ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA DE SISTEMAS

APLICACIÓN DE TÉCNICAS DE APRENDIZAJE DE MÁQUINA PARA LA DETECCIÓN DE AGLOMERACIONES DE PERSONAS Y COMPORTAMIENTOS ANÓMALOS MEDIANTE SEGMENTACIÓN SEMÁNTICA Y REDES CONVOLUCIONALES

MODELO DE REDES CONVOLUCIONALES BASADAS EN REGIONES PARA LA DETECCIÓN DE AGLOMERACIONES EN VIDEOS APLICANDO SEGMENTACIÓN SEMÁNTICA.

TRABAJO DE INTEGRACIÓN CURRICULAR PRESENTADO COMO REQUISITO PARA LA OBTENCIÓN DEL TÍTULO DE INGENIERÍA EN CIENCIAS DE LA COMPUTACIÓN

CÉSAR PAOLO TACO APOLO
cesar.taco@epn.edu.ec

DIRECTOR: ING. HENRY PATRICIO PAZ ARIAS Msc.
henry.paz@epn.edu.ec

DMQ, septiembre 2022

CERTIFICACIONES

Yo, CÉSAR PAOLO TACO APOLO declaro que el trabajo de integración curricular aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

CÉSAR PAOLO TACO APOLO

Certifico que el presente trabajo de integración curricular fue desarrollado por CÉSAR PAOLO TACO APOLO, bajo mi supervisión.

ING. HENRY PATRICIO PAZ ARIAS Msc
DIRECTOR

DECLARACIÓN DE AUTORÍA

A través de la presente declaración, afirmamos que el trabajo de integración curricular aquí descrito, así como el (los) producto(s) resultante(s) del mismo, son públicos y estarán a disposición de la comunidad a través del repositorio institucional de la Escuela Politécnica Nacional; sin embargo, la titularidad de los derechos patrimoniales nos corresponde a los autores que hemos contribuido en el desarrollo del presente trabajo; observando para el efecto las disposiciones establecidas por el órgano competente en propiedad intelectual, la normativa interna y demás normas.

CÉSAR PAOLO TACO APOLO

ING. HENRY PATRICIO PAZ ARIAS Msc

DEDICATORIA

A Miriam, mi madre,

a César, mi padre,

a Thiago, mi hijo,

a Mateo, mi hermano,

y a toda mi familia, quienes siempre creyeron en mí.

AGRADECIMIENTO

En primer lugar, quiero dar gracias a Dios por las bendiciones en cada paso de este bravío camino y darme el paso de poder culminar una de las tantas metas en mi vida.

Gracias a mi madre, mi padre, mi hermano y toda mi familia, por la confianza y el apoyo que me han dedicado a lo largo de todo este arduo camino, por toda la dedicación y esfuerzo que con tanto anhelo han realizado para ayudarme tanto emocional como económicamente en mi vida universitaria, quiero que noten que siempre he contemplado su voz de aliento y su mano protectora en los momentos más difíciles. Los amo.

Quiero pronunciar el más sincero agradecimiento a mi director Msc. Henry Paz Arias, el cual fue el principal colaborador durante todo este proceso en rumbo a mi desarrollo como profesional, quien con su guía y cooperación dio paso al proceso y culminación de este trabajo integrador.

A la facultad de Sistemas por aprobar la realización de este trabajo y a todos los profesores que impartieron sus conocimientos y experiencias a lo largo de toda mi etapa estudiantil, aportando con fervor a mi formación integral.

A mis compañeros y a quienes resultaron en verdaderos amigos, gracias por forjar de esta etapa de mi vida en una de las excelentes experiencias, me llevo lo mejor de cada uno de ustedes.

César Paolo Taco Apolo

ÍNDICE DE CONTENIDO

CERTIFICACIONES I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
RESUMEN VI
ABSTRACT VII
1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO 1
1.1 Objetivo general 2
1.2 Objetivos específicos 2
1.3 Alcance 2
1.4 Marco teórico 4
2 METODOLOGÍA 17
2.1 Metodología Action Research 18
2.2 Análisis exploratorio y validación de la calidad de los datos. 19
2.3 Selección, limpieza y construcción de los datos 23
2.4 Creación del modelo de redes neuronales convolucionales basado en un modelo previamente entrenado 28
2.5 Incorporación del algoritmo de distanciamiento 39
3 PRUEBAS, RESULTADOS, CONCLUSIONES Y RECOMENDACIONES 44
3.1 Pruebas 44
3.2 Resultados 46
3.3 Conclusiones 54
3.4 Recomendaciones 55
4 REFERENCIAS BIBLIOGRÁFICAS 56
5 ANEXOS 61

RESUMEN

La visión por computador es una rama perteneciente al campo de la inteligencia artificial que engloba tanto la teoría como los insumos tecnológicos necesarios para poder emular la percepción visual humana, teniendo como objetivo el construir sistemas artificiales capaces de interpretar escenas naturales o datos multidimensionales. La correcta detección y segmentación de personas en lugares altamente aglomerados es de vital importancia debido a la enfermedad que provocó la muerte de miles de millones de personas a nivel mundial conocida como COVID 19.

Por tal razón en este trabajo integrador se presenta la aplicación de una técnica de visión por computador conocida como segmentación semántica, basándose en el análisis, diseño e implementación de un sistema para la detección y segmentación de personas en un lugar altamente concurrido y aglomerado proveniente de un video situado a unos 5 metros de altura aproximadamente, teniendo como punto de partida un modelo previamente entrenado el cual se lo conoce por el nombre de Mask R-CNN, con este modelo se efectuará la transferencia de aprendizaje en sus capas superiores para que únicamente detecte y segmente personas.

El estudio consiste en analizar, valorar y evaluar el modelo cuyo proceso de entrenamiento será analizado en base a métricas relacionadas entorno a lo que dicta el estado del arte, como, por ejemplo, su precisión media y su intersección sobre la unión, los cuales determinaran la validez del modelo para concluir si en conjunto con la segmentación semántica son capaces para detectar y segmentar personas en lugares altamente aglomerados.

PALABRAS CLAVE: segmentación Semántica, aglomeración, visión computacional, Mask R-CNN, segmentación de instancias, distanciamiento social.

ABSTRACT

Computer vision is a branch of the field of artificial intelligence that develops the theory and technology necessary to emulate human visual perception, aiming to build artificial systems capable of interpreting natural scenes or multidimensional data. The correct detection and segmentation of people in highly crowded places is of vital importance due to the disease that caused the death of billions of people worldwide known as COVID 19.

For this reason, this curricular integration work presents the application of a computer vision technique known as semantic segmentation, based on the analysis, design and implementation of a system for the detection and segmentation of people in a highly crowded and agglomerated place from a video located approximately 5 meters high, taking as a starting point a previously trained model which is Mask R-CNN, with this model the transfer of learning in its upper layers will be carried out so that only detects and segments people.

The study consists of analyzing, assessing and evaluating the model whose training process will be analyzed based on metrics related to the state of the art, such as, for example, its average accuracy and its intersection over the union, which will determine the validity of the model to conclude whether in conjunction with semantic segmentation are able to detect and segment people in highly agglomerated places.

KEYWORDS: semantic segmentation, agglomeration, computational vision, Mask R-CNN, instance segmentation, social distancing.

1 DESCRIPCIÓN DEL COMPONENTE DESARROLLADO

La detección de aglomeraciones o amontonamiento grande y desordenado de personas, especialmente, en un lugar reducido tanto abierto como cerrado, toma mucho impacto en la actualidad debido a la enfermedad que provocó la muerte de miles de millones de personas a nivel mundial conocida como COVID 19, debido a esto su detección es de vital importancia para mantener un ambiente aislado y no concurrido para evitar contagios no deseados de la enfermedad y así procurar por el bienestar y la seguridad de las personas que transiten en dicho lugar que es propenso a ser altamente aglomerado, Iglesias-Osores, S. (2020).

Por esta razón la creación de un modelo de Redes Neuronales Convolucionales para la detección y segmentación de aglomeraciones de personas en videos ayudaría a evitar contagios no deseados en beneficio de la salud de las personas en la comunidad.

Lo que se cuestiona dentro de este proyecto integrador es lo siguiente:

- ¿Es posible aplicar la técnica de segmentación e instanciación semántica en un lugar sumamente concurrido y aglomerado?

Con esta pregunta se obtienen las siguientes interrogantes:

- ¿Es posible crear un modelo que detecte y segmente aglomeración de personas, obteniendo el procesamiento de imágenes que tienen como fuente un video de una cámara situada a una altura de entre 3 a 8 metros de altura?

- ¿La técnica de segmentación e instanciación semántica puede ser utilizada para el conteo y detección de personas en lugares altamente aglomerados?

- ¿Hasta qué distancia, en tanto altura y profundidad, las técnicas de segmentación semántica son capaces de entregar resultados óptimos conforme lo establecido anteriormente?

- ¿Cuáles son las limitaciones reales que se enfrenta este tipo de modelos de aprendizaje de máquina con las técnicas anteriormente mencionadas?

Este componente busca dar respuesta a estas interrogantes mediante la investigación respecto a todo según corresponda al estado del arte en visión artificial y la aplicación de segmentación e instanciación semántica, en el entrenamiento de un modelo de detección, segmentación, instanciación y conteo de personas en lugares altamente aglomerados provenientes de un video, el cual constará de etapas de entrenamiento, validación y testeo

para que el mismo sea un producto fiable y de alto rendimiento dentro de los parámetros y limitaciones que se detallan en el alcance de este proyecto integrador.

1.1 Objetivo general

Aplicar Segmentación semántica en el entrenamiento de un modelo de redes neuronales convolucionales previamente entrenado en base a la detección de aglomeraciones de personas involucradas dentro de una zona delimitada por un video.

1.2 Objetivos específicos

1. Investigar sobre visión computacional, procesamiento de videos, segmentación e instancia semántica y detección de objetos.
2. Analizar la probabilidad de que el producto sea reproducible y reproducir el modelo de forma que quede a disposición de ser entrenado y validado.
3. Desarrollar un modelo de redes neuronales convolucionales recurrentes mediante técnicas de segmentación semántica.
4. Evaluar el desempeño del modelo desarrollado mediante técnicas de segmentación semántica.

