobileNet V2 FPNLite 320x320 con CPU



(b) SSD MobileNet V2 FPNLite 320x320 con GPU



(c) SSD MobileNet V2 FPNLite 640x640 con CPU



(d) SSD MobileNet V2 FPNLite 640x640 con GPU

Figura 4.8: Tiempo promedio por iteración

del desempeño del detector de rostros, dependiendo de si la red neuronal es capaz de detectar una mayor o menor cantidad de rostros es el tiempo que demora el algoritmo por cada iteración en procesar todos los rostros obtenidos y aplicar algoritmos como el de seguimiento y en como se muestran los resultados. Se observó por ejemplo en la red SSD 640 un mayor tiempo promedio respecto a las red SSD 320 dado que la primera logra reconocer una mayor cantidad de rostros correctamente, por otro lado se obtuvo que las redes implementadas solamente en CPU tienen un menor tiempo promedio por iteración a comparación de las redes ejecutadas con GPU en el caso de este hardware en específico.

4.3 Resultados generales

De los resultados presentados, se puede destacar en una primera instancia el mejor desempeño del algoritmo al utilizar la red neuronal SSD 640, donde las pruebas realizadas se obtuvo valores sobre 0.9 en precisión, exhaustividad y valor F1 hasta los 65 rostros, y se mantuvo por sobre 0.8 en los 3 factores hasta los 80 rostros, a comparación de la red neuronal SSD 320 que logró valores en estos factores sobre 0.9 solo hasta los 15 rostros, disminuyendo después hasta valores inferiores a 0.6 en los factores mencionados. Además, utilizando estos factores se puede determinar la cantidad de rostros seguidos correctamente, dado que la correcta detección de rostros confirma el seguimiento de rostros asociado a una posición dentro la imagen procesada, a la vez que la falta de detección de un rostro o las falsas detecciones implican un mal seguimiento.

En el aspecto de uso de recursos, la red neuronal SSD 320 obtuvo menor utilización de CPU con un promedio de 25 % en comparación a la red SSD 640 con un promedio de 35% cuando solo se usó CPU, al utilizar GPU los rangos de utilización de CPU de ambas redes fue cercano al 10%. En relación a la utilización de GPU, la red neuronal SSD 640 obtuvo un máximo superior al 90% a diferencia de la red neuronal SSD 320 que alcanzó un máximo inferior al 70%. Mientras que en el uso de memoria RAM al comparar entre las redes neuronales utilizadas ambas obtuvieron valores similares, la diferencia se observa al comparar el uso de GPU y CPU, dado que las redes obtuvieron un rango de utilización cercano a 1GB al utilizar CPU, y 2GB al utilizar GPU, además se observó la optimización de memoria RAM después de un tiempo de ejecución del algoritmo.

Finalmente el tiempo procesamiento promedio se ligó al desempeño de las redes neuronales, resultando en un aumento del tiempo en las pruebas que presentan una mayor cantidad de rostros detectados correctamente. Por otro lado, las pruebas realizadas con solo

CPU obtuvieron un menor tiempo promedio que las pruebas realizadas con GPU, siendo la prueba con menor tiempo promedio por iteración la red neuronal SSD 320 con solo CPU con un tiempo entre 0.95 y 0.13 segundos, y la prueba con mayor tiempo promedio por iteración la realizada con la red neuronal SSD 640 y GPU con un tiempo entre 0.15 y 0.4 segundos.

En comparación, la prueba con mejores resultados tanto en desempeño como en uso de recursos computacionales es la realizada con la red neuronal SSD 640 y utilizando CPU, dado que en este caso se logró una mayor cantidad de rostros detectados correctamente, con un tiempo menor y una menor cantidad de memoria RAM utilizada en comparación a la prueba de la misma red con GPU en los cálculos, pero teniendo un mayor costo en la utilización de CPU.

Por otro lado, al estudiar la documentación entregada por el fabricante de la tarjeta gráfica se obtuvo que este dispositivo tiene 640 núcleos CUDA con una frecuencia de operación de 1354 MHz, mientras que la CPU tiene 6 núcleos a una frecuencia de operación base de 2.6 GHz y una máxima de 4.5GHz, y se infiere que la CPU al tener una mayor frecuencia de operación logra obtener un menor tiempo en la predicción de los rostros detectados.