1.3 Alcance

Action-Research (AC) será la metodología a utilizar para el desarrollo de este componente; como su nombre lo indica y enfocándose en la premisa de (Dick, 2011)., action-research es una metodología que tiene un dualismo entre la investigación y la acción; la investigación para incrementar la comprensión por parte del investigador o una comunidad más amplia y la acción para traer cambios en alguna comunidad, organización o programa. La metodología surge en los trabajos de Kurt Lewis, en donde definió AC como un proceso cíclico de exploración, actuación y valoración de los resultados. Se escoge esta metodología ya que es una manera de producir nuevos conocimientos a partir de la práctica; además, su principal motivación es la búsqueda de soluciones a problemas del mundo real. Específicamente para el desarrollo de este proyecto la metodología está basada en el libro “Action Research” de Ernest Stringer, tercera edición.

El alcance por parte de la detección de aglomeraciones de personas mediante segmentación semántica radica en poder entrenar y evaluar un modelo de aprendizaje de máquina con redes neuronales convolucionales basadas en regiones a partir de un modelo pre entrenado con la finalidad de poder determinar el conteo y segmentación de cada

persona involucrada dentro de una zona delimitada por un video, juntamente con toda la revisión de la literatura en el estado del arte de la aplicación de técnicas de aprendizaje de máquina para poder optimizar dicho modelo y que el mismo contenga la mínima cantidad de errores en cuanto a conteo y segmentación de las personas.

En este contexto el modelo de aprendizaje de máquina a realizar consta de cuatro fases:

1. Análisis exploratorio y validación de la calidad de los datos, en cuanto a la revisión de la literatura proporcionada.

2. Selección, limpieza y construcción de datos, los cuales serán provenientes de la herramienta OIDv4 desarrollada por (Papadopoulos, 2016) y el conjunto de datos implementados por (Lin, 2014, September) para alimentar el conjunto de datos en base a un número determinado de imágenes para la implementación del modelo de aprendizaje de máquina, 400 y 100 imágenes para entrenamiento y testeo respectivamente.

3. Creación del modelo de redes neuronales convolucionales, con la ayuda de un modelo previamente entrenado basándose en lo que dictan (He, 2017) y (Liu, 2018, December) en sus investigaciones, en donde se modifica los archivos de configuración y la modificación de las primeras capas para que el modelo solo detecte y segmente personas basándose en las técnicas que dicta (Bolya, 2019) y He, K.(2017) como herramienta de segmentación de las imágenes y el algoritmo capaz de calcular el número máximo de personas (3) delimitada por un área de un metro cuadrado.

4. Prueba y medición de resultados del modelo entorno a las siguientes limitaciones:

    a) El video que servirá como entrada del modelo de detección y segmentación estará situado a una distancia de 3 a 8 metros de altura para ser sujeto a análisis.

    b) La zona delimitada para la aglomeración de personas sigue el principio de que por cada metro cuadrado no deben existir más de 3 personas.

    c) El porcentaje de detección de las personas rodea del 15% al 90%, esto se tiene debido al manejo de sobreposición de las imágenes en cuanto al número de personas se tengan en una determinada instancia.

    d) Finalmente, el modelo tendrá alrededor de un 80% a 86% de precisión en cuanto a detección y segmentación de las personas se refiere.

1.4 Marco teórico

En esta sección se establece todo el contexto relacionado al estado del arte en cuanto a aprendizaje automático se refiere por el cual se desarrolló este trabajo basándose en lo técnico y teórico, como también el uso de herramientas y librerías las cuales son fundamentales para el desarrollo del modelo que será sujeto a análisis.

1.4.1 Aprendizaje Automático

Es una rama que existe en el campo de la inteligencia artificial con el fin de que los dispositivos computacionales aprendan sin la necesidad de que se programe explícitamente para ello. Una destreza vital para realizar sistemas orientados a identificar patrones entre la información y los datos para obtener predicciones y análisis, en la figura 1 a continuación se ilustra la jerarquización del campo de la inteligencia artificial (IA).

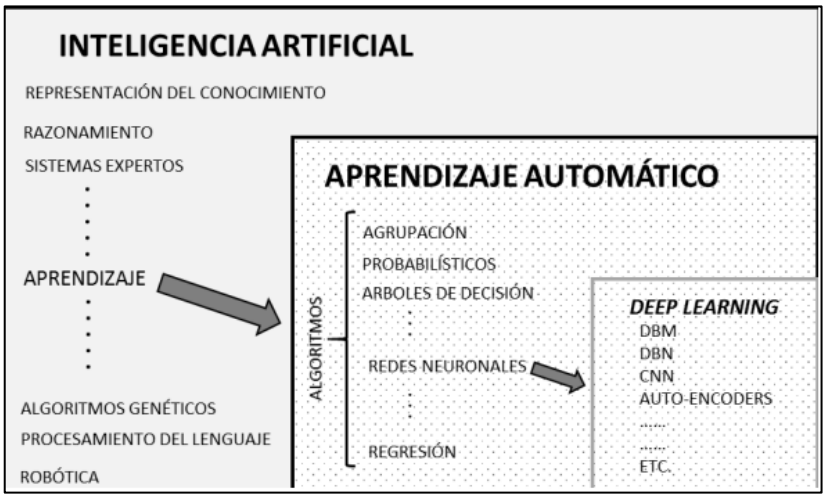


Figura 1. Representación de la jerarquización del campo de la inteligencia artificial [30].

Para lograr el entrenamiento óptimo de los modelos dentro de este campo se tienen tres grandes variantes que se emplean en la actualidad:

El aprendizaje por refuerzo hace referencia a la producción en cuanto una maquina aprende por medio de prueba y error hasta alcanzar el medio óptimo de concretar una asignación determinada de tareas. A través de estos resultados el sistema aprende a modificar y alterar su conducta en base a recompensas para que la resolución de las tareas sea más eficiente, sin la necesidad de generar los scripts (códigos) necesarios para que el sistema realice una tarea en concreto de una forma específica y determinada.

El aprendizaje supervisado ocurre cuando se entrena a las maquinas con información debidamente etiquetadas. Por ejemplo, fotos con atributos o características de los objetos que aparecen en ellas. El algoritmo empleado es capaz de determinar esas mismas

características en otro conjunto de datos que no sea con el cual se ha realizado el proceso de aprendizaje con la finalidad de, por ejemplo, reconocer a personas en imágenes que no están dentro del conjunto de datos por el cual se ha entrenado el sistema.

Por último, se tiene al aprendizaje no supervisado, en donde las máquinas no identifican patrones ni características en conjuntos de datos etiquetados, sino que en su defecto buscan analogías y semejanzas. De esta forma, los algoritmos no son capaces, ni están programados para identificar datos de un objeto en específico, más bien buscan ejemplos que se parezcan y que puedan ser agrupados en base a sus semejanzas. Por ejemplo, en el reconocimiento facial, el algoritmo no busca rasgos y características en concreto del rostro, sino se basa en una serie de patrones comunes que informan que se trata del mismo rostro.

En el presente trabajo integrador, se opta por escoger la segunda de las variantes, es decir, el aprendizaje supervisado. Resumidamente se trata de obtener imágenes en donde existan objetos etiquetados (personas) para su detección y segmentación, con el objetivo de que el modelo vaya aprendiendo las características de los objetos, en base a sus regiones de interés y niveles de pixeles, y una vez concluido el proceso de aprendizaje, comience el proceso de inferencia, en el cual se debe comprobar la validez del entrenamiento del modelo en base a su correcta extracción de características fundamentales de los objetos, más en concreto, personas.

1.4.2 Aprendizaje Profundo

El Deep learning o de su traducción, aprendizaje profundo, es una arista perteneciente al área del Machine Learning (ML). Se trata de un paradigma de aprendizaje que pretende asemejarse al aprendizaje humano para adquirir conocimiento respecto a determinados patrones y tareas [31].

Para el procesamiento de la totalidad de la información, sus redes neuronales artificiales son usadas concatenando una gran variedad de capas conformadas por unidades de procesamiento sencillas (neuronas), las cuales transforman progresivamente la información. Existen capas de entrada a la red, capas ocultas y capas de salida, las cuales en futuras secciones se analiza a detalle la función que tiene cada una de estas capas. Resumidamente la capa de entrada hace referencia a la capa por la cual los datos entraran a la red, las capas ocultas permiten todo el procesamiento de datos en diversas técnicas, y finalmente la capa de salida es quien realiza la toma de decisiones en base al procesamiento realizado en las capas anteriores [31].

1.4.3 Detección de Objetos

Dentro de este apartado lo más habitual está anclado a los problemas de clasificación, lo cual dentro del machine learning se trata de encontrar características y patrones concretos en función a una serie de parámetros con los que se configura el sistema para que el mismo pueda clasificar de una manera adecuada en base a la tarea asignada, es decir, en el caso de que el modelo quiera clasificar imágenes que contengan perros o gatos, lo primero que se tiene que realizar es la alimentación de una gran cantidad de imágenes de perros y gatos, cada uno de estas imágenes debe tener por lo mínimo como atributo una etiqueta que lo identifique, con toda esta información el sistema es capaz de ser entrenado durante un tiempo considerable en base a un determinado número de configuraciones, las cuales se verán más adelante. De esta manera el sistema puede ser alimentado por imágenes diferentes a las mencionadas anteriormente y ser capaz de clasificar dichas imágenes.

A esto se le suma un problema más complejo que es el poder detectar y determinar en qué parte de la imagen exactamente se encuentra el objeto u objetos que se quieren detectar. Para hacer un poco más intuitivo se escoge el ejemplo de un ser humano en su etapa inicial el cual es capaz de clasificar los objetos a su alrededor en base a las imágenes repetitivas que sus ojos y cerebro son capaces de procesar, provenientes de la interacción con sus padres y demás personas a su alrededor, juntamente con las indicaciones de qué y cómo son los objetos que está reconociendo en su edad temprana, esto hace que la conexión que existe entre los ojos humanos y la sinapsis efectuada en el cerebro en conjunto con las millones de millones de neuronas después de los estímulos provenientes del mundo exterior otorgue la capacidad de reconocer un sinfín de objetos similares.

En el apartado de metodología se va a manifestar las partes en las cuales se componen un detector de objetos, como su desarrollo en el proceso de la detección, segmentación e instanciación para que el detector funcione y sea validado dentro de los parámetros establecidos en el alcance de este trabajo integrador.

1.4.4 Red Neuronal

El eje interno de un detector de objetos basado en aprendizaje profundo es la comúnmente conocida red neuronal, la cual es encargada de realizar operaciones y funciones netamente necesarias para la extracción de características de los objetos existentes en una determinada imagen, con el objetivo de que aprenda en base al aprendizaje supervisado y pueda reconocer los objetos establecidos como objetivos y que, de la misma manera, pueda ser capaz de segmentar cada uno de ellos.

Adentrándose en el conocimiento de lo que es una red neuronal, se detalla lo que es una TLU (threshold logic unit) o unidad lógica de umbral. Esta hace referencia a una neurona artificial la cual se compone de una serie de entradas y de una sola salida. Las entradas están ligadas a pesos sinápticos de manera en que la TLU calcule y pondere la suma de dichas entradas para conseguir una salida [29]. Un ejemplo de se aborda en la Figura 2.

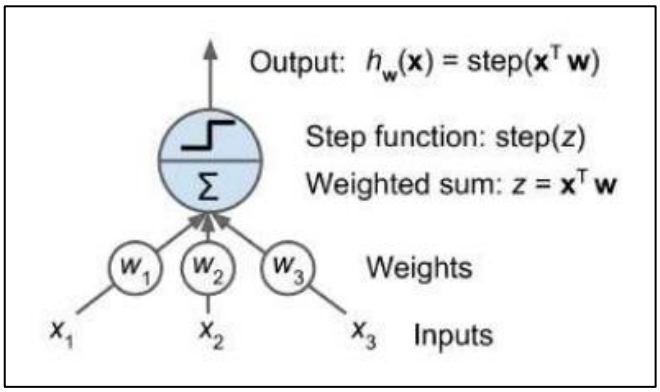


Figura 2. Estructura de un TLU [29].

Un perceptrón en cambio, existe como una capa de TLUs con cada TLU interconectada a la totalidad de sus entradas. En la Figura 3 se observa una serie de TLU que son partidarios de la capa de salida, con la peculiaridad de que están completamente conectadas a sus entradas [29].

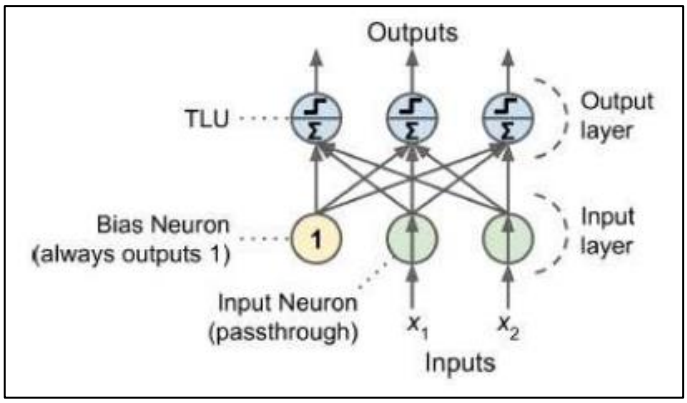


Figura 3. Composición de un perceptrón [29].

En la profundidad de cada TLU, existe una función de activación que se activa en base a la suma promedio de las entradas al tener un valor mayor o menor que un umbral asignado. Si dicha suma es menor al umbral, su salida será nula o -1, en caso de ser mayor al umbral su salida será igual a 1. Las funciones de activación otorgan un resultado en base a los valores que se obtengan en las entradas, estos valores pueden estar entre 0 y 1, -1 y 1, pero no quiere decir que sean las únicas posibles salidas.

Para concluir con este apartado se tienen los MLP, traducido del inglés: multilayer perceptron, capa de entrada que contienen una o más capas ocultas y una capa de salida, en donde las capas cercanas a la entrada se llaman capas inferiores, y las que se ubican más cercanas a la salida se llaman superiores. En la Figura 4 se adjunta un ejemplo.

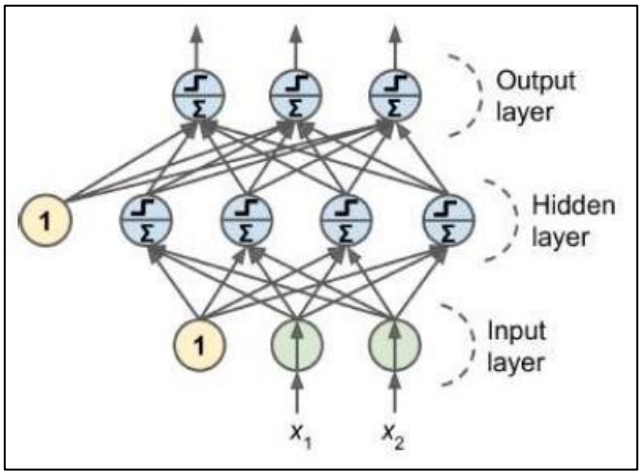


Figura 4. Estructura de un MLP.

1.4.5 Redes Neuronales Convolucionales Profundas

Una de las grandes revoluciones que se obtuvo en los últimos años en cuanto a la inteligencia artificial, visión por computadora y del aprendizaje profundo es el desarrollo de las redes neuronales convolucionales (CNN). Dando una alta eficacia y utilidad que supera con creces los resultados de métodos anteriores a estos, siendo cada vez más común ver aplicaciones con modelos de aprendizaje profundo basados en redes neuronales convolucionales.

Acorde a esto se debe tener mucho en cuenta la morfologia de la fuente de información del sistema ya que en los sistemas de reconocimiento, las imágenes son matrices cuyos valores provienen del espectro RGB(red, green, blue) y con una máscara que son imágenes de valores de tipo bool con regiones definidas de los objetos a detectar.

Con este tipo de redes es importante entender conceptos como padding, traducido al español como relleno, este tipo de operación sirve para el procesamiento de la imagen de una forma equitativa sin eliminar detalles importantes que suelen estar en los bordes de las imágenes, en la figura 5 se detalla un ejemplo de los canales de datos numéricos que estan contenidos en una imagen, a la izquierda se tiene la foto original que entra en la capa de la red neuronal convolucional, en el medio una representación del cojunto de datos que conforman la imagen y a la derecha cada uno de los canales con su respectivo padding [33].

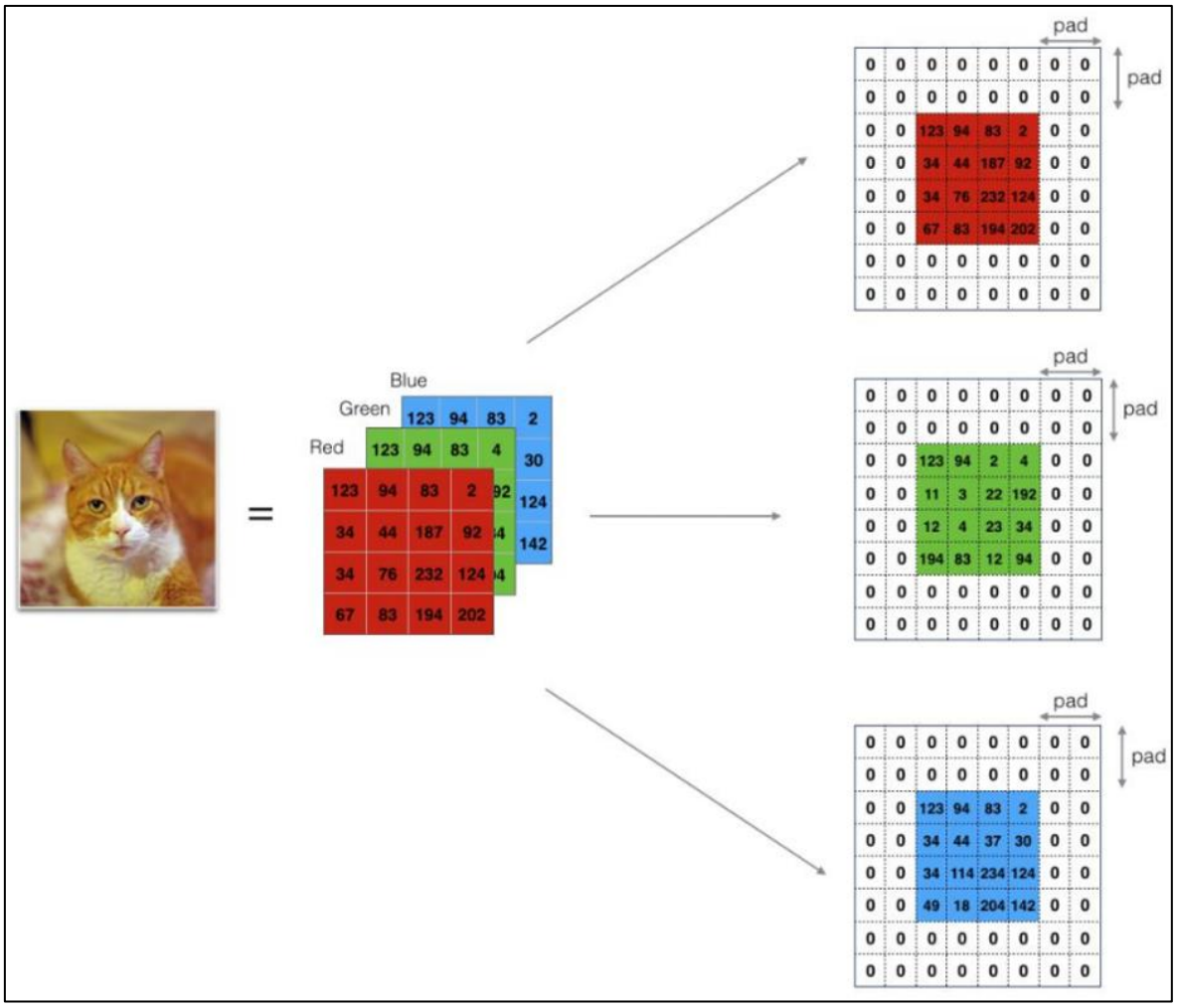


Figura 5. Canales de datos numéricos contenidos en una imagen [33].