Conclusiones

El proceso de generar un algoritmo capaz de detectar rostros se puede abordar desde diferentes soluciones, en este caso se implementó redes neuronales dado que son las recomendadas en el estado del arte de este tema. Sin embargo, dada la naturaleza de esta solución, el uso de recursos computacionales suele presentar un problema dado los requerimientos de ciertas redes neuronales, durante el desarrollo de esta memoria y de las diferentes pruebas realizadas se presenció este problema en relación a la memoria RAM disponible, no se logró implementar algunos tipos de redes neuronales y las que se presentan en este trabajo son las que se logró entrenar y utilizar exitosamente.

Otro aspecto importante a considerar en este trabajo es la aplicación final del programa desarrollado, el detectar rostros para luego implementarlo en un detector de emociones conlleva realizar una captura y procesamiento en un tiempo tal que se logre almacenar el cambio en las facciones del rostro al formar alguna emoción, es por esto que se estableció un tiempo máximo por cada iteración de 0.5 segundos [9] dado que este es el tiempo mínimo para crear la expresión facial mas rápida asociada a una emoción, y en los resultados presentados se observó que el tiempo promedio máximo menor a 0.4 segundos en el peor de los casos al utilizar la red neuronal SSD 640 con GPU, logrando satisfacer el requerimiento mínimo del trabajo presentado correspondiente a los 0.5 segundos.

En relación al desempeño en detección de rostros se logró mostrar el comportamiento en diferentes factores de desempeño comparando 2 redes neuronales y variando la cantidad de rostros presentes en las imágenes a procesar. La red neuronal SSD 640 demostró mejores resultados en comparación a las red neuronal SSD 320, logrando factores precisión, exhaustividad y *F1* por sobre 0.8 hasta los 80 rostros presentes en una imagen, en comparación a las red SSD 320 que logra los factores por 0.8 solo hasta los 15 rostros. Concluyendo que la red neuronal SSD 640 logra mejores resultados sin utilizar GPU y con una cantidad de memoria RAM similar en relación a la utilizada por la red SSD 320, pero con un mayor porcentaje de utilización de CPU.

En relación a los objetivos presentados para este trabajo se logró detectar exitosamente hasta los 80 rostros por imagen con márgenes de detecciones cerca del 80% correctas, también se logró almacenar los rostros almacenados identificando los correspondientes a un mismo rostro creando un vídeo para cada rostros en seguimiento. Se logró un tiempo de procesamiento máximo de 0.4 segundos, inferior al límite de 0.5 segundos establecido para la generación de las emociones mas rápidas. Además se logró implementar una interfaz para que el usuario pueda iniciar la captura de imágenes, donde también se presentan los rostros detectados y se puede definir el mínimo de certeza respecto a la probabilidad de que un rostros detectado por la red sea considerado como un rostro real.

Finalmente, se logró caracterizar el comportamiento de los recursos computacionales utilizados en los diferentes casos presentados, tales como el uso de CPU, GPU y memoria RAM al utilizar una red neuronal SSD 320 o la red SSD 640, y al realizar los cálculos computacionales utilizando solo CPU o complementando con GPU, y también se presentó las variaciones en los recursos computacionales al procesar imágenes con diferentes cantidades de rostros presentes.