Por otro lado, el paso o stride, hace referencia a la frecuencia en que se van a ejecutar las operaciones, es decir, determina el número de pixeles que habrá de distancia entre dos operaciones de los filtros o kernels de las operaciones, un ejemplo se tiene en la convolución. Es imprescindible que el paso no sea ni lo suficientemente pequeño ni que sea demasiado grande para que la información se procese en exceso o pueda ignorar valores relevantes de los pixeles respectivamente. En la figura 6 se observa un ejemplo de convolución con paso de 2, a la izquierda los valores de los canales de la imagen original con dos recuadros del primer y segundo filtro, en el recuadro verde se representa el filtro de convolución en conjunto con los parámetros fundamentales y a la derecha el resultado del filtro de convolución de la imagen entrante con las operaciones de obtención [34].

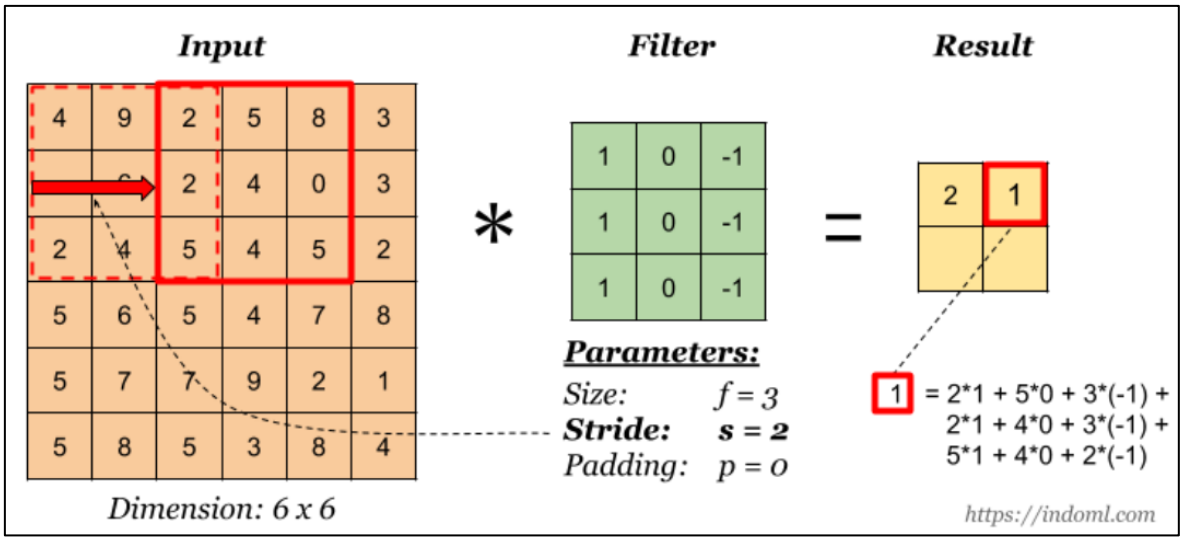


Figura6. Ejemplo de operación de filtro de convolución [34].

La convolución es la operación fundamental que se va a utilizar; la información que contiene un área de pixeles se converge en un único valor, que se obtiene luego de la operación lineal al pasar por el filtro los valores obtenidos en esa área de pixeles. Esta operación se repite en base al número de paso que se tenga en ese momento, y su operación complementaria viene a ser el pooling o de su traducción la reducción, de la cual se aplica un filtro diferente al de las áreas de pixeles, comúnmente a dimensiones diferentes para reducir el número de las dimensiones presentes en toda la información. En la figura 7 se ejemplifica una red en función de las principales operaciones fundamentales [33].

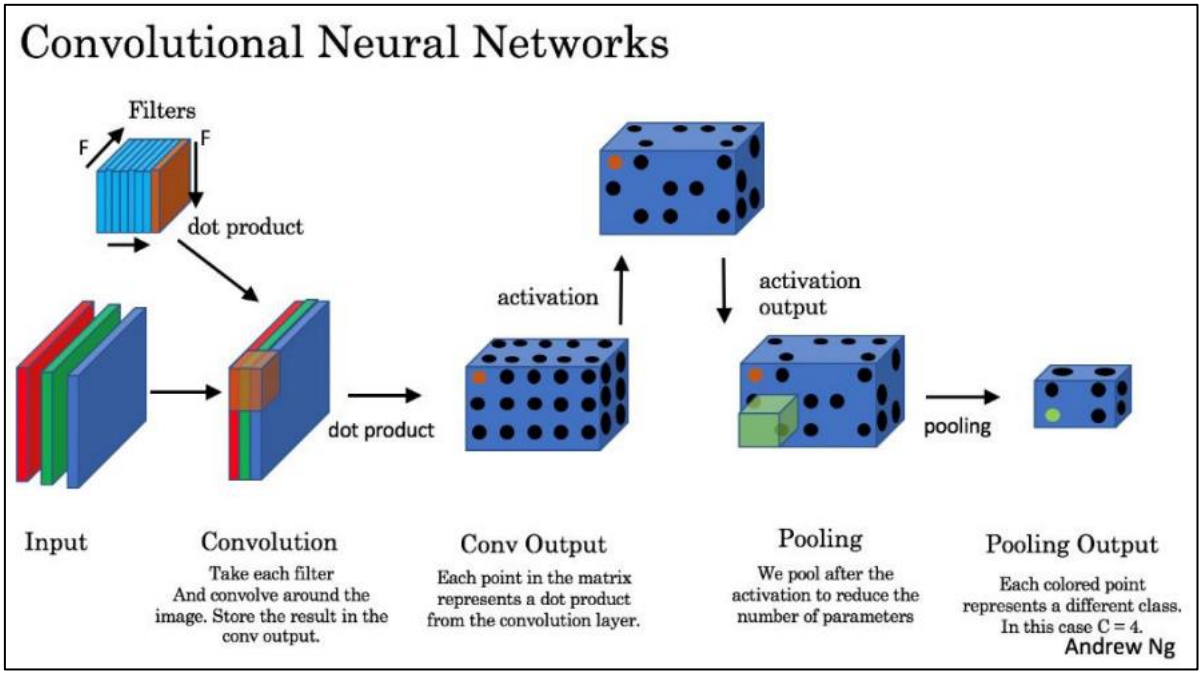


Figura 7. Ejemplo de Red Neuronal Convolucional a nivel de operación fundamental [33].

Para finalizar con este apartado referente a redes neuronales convolucionales profundas, cabe destacar las diferentes operaciones típicas y capas que forman parte del resto de la red y su estructura. Funciones de activación y Softmax vienen a ser las principales, dado que la operación de activación determina si un pixel posee las características mínimas de la información de partida; en cuanto a sus funciones se pueden utilizar la sigmoide, tanH o la más utilizada la ReLu, como se detalla a continuación.

Sigmoide: La función sigmoide se conoce como la función logística que ayuda a normalizar la salida de cualquier entrada en el rango entre 0 y 1. El objetivo principal de esta función de activación es mantener la salida o el valor predicho en el rango particular, lo que hace que la eficiencia del modelo sea buena [36].

TanH: La función de activación Tangente Hiperbólica es superior a la función de activación sigmoide porque el rango de esta función de activación es mayor que la función de activación sigmoide. Esta es la principal diferencia entre la función de activación Sigmoide y Tanh [36].

ReLu: La función ReLu es la mejor y más avanzada función de activación en estos momentos en comparativa con la sigmoide y TanH ya que todos los inconvenientes como el problema del gradiente de fuga se eliminan por completo en esta función de activación, lo que hace que esta función sea más avanzada en comparación con otras funciones [36].

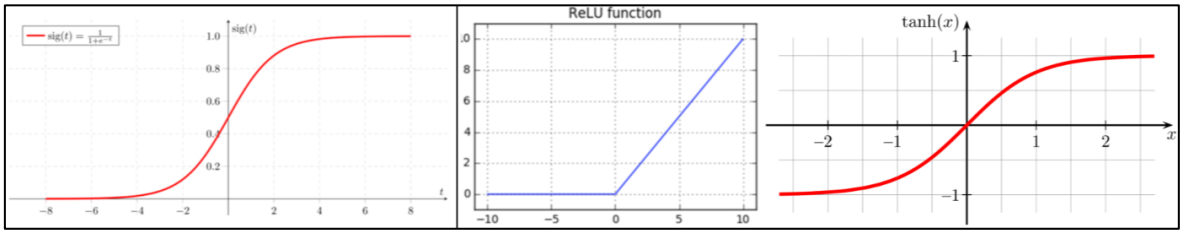


Figura 8. Funciones de activación Sigmoide(izquierda), ReLu(centro), y TanH(derecha) [36].

La Softmax hace referencia a una capa totalmente conectada la cual presenta números de neuronas finales igual al número de clases totales dentro de la clasificación, siendo que sus valores sean enteros. Esta última se menciona porque es un pilar en los problemas de clasificación. En la figura 9 se observa un ejemplo de convolución de una red CNN, empezando con el segundo bloque desde la izquierda, se activa con una función de activación ReLu y se realiza la reducción y culmina con las capas de clasificación de los resultados.

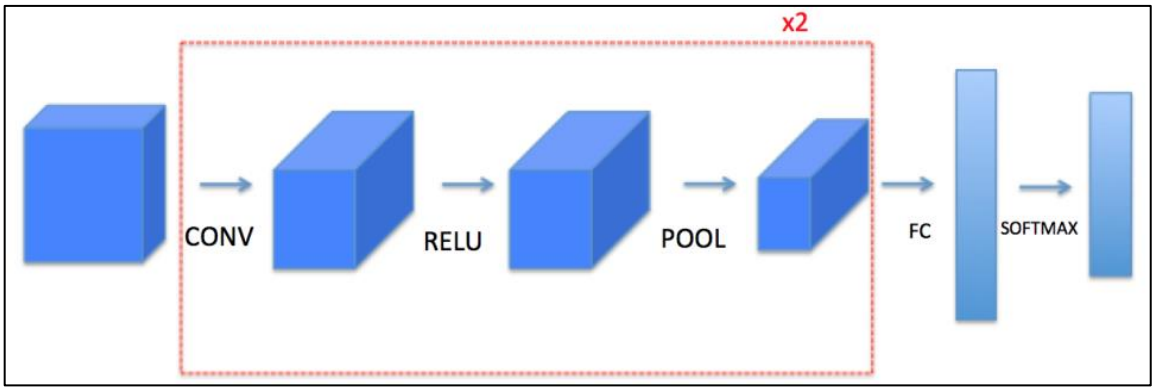


Figura 9. Ejemplo a nivel de proceso de una red neuronal convolucional [35].

1.4.6 Segmentación Semántica

De acuerdo a las tendencias actuales en el campo de la investigación, los modelos obtenidos a partir de las redes neuronales convolucionales para segmentación semántica se han diversificado en varias categorías, por la cual dentro de este trabajo integrador se hará referencia a la segmentación semántica basada en la región. Este método sigue el principio de “segmentación que usa reconocimiento”, en el cual se determina y se extraen regiones libres de la imagen y luego se seleccionan conjuntos en función de la clasificatoria de las regiones propuestas.

Este tipo de redes combinan dos procesos:

1. Computo de regiones propuestas en el cual se albergan una gran cantidad de características fundamentales de un gran número de regiones.
2. Clasificación de identidades en cada segmentación del proceso.

Un ejemplo es Mask-RCNN, que sigue los pasos descritos como se muestra en la figura 10, en el cual se comienza por la detección a nivel ROI (región de interés), se hace una clasificación de las regiones, se asigna una clase a la región de interés y finalmente se acaba segmentando cada pixel de la región de interés con capas convolucionales.

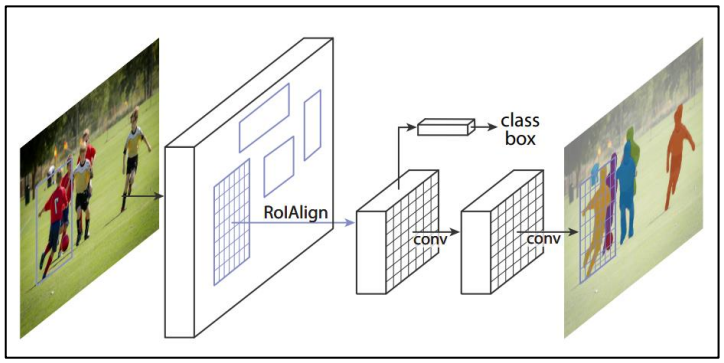


Figura 10. Marco de referencia de Mask R-CNN para la segmentación de instancias [3].

Una implementación típica se basa en técnicas de segmentación de imágenes impulsadas por la red neuronal convolucional (CNN) previamente descrita, que básicamente implica dibujar límites a nivel de píxel en los objetos de una imagen, concretamente en personas.

Hay dos técnicas poderosas, pero distintas, en la segmentación de imágenes que ayudan a los proyectos de visión por computadora:

Segmentación semántica: esto implica detectar objetos dentro de una imagen y agruparlos según categorías definidas. Por ejemplo, en una escena callejera, dibujaría límites y etiquetaría elementos como: personas, automóviles, bicicletas, semáforos, pasarelas, cruces, carriles, etc. [37]

Segmentación de instancias: lleva la segmentación semántica un paso más allá e implica la detección de objetos dentro de categorías definidas. Por ejemplo, en la misma escena de la calle, dibujaría límites individualmente para cada categoría y los etiquetaría de forma única: humanos (adulto, niño), automóviles (automóviles, autobuses, motocicletas) [37].

Si bien el etiquetado de segmentación de instancias es costoso, es uno de los métodos más sólidos y completos para lograr la detección de objetos en el análisis de imágenes. Además, la aplicación amplifica la evaluación de videos mediante el análisis de cuadros individuales dentro del mismo video. Se puede aprovechar la inteligencia real una vez que identifique de forma única cada instancia de objetos en una imagen que está segmentada por categorías definidas más concretamente en un grupo aglomerado de personas para este trabajo, por tal razón se ayuda para el etiquetado del conjunto de imágenes para el entrenamiento y testeo del modelo la herramienta MakeSenseAI.

1.4.7 Herramientas y librerías de desarrollo

1.4.7.1 Python

Es un lenguaje de programación interpretado, orientado a objetos y de alto nivel. Es una de los lenguajes más atractivos por su simplicidad y sus estructuras de datos de alto nivel. Dentro de sus múltiples objetivos este tipo de lenguaje es escogido para el desarrollo de modelos y prototipos dentro del ámbito del aprendizaje automático [46].

1.4.7.2 Google Colab

Esta herramienta es un producto de Google Research, Colaboratory o “Colab” en su abreviación, permite a un usuario cualquiera codificar y ejecutar cualquier código de Python en el navegador. Es adecuado especialmente para tareas de aprendizaje automático, educación y análisis de datos. Sus recursos informáticos no tienen coste adicional [47].

1.4.7.3 Miniconda Individual Edition

Miniconda es un instalador mínimo gratuito para conda. Es una pequeña versión de arranque de Anaconda que incluye solo conda, Python, los paquetes de los que dependen y una pequeña cantidad de otros paquetes útiles, incluidos pip, zlib entre otros. [48].

1.4.7.4 JSON

JSON (Notación de objetos de JavaScript) es un intercambio de datos con formato ligero. Se basa en JavaScript ECMA-262 3ra edición – diciembre 1999, como subconjunto. JSON es completamente independiente del lenguaje, pero utiliza convenciones que son familiares para los programadores de la familia de lenguajes Java, JavaScript, C, Perl, Python, entre otros. Estas propiedades lo hacen un lenguaje de intercambio de datos ideal [49].

1.4.7.5 MakeSenseAI

La herramienta en línea makesense.ai es de uso gratuito para etiquetar fotos. Gracias al uso de un navegador, no requiere ninguna instalación complicada: simplemente se visita el sitio web y estará listo para comenzar. Tampoco importa en qué sistema operativo se esté ejecutando: hacen todo lo posible para ser verdaderamente multiplataforma. Es perfecto para pequeños proyectos de aprendizaje profundo de visión por computadora, lo que hace que el proceso de preparación de un conjunto de datos sea mucho más fácil y rápido. Las etiquetas preparadas se pueden descargar en uno de los múltiples formatos admitidos. La aplicación fue escrita en TypeScript y está basada en React.

makesense.ai se esfuerza por reducir significativamente el tiempo que se tiene que dedicar a etiquetar fotos. Para lograr esto, se utilizan muchos modelos diferentes de IA que podrán brindar recomendaciones y automatizar actividades repetitivas y tediosas [32].

1.4.7.6 OIDv4 Toolkit – herramienta OIDv4

Esta herramienta permitirá obtener de una manera mucho más rápida el conjunto de datos (imágenes) necesarias para poder encontrar el camino correcto hacia una buena calidad y construcción del modelo, en tanto a su entrenamiento y testeo. Es una herramienta basada en python3 para tareas de clasificación y detección de objetos, las cuales pueden ser:

Detección de objetos:

- Descargar cualquiera de las 600 clases del conjunto de datos individualmente.
- Descargar múltiples clases al mismo tiempo creando carpetas separadas.

- Descargar múltiples clases y creando una carpeta común para todos de ellos con un archivo de anotación único de cada imagen.
- Descargar una sola clase o varias clases con los atributos deseados.
- Usar el práctico visualizador para inspeccionar las clases descargadas.

Clasificación de imágenes

- Descargar cualquiera de las 19,794 clases en una carpeta etiquetada común.
- Aprovechar decenas de posibles comandos para seleccionar solo las imágenes deseadas (por ejemplo, como solo imágenes de prueba).

El código está bastante documentado y diseñado para que sea fácil de ampliar y mejorar, el cual está basado en el conjunto de datos de imágenes abiertas de Google [6].

1.4.8 Métricas de Rendimiento y Precisión

Todo lo mencionado hasta el momento se va a estudiar y comparar. Por tal razón se enlista una seria de métricas que serán de mucha utilidad a la hora de poder determinar si el modelo o sistema esta con un comportamiento correcto acorde a lo establecido en el alcance de este trabajo integrador. De esta manera se incluyen las siguientes métricas [38].

1.4.8.1 mAP (mean Average Precission)

El valor de la precisión a la cual un modelo está ligado, es el mAP (mean Average Precission) la cual es la medida con la que mide el rendimiento de los modelos de machine learning en detección de objetos.

1.4.8.2 IoU (Intersection over Union)

Esta métrica de evaluación, intersección sobre unión, es utilizada para medir la precisión a la cual está ligada a un detector de objetos en un conjunto de datos en particular. El modelo que proporcione cuadros delimitadores en base a predicciones como salida puede ser evaluado con esta métrica IoU.

1.4.8.3 mean IoU (mean Intersection over Union)

La intersección media sobre unión (mean_iou) es una métrica de evaluación común para la segmentación semántica de imágenes, que primero se calcula para cada clase semántica y luego calcula el promedio sobre las clases totales.

La figura 11 muestra un ejemplo que hace uso de los cuadros delimitadores predichos [38].

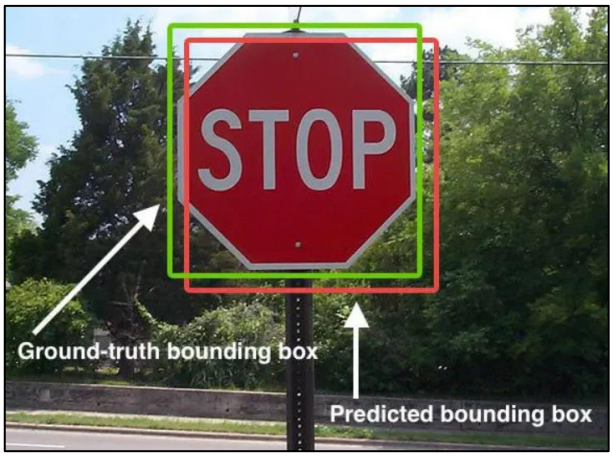


Figura 11. Ejemplo de detección de objeto.

1.4.8.4 Precisión y Recall

La precisión y el recall son dos métricas de evaluación de modelos con relevancia de factor importante. Mientras que la precisión se atribuye al resultado en base a un porcentaje que es relevante, la recuperación en cambio se detalla como el porcentaje de los resultados relevantes clasificados idónea y correctamente por el algoritmo [51]. La precisión y el recall viene dada por la ecuación 1 y la ecuación 2 respectivamente.

$$P = \frac{Verdaderos\ Positivos}{Resultados\ Actuales}$$

Ecuación 1. Precisión

$$R = \frac{Verdaderos\ Positivos}{Resultados\ Predichos}$$

Ecuación 2. Recall

De la ecuación 1 se puede decir que los verdaderos positivos al igual que en la ecuación 2 son los valores obtenidos por el algoritmo en cuanto son totalmente verídicos y sujetos a análisis, y que por su lado los resultados actuales de la ecuación 1 vienen dado por aquellos resultados totales reales existentes en el conjunto de prueba a diferencia de los resultados predichos en la ecuación 2, en donde refiere a los resultados obtenidos de la predicción del modelo que no necesariamente tienen que ser cien por ciento reales.