Referencias

- [1] ZHU Yanjia , CAI Hongxiang , ZHANG Shuhan , WANG Chenhao , XIONG Yichao. Tinaface: Strong but simple baseline for face detection. [En línea] arXiv. 2021. <<https://arxiv.org/pdf/2011.13183.pdf>> [Consulta: 15 Mayo 2021].
- [2] HE Kaiming , ZHANG Xiangyu , REN Shaoqing , SUN Jian. Deep residual learning for image recognition. [En línea] arXiv. 2015. <<https://arxiv.org/pdf/1512.03385.pdf>> [Consulta: 15 Mayo 2021].
- [3] YANG , Shuo , LUO, Ping , CHANGE , Chen , TANG , Xiaoou. WIDER FACE: A Face Detection Benchmark. [En línea] arXiv. 2015. <<https://arxiv.org/pdf/1511.06523.pdf>> [Consulta: 01 Agosto 2021].
- [4] CHUN Lin , DIAN Li , ZHI Jiang , JING Wang , ZHANG Chao. Yolov3: Face detection in complex environments. [En línea] International Journal of Computational Intelligence Systems. 2020. <<https://www.atlantis-press.com/journals/ijcis/125943382/view>> [Consulta: 15 Mayo 2021].
- [5] REDMON Joseph , FARHANDI Ali. Yolov3: An incremental improvement. [En línea] arXiv. 2018. <<https://arxiv.org/pdf/1804.02767.pdf>> [Consulta: 15 Mayo 2021].
- [6] GOWDA Dhanush , VISHAL K , B. Keertiraj , KUMARI Neha, M. Pooja. Face recognition based attendance system. [En línea] International Journal of Engineering Research & Technology (IJERT). 2020. <https://www.researchgate.net/publication/341876647_Face_Recognition_based_Attendance_Management_System> [Consulta: 15 Mayo 2021].
- [7] KING Davis. Dlib-ml: A machine learning toolkit. [En línea] Journal of Machine Learning Research. 2009. <<https://www.jmlr.org/papers/volume10/king09a/king09a.pdf>> [Consulta: 15 Mayo 2021].

- [8] LUA Peng , SONGA Baoye , XUB Lin. Human face recognition based on convolutional neural network and augmented dataset. [En línea] SYSTEMS SCIENCE & CONTROL ENGINEERING: AN OPEN ACCESS JOURNAL. 2020. <https://www.researchgate.net/publication/346429475_Human_face_recognition_based_on_convolutional_neural_network_and_augmented_dataset> [Consulta: 15 Mayo 2021].
- [9] MATSUMOTO , David , SUNG H , Hyi. Reading facial expressions of emotion. [En línea] American Psychological Association. Mayo 2011. <<https://www.apa.org/science/about/psa/2011/05/facial-expressions>> [Consulta: 15 Mayo 2021].
- [10] CLARK , Alex. Pillow (pil fork) documentation. [En línea] readthedocs. 2015. <<https://buildmedia.readthedocs.org/media/pdf/pillow/latest/pillow.pdf>> [Consulta: 3 Junio 2021].
- [11] BRADSKI , Gary. Open source computer vision library. [En línea] opencv.org.2015. <<https://github.com/itseez/opencv>> [Consulta: 3 Junio 2021].
- [12] MIJIWEL Maad M., ESEN Adam, SHAMI Aysar. Overview of neural networks. [En línea] ResearchGate.2019. <https://www.researchgate.net/publication/332655457_Overview_of_Neural_Networks/link/5cc1f626299bf120977f7989/download> [Consulta: 05 Junio 2022].
- [13] GROSSIA Enzo, BUSCEMA Massimo. Introduction to artificial neural networks. [En línea] ResearchGate.2019. <https://www.researchgate.net/publication/5847739_Introduction_to_artificial_neural_networks> [Consulta: 05 Junio 2022].
- [14] GARG Saksham. A study on the structure of neural networks and the mathematics behind backpropagation. [En línea] ResearchGate.2019. <https://www.researchgate.net/publication/342412635_A_Study_on_the_Structure_of_Neural_Networks_and_the_Mathematics_behind_Backpropagation> [Consulta: 05 Junio 2022].
- [15] SHARMA Siddharth , SHARMA Simone , ATHAIYA Anidhya. Activation functions in neural networks. [En línea] International Journal of Engineering Applied Sciences and Technology.2020. <<http://ijeast.com/papers/310-316,Tesma412,IJEAST.pdf>> [Consulta: 05 Junio 2022].