1.4.8.5 Ground Truth

El Ground Truth en el aprendizaje automático se refiere a la realidad que desea modelar con el algoritmo de aprendizaje automático supervisado implementado. Ground Truth también se conoce como el objetivo para entrenar o validar el modelo con un conjunto de datos etiquetado. Durante la inferencia, un modelo de clasificación predice una etiqueta, que se puede comparar con la etiqueta de verdad básica (Ground Truth), si la misma está disponible para su comparación [52].

2 METODOLOGÍA

Este capítulo alberga todas las actividades que se realizan para el desarrollo del modelo de redes neuronales convolucionales con técnicas de segmentación semántica propuesto en base a la teoría relacionada y expresada en el capítulo anterior. El capítulo inicia con una descripción referente a la metodología Action-Research (AC) [40], en conjunto con la planificación de acción en base a la misma metodología y sus respectivas actividades.

El tipo de trabajo relacionado a la investigación se basa en el análisis exploratorio de los datos (imágenes) que sirven de entrada para el modelo de redes neuronales convolucionales, el cual genera la validación y la calidad de los datos obtenidos, juntamente con el detalle de como se selecciona, limpia y se construyen los datos (imágenes) que sirven para el óptimo proceso de entrenamiento y testeo del modelo implementado.

Las técnicas de recolección y tratamiento de la información se basan en las ya mencionadas herramientas OIDv4 y MakeSenseAI, describiendo los procesos de cómo se utilizan cada una de ellas para cumplir con los objetivos de este trabajo integrador.

Las técnicas de análisis de información por su parte están basadas en:

- Revisión y reducción de los datos (imágenes).
- Análisis de contenido.
- Obtención y verificación de resultados y conclusiones.

Finalmente se detalla una serie de actividades para cumplir con los objetivos planteados en este documento, las cuales son:

1. Análisis exploratorio y validación de la calidad de los datos.
2. Selección, limpieza y construcción de datos.
3. Creación del modelo de redes neuronales convolucionales, basado en un modelo previamente entrenado.
4. Incorporación del algoritmo de distanciamiento.
5. Pruebas y medición de resultados del modelo entorno a las limitaciones mencionadas.

La última de estas actividades se discutirá en la sección 3 de este documento.

2.1 Metodología Action Research

AC es conocida de muchas maneras dentro del ámbito investigativo, es así que toma muchas definiciones pero que todas detallan el mismo objetivo, el cual es “aprender haciendo”, en pocas palabras se define un problema, se analiza y se otorga una solución al problema, ver resultados y definir que cuan satisfactorio fueron dichos resultados y si no cumple con lo esperado se procede a repetir el ciclo; de manera más visual se ilustra dicho ejemplo en la figura 12.

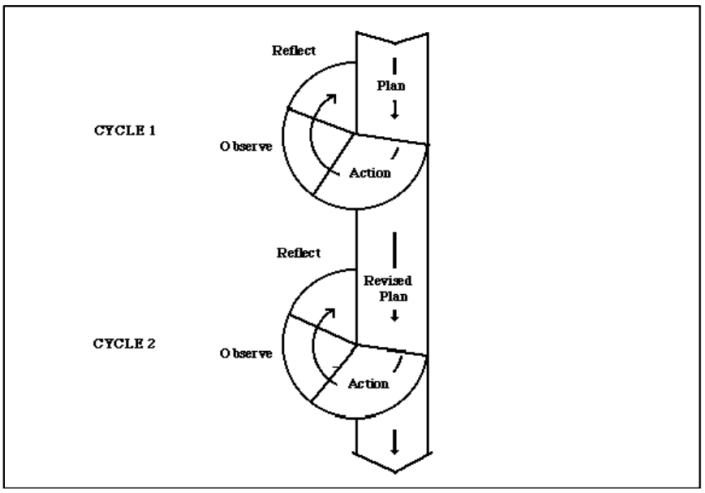


Figura 12. Modelo básico de AC.

De esta manera se puede mencionar que dicha metodología se enfoca y utiliza situaciones reales y da soluciones a problemas reales, en diferencia con otras metodologías que son netamente experimentales y con fines investigativos.

Basándose en el modelo de Stringer [41], señala que a medida que las personas interesadas implementen un estudio, el mismo debe tener unos procesos que deben ser detallados en base a lo siguiente:

- Diseñar el estudio: se refina esmeradamente el tema de investigación, teniendo en cuenta la planificación sistemática investigativa, evidenciando la ética y la eficacia del trabajo en proceso de realización.
- Recopilar datos: Incluir todos los datos necesarios de las diversas fuentes verídicas y comprobables que serán utilizadas para el fin específico con su respectivo fundamento científico
- Analizar datos para identificar patrones y características claves, con la finalidad de obtener una síntesis de los datos y desencadenar en conclusiones que aporten al trabajo final.

- Usar resultados en base al estudio planteado y obtener la resolución del problema propuesto a investigar.
- Comunicar los resultados de la investigación al público objetivo y pertinente.

Con esta información se utiliza el marco Ver > Pensar > Actuar, el cual es propuesto por Stringer [40], La siguiente tabla indica en que consiste este marco de referencia, en base a cada una de las fases que corresponden con las prácticas tradicionales de investigación tradicionales, las cuales son utilizadas a lo largo de la implementación de este proyecto integrador.

TABLA 1. Rutina elemental de Action-Research

VER	PENSAR	ACTUAR
- Recopilar Datos **- Definir y describir la situación.**	- Explorar y analizar - Interpretar y explicar	- Plan (Informe) - Implementar - Evaluar

2.2 Análisis exploratorio y validación de la calidad de los datos.

2.2.1 Conjunto de datos: Common Objects in Context (COCO)

COCO es un conjunto de datos de imágenes de gran tamaño diseñado para la detección de objetos, la segmentación, la detección de puntos clave de personas, la segmentación de cosas y la generación de subtítulos [7].

En la figura 13 se observa como este conjunto de datos viene anotado, para tal efecto el mismo se apoya en la herramienta FiftyOne [42] para facilitar su visualización y acceso a todos los datos que COCO puede otorgar en cuanto a análisis y evaluación de modelos basados en este conjunto de datos.

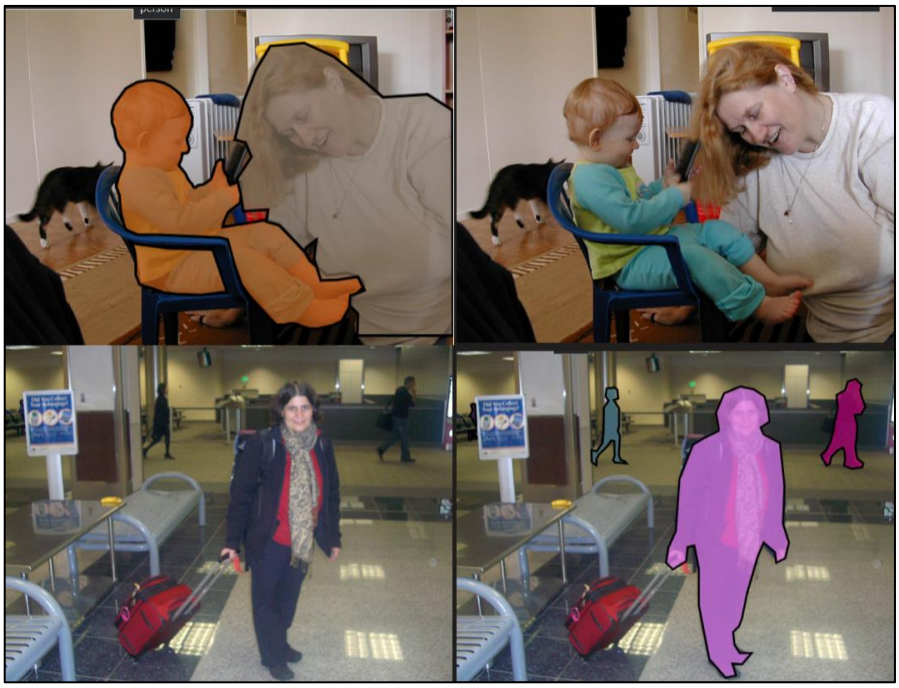


Figura 13. Muestras de imágenes anotadas en el conjunto de datos MS COCO.

El conjunto de datos de COCO sirve como soporte de entrenamiento al modelo Mask R-CNN que se detalló previamente, el cual se demoró una gran cantidad de tiempo en costos de procesamiento de GPU para obtener resultados como se observa en la figura 14.

Figura 14. Ejemplo de implementación del modelo Mask R-CNN [43].

Cabe recalcar que el contenido de la figura 14 fue realizada siguiendo los principios de [3] pero que se distingue su implementación en ciertos detalles como la tasa de aprendizaje, el redimensionamiento de las imágenes, entre otras, como se establece más a fondo y con mayor detalle en [43].

2.2.2 Conjunto de datos: Open Images Dataset V4 (OIDv4)

Open Images es un conjunto de datos de aproximadamente 9 millones de imágenes que se han anotado con etiquetas a nivel de imagen, cuadros delimitadores de objetos y relaciones visuales. El conjunto de entrenamiento de V4 contiene 14,6 millones de cuadros delimitadores para 600 clases de objetos en 1,74 millones de imágenes. Los cuadros han sido dibujados en gran parte manualmente por anotadores profesionales para garantizar la precisión y la coherencia. Las imágenes son muy diversas y suelen contener escenas complejas con varios objetos (8,4 por imagen de media). Esto también fomenta las anotaciones de imágenes estructurales, como las relaciones visuales. Además, el conjunto de datos se anota con etiquetas a nivel de imagen que abarcan miles de clases [6].