- [16] HOSSAIN Ali , RAHMAN Mijanur , KUMAR Uzzal, KHAN Farukuzzaman. Implementation Of Back-Propagation Neural Network For Isolated Bangla Speech Recognition. [En línea] International Journal of Information Sciences and Techniques. 2020. <<https://arxiv.org/ftp/arxiv/papers/1308/1308.3785.pdf>> [Consulta: 23 Agosto 2021].
- [17] DU Simon , LEE Jason , LI Haochuan , WANG Liwei , ZHAI Xiyu. Gradient descent finds global minima of deep neural networks. [En línea] Proceedings of the 36 th International Conference on Machine Learning.2020. <<http://proceedings.mlr.press/v97/du19c/du19c.pdf>> [Consulta: 05 Junio 2022].
- [18] O'SHEA Keiron , NASH Ryan. An introduction to convolutional neural networks. [En línea] arXiv.2015. <<https://arxiv.org/pdf/1511.08458.pdf>> [Consulta: 05 Junio 2022].
- [19] LIU , Wei , ANGUELOV , Dragomir , ERHAN , Dumitru , SZEGEDY , Christian , REED , Scott , FU , Cheng-Yang , BERG , Alexander C. SSD: Single Shot MultiBox Detector. [En línea] arXiv. 2016. <<https://arxiv.org/pdf/1512.02325.pdf>> [Consulta: 15 Julio 2021].
- [20] HOSANG Jan , BENENSON Rodrigo , SCHIELE Bernt. Learning non-maximum suppression. [En línea] arXiv.2017. <<https://arxiv.org/pdf/1705.02950.pdf>> [Consulta: 08 Junio 2022].
- [21] YU , Hongkun , CHEN , Chen , DU , Xianzhi , LI Yeqing , RASHWAN , Abdullah , HOU , Le , JIN , Pengchong , YANG , Fan , LIU , Frederick , KIM , Jaeyoun , LI , Jing. TensorFlow Model Garden. [En línea] GitHub. 2020. <<https://github.com/tensorflow/models>> [Consulta: 01 Agosto 2021].
- [22] HUANG Jonathan , RATHOD Vivek , SUN Chen , ZHU Menglong , KORATTIKARA Anoop , FATHI Alireza , FISCHER Ian , WOJNA Zbigniew , SONG Yang , GUADARRAMA Sergio , MURPHY Kevin. Speed/accuracy trade-offs for modern convolutional object detectors. [En línea] arXiv.2017. <<https://arxiv.org/pdf/1611.10012.pdf>> [Consulta: 08 Junio 2022].
- [23] LIN , Tsung-Yi , MAIRE , Michael , BELONGIE , Serge , BOURDEV , Lubomir , GIRSHICK , Ross , HAYS , James , PERONA , Pietro , RAMANAN , Deva , ZITNICK , C. Lawrence , DOLLÁR , Piotr. Microsoft COCO: Common Objects in Context.

- [En línea] arXiv. 2015. <<https://arxiv.org/pdf/1405.0312.pdf>> [Consulta: 15 Agosto 2021].
- [24] PADILLA Rafael ,NETTO Sergio , SILVA Eduardo. A Survey on Performance Metrics for Object-Detection Algorithms. [En línea] International Conference on Systems, Signals and Image Processing (IWSSIP). 2020. <<https://ieeexplore-ieee-org.usm.idm.oclc.org/document/9145130>> [Consulta: 20 Mayo 2022].
- [25] DAVIS Jesse , GOADRICH Mark. The Relationship Between Precision-Recall and ROC Curves. [En línea] Proceedings of the 23 rd International Conference on Machine Learning, Pittsburgh, PA. 2006. <<https://www.biostat.wisc.edu/~page/rocpr.pdf>> [Consulta: 20 Mayo 2022].
- [26] HUANG Gary, RAMESH Manu , BERG Tamara , LEARNED-MILLER Erik. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. [En línea] University of Massachusetts, Amherst, Technical Report 07-49. 2007. <<http://vis-www.cs.umass.edu/lfw/lfw.pdf>> [Consulta: 15 Abril 2022].
- [27] LEE Juhyun Lee , PISARCHYK Yury. Optimizing TensorFlow Lite Runtime Memory. [En línea] Tensorflow blog. 2020. <<https://blog.tensorflow.org/2020/10/optimizing-tensorflow-lite-runtime.html>> [Consulta: 15 Julio 2022].