2.2.2.1 Organización de los datos

El conjunto de datos se separa en tres grupos; un conjunto de entrenamiento, uno de validación y otro de testeo, como se menciona en [6] las imágenes están anotados en base a tres pilares que son los siguientes:

Etiquetas a nivel de imagen

La Tabla 2 muestra una descripción general de las etiquetas de nivel de imagen en todas las divisiones del conjunto de datos. Todas las imágenes tienen etiquetas a nivel de imagen generadas automáticamente por un modelo de visión artificial similar a la API de Google Cloud Vision. Estas etiquetas generadas automáticamente tienen una tasa sustancial de falsos positivos.

TABLA 2. Etiquetas a nivel de imagen

	Entrenamiento	**Validación**	**Testeo**	**Número de clases**	**Número de clases entrenables**
Imágenes	9 011 219	41 620	125 436	-	-
Etiquetas generadas por computadora	78 977 695	512 093	1 545 835	7 870	4 764
Etiquetas verificadas por humanos	27 894 289 pos: 13 444 569 neg: 14 449 720	551 390 pos: 365 772 neg: 185 618	1 667 399 pos: 1 105 052 neg: 562 347	19 794	7 186

Además, los conjuntos de validación y prueba, así como parte del conjunto de entrenamiento, tienen etiquetas a nivel de imagen verificadas por humanos. La mayoría de las verificaciones se realizaron con anotadores internos en Google como se detalla en [6].

Cuadros delimitadores

La Tabla 3 muestra una descripción general de las anotaciones del cuadro delimitador en todas las divisiones del conjunto de datos, que abarcan 600 clases de objetos. Estos ofrecen una gama más amplia que los de los desafíos de detección de ILSVRC y COCO [6].

TABLA 3. Cuadros

	Entrenamiento	**Validación**	**Testeo**	**Número de clases**
Imágenes	1 743 042	41 620	125 436	-
Cuadros	14 610 229	204 621	325 282	600

Relaciones visuales

La Tabla 4 muestra una descripción general de las anotaciones de relaciones visuales en la división del tren del conjunto de datos.

TABLA 4. Relaciones

	Entrenamiento	**Validación**	**Testeo**	**Tripletas de relaciones distintas**	**Número de clases**	**Número de atributos**
Relación de tripletas	374 768	3 983	12 248	329 Obj-obj: 287 Atributo: 42	57	5

2.2.3 Calidad y validación de los datos

Al contar con dos conjuntos de imágenes ampliamente de código abierto y de gran calidad, en cuanto a diversas técnicas de detección, anotación, segmentación y segmentación de instancias, se determina que cualquier imagen proveniente de los mismos será de una alta calidad, exceptuando casos en donde la descarga de las imágenes requiera un tratamiento manual el cual se determinará en próximas secciones, así también, como su validación entorno a la calidad del etiquetado de datos.

2.3 Selección, limpieza y construcción de los datos

2.3.1 Selección de los datos mediante OIDv4

Mediante la herramienta OIDv4 se hace el uso de los scripts dentro del repositorio, para poder descargar las imágenes necesarias, las cuales serán tratadas con el software MakeSenseAI para la anotación de las máscaras que sirven para la técnica de segmentación semántica como dicta en [32], mediante la ayuda proporcionada en [44] se procede a descargar las imágenes deseadas con el siguiente comando:

python3 main.py downloader --classes 'Person' --type_csv train –limit 1500 [44].

Esta operación da como resultado el siguiente árbol de directorios, como se observa en la figura 15, en base a 1500 imágenes objetivo junto a su archivo anclado.

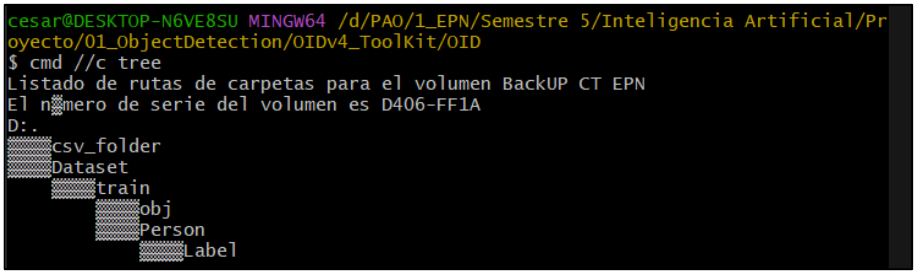


Figura 15. Árbol de directorios descargados de OIDv4.

Una vez descargadas las 1500 imágenes se determina como se componen cada uno de los archivos descargados por la herramienta, por la cual dentro de la carpeta "csv_folder" se encuentra alojados los archivos csv que arrojan la información de los cuadros delimitadores existentes en cada una de las imágenes, como indica la figura 16, esto con la finalidad de recabar información para la ejecución del algoritmo de distanciamiento a nivel de pixeles que se detallará en futuras secciones.

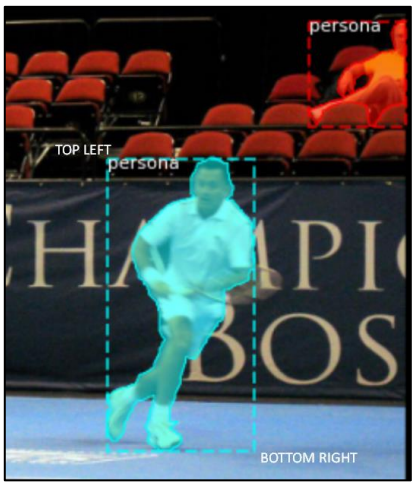


Figura 16. Ejemplo de Cuadro delimitador en personas.

Cada cuadro delimitador tiene los siguientes atributos en base a su ROI:

- Contiene el punto superior izquierdo (x.y) de la región de interés.
- Contiene tanto anchura como holgura (h,w) que delimitará la zona del objeto.

2.3.2 Limpieza de los datos

Ya que la limitación de la descarga de las imágenes mediante la herramienta OIDv4 es que la misma no contiene un script que limpie datos que sean irrelevantes para el objetivo del proyecto, se procede a eliminar un total de 1100 imágenes de forma manual que no atribuyen calidad y que por tiempo se reducen significativamente, es decir, del total de 1500 únicamente se albergan 400 imágenes que cumplen con las características de las personas en lugares transitados, así como imágenes de pocas personas, al igual que imágenes con una gran variedad de personas.

Algunos de los ejemplos de las personas contenidas en las imágenes a razón de estudio se observan en la figura 17, las cuales ya sirven como punto de partida para proceder con la anotación de cada una de ellas con la herramienta makeSenseAI, como se ilustra en la siguiente sección.

Figura 17. Ejemplos del conjunto de datos propuesto.

2.3.3 Construcción de las anotaciones de los datos mediante MakeSenseAI

Como se evidencia en [45], esta herramienta permite continuar con las anotaciones de cada una de las imágenes a nivel de pixeles, es decir, ir encapsulando a los objetos de interés, en este caso personas, como se observa en la figura 18.

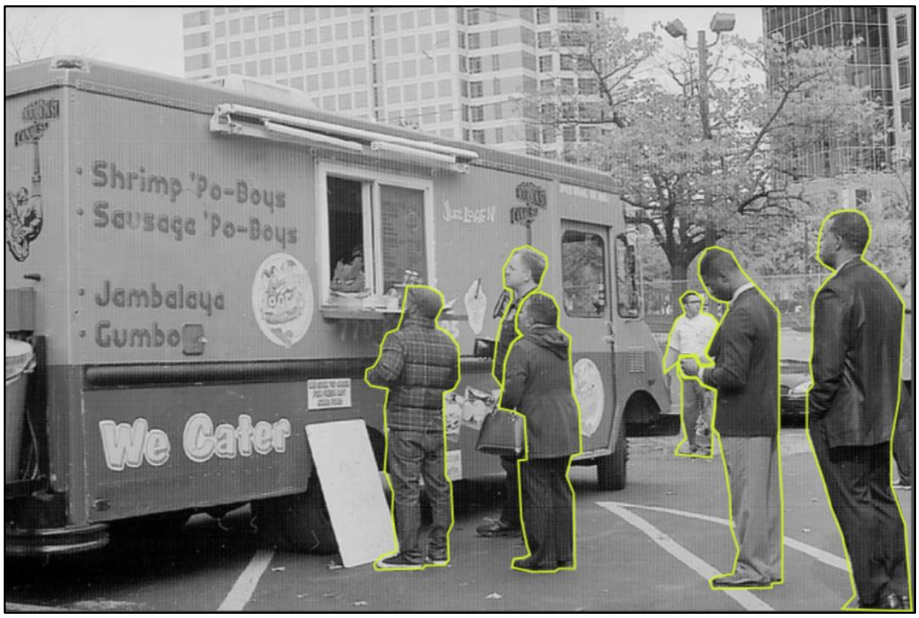


Figura18. Polígono delimitador con la ayuda de MakeSenseAI.

De tal manera que al final de todas las anotaciones de todas las imágenes existentes en el conjunto de datos, se debe extraer su archivo JSON en formato COCO el cual cuenta con los siguientes elementos:

- **Images**
  - **id**: identificador de la imagen.
  - **width**: ancho de la imagen (pixeles).
  - **height**: altura de la imagen (pixeles).
  - **filename**: nombre del archivo.
- **annotations**
  - **id:** identificador de la anotación.
  - **iscrowd**: si existe multitud 1, si no existe multitud 0.
  - **image_id**: identificador de la imagen relacionada.
  - **category_id**: identificador de la categoría (al solo existir una todas llevan el valor de 1).
  - **segmentation**: tiene todas las coordenadas de los pixeles del polígono delimitador.

- **bbox**: contiene todas las coordenadas del cuadro delimitador de cada segmentación.
- **área**: contiene el valor del área delimitada por el polígono.

- **Categories**
  - **id**: identificador de la categoría asociada.
  - **name**: nombre de las categorías existentes.

Un ejemplo de anotación más visual se ilustra en la figura 19. En donde se tiene todos los elementos ya mencionados de una imagen en particular.


```
{"info":
{"description":"my-project-name"},
"images":[{"id":1,
            "width":1024,
            "height":685,
            "file_name":"0423423f59cf4762.jpg"},
"anotations":[{"id":0,
                "iscrowd":0,
                "image_id":1,
                "category_id":1,
                "segmentation":[964.2534886596851,590.2657421095447,80.251572
                "bbox":[964.2534886596851,590.2657421095447,80.25157232704385
                "area":6416.677174882353}]
"categories":[{"id":1,"name":"persona"}]}
```


Figura 19. Ejemplo de anotación en archivo JSON.

Con toda la información adoptada mediante la herramienta es posible hacer paso a la validación de este archivo mediante el formato proveniente de COCO [50].

2.3.3.1 Validación de la construcción de las anotaciones

Como se menciona en [50], es importante validar las anotaciones realizadas en el apartado anterior, para esto se tiene en cuenta el formato de COCO, el cual es una estructura JSON específica que dicta cómo se guardan las etiquetas y los metadatos para un conjunto de datos de imagen. Con esto se asegura que la exportación de las anotaciones estén en dicho formato ya que el modelo pre-entrenado Mask R-CNN que se detallará con mayor profundidad en futuras secciones utiliza este tipo de formatos de etiquetas, de esta manera dentro del software makeSenseAI se descarga todas las anotaciones realizadas en un único archivo en formato COCO como indica la figura 20, en donde también cabe destacar que la misma herramienta cuenta con algoritmos de aprendizaje automático para facilitar la detección de objetos, más no la de segmentación, por lo que esta parte de la implementación se la realiza de forma manual para cada imagen dentro del conjunto de datos.

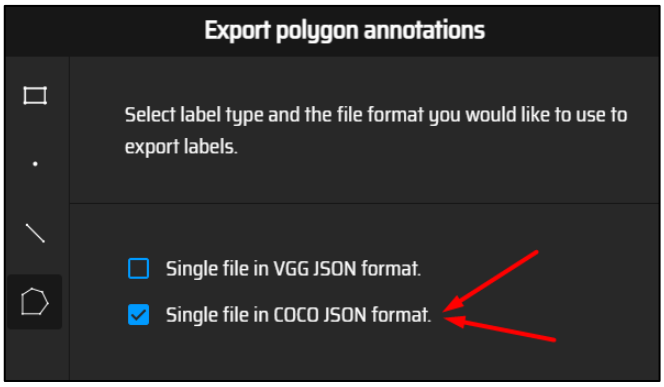


Figura 20. Exportación de anotaciones en formato COCO.

De esta manera se puede validar con la ayuda de scripts de Python para verificar que las anotaciones de las máscaras de los objetos delimitados por el polígono sean correctas, para mayor detalle de los scripts véase anexo V, en la figura 21 se observa algunas muestras de las imágenes dentro del conjunto de datos que se va a utilizar para entrenar el modelo, juntamente a sus respectivas máscaras de la clase persona.

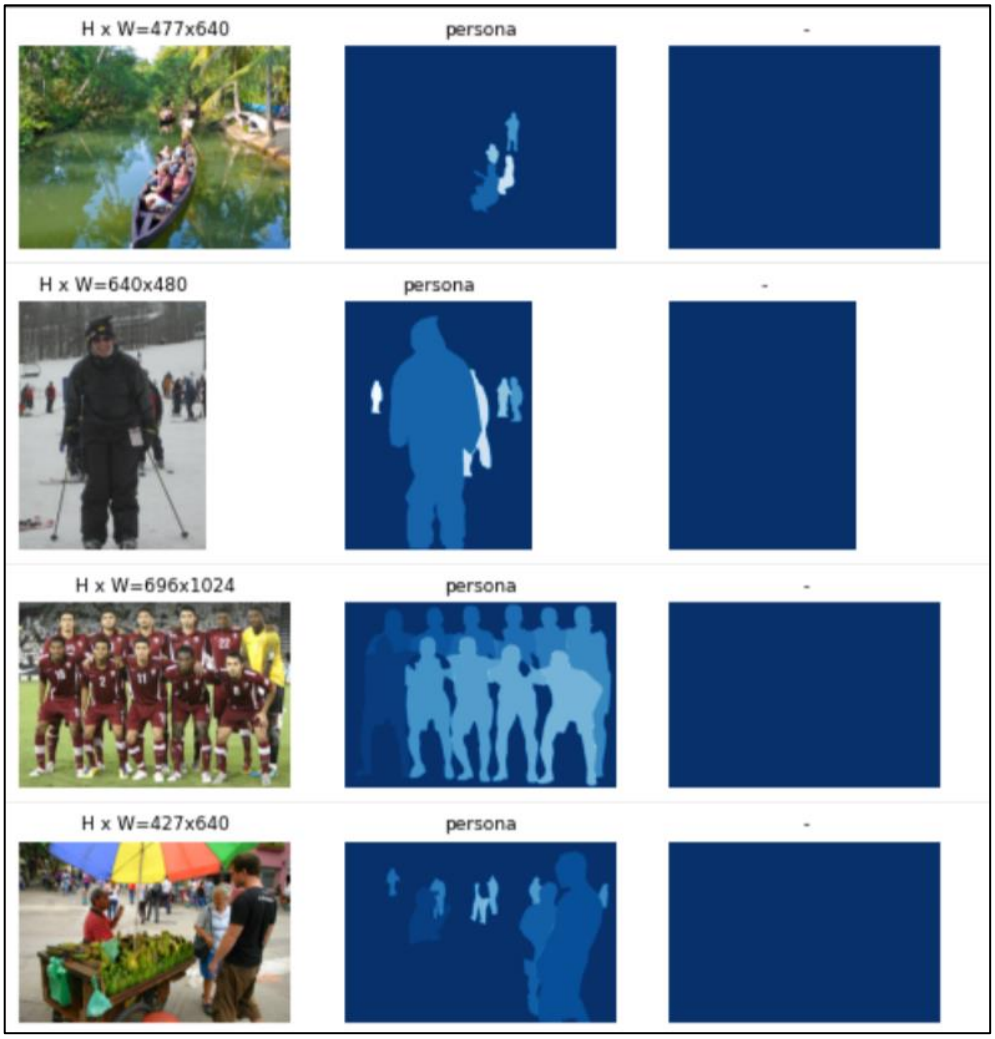


Figura 21. Ejemplos de máscaras del conjunto de datos.

2.3.3.2 Factor tiempo en la construcción de las anotaciones

Como el estado del arte lo dicta tanto en [3] y [6], existen modelos de aprendizaje automático que ayudan a las anotaciones, sin embargo también se necesita de la verificación humana para obtener una calidad óptima en todas las anotaciones referente a detección de objetos y segmentación semántica se refiere; es por eso, que el tiempo cumple un factor importante ya que a mayor número de imágenes, y mayor número de objetos identificados en dicha imagen se tenga, el tiempo que llevará realizar el polígono delimitador por cada uno de los objetos en cada una de las imágenes será mucho más demandante y prolongado.

Por tal razón se ha limitado a un número de 200 imágenes en total para que el mismo sea el conjunto de datos que será incorporado dentro del entrenamiento y testeo del modelo a realizarse, en específico 151 imágenes para entrenamiento, 18 para validación y el restante para testeo, teniendo en cuenta que la única clase que se tendrá dentro de los archivos de configuración del modelo será una, la cual es designada a la clase persona y su background por defecto, con mayor detalle se analizará lo dicho en la siguiente sección.

2.4 Creación del modelo de redes neuronales convolucionales basado en un modelo previamente entrenado

Para la creación del modelo, se procede a realizar una breve comparativa de los modelos existentes referentes a detección de objetos y segmentación semántica se refiere, la tabla 5 muestra información detallada sobre los modelos basados en redes neuronales convolucionales escogidos para la comparación.

Esta comparativa es de vital importancia ya que se tiene un mejor entendimiento de cómo y por qué se elige el modelo pre entrenado que sirve de soporte para la creación del nuevo modelo que detecte y segmente aglomeración de personas provenientes del conjunto de imágenes existentes en un video (frames), junto a esto se analiza la posibilidad de la transferencia de aprendizaje en base a que tan profundo se debe entrenar la red seleccionada Mask R-CNN, como también determinando el número de épocas, número de pasos de validación, tasas de aprendizajes y demás configuraciones que serán detalladas en esta sección con la finalidad de obtener un modelo que cumpla con los objetivos y alcance de este proyecto integrador.

TABLA 5. Comparativa entre Modelos de Machine Learning

TÉCNICA/ FRAMEWORK	MODELO	ACCURACY (mAP)	FUNCIONAMIENTO
DETECCIÓN DE OBJETOS			
CAFFE [57]	SSD300	87.09% En dataset VOC2007	El modelo es la implementación del marco Caffe del algoritmo Single-Shot multibox Detection (SSD) con resolución de entrada de 300x300 y backbone. La red destinada a realizar la detección visual de objetos. Este modelo está preentrenado en el conjunto de datos VOC2007 + VOC2012 + COCO y es capaz de detectar 20 clases de objetos PASCAL VOC2007: ssd300VGG-16
CAFFE [57]	SSD512	91.07% En dataset VOC2007	El modelo es la implementación del marco Caffe del algoritmo Single-Shot multibox Detection (SSD) con resolución de entrada y backbone de 512x512. La red destinada a realizar la detección visual de objetos. Este modelo está preentrenado en el conjunto de datos VOC2007 + VOC2012 + COCO y es capaz de detectar 20 clases de objetos PASCAL VOC2007:ssd512VGG-16
KERAS [58]	YOLOv3-tf	67.7%	YOLO v3 es un modelo de detección de objetos en tiempo real implementado con Keras convertido al marco TensorFlow. Este modelo fue entrenado previamente en el conjunto de datos Common Objects in Context (COCO) con 80 clases.
DARKNET [59]	YOLOv3	87.54%	Se aplica una única red neuronal a toda la imagen. Esta red divide la imagen en regiones y predice cuadros delimitadores y probabilidades para cada región. Estos cuadros delimitadores se calculan de acuerdo con las probabilidades previstas. Considera la imagen completa en el momento de la inspección para que sus predicciones estén informadas por el contexto general de la imagen. También hace predicciones con una calificación de red única a diferencia de sistemas como R-CNN que requieren miles para una sola imagen [59].
Autoría propia César Taco [63]	CT	77%	En base al modelo de yolov3 se customiza tanto las ultimas capas de la arquitectura con la edición del archivo .cfg, obtener los archivos obj.data y obj.names que hacen referencia al label de cada caja para el entrenamiento de toda la data personalizada y etiquetada con la ayuda del toolkit OIDv4 (Open Image Dataset versión 4) y como ultimo el archivo train.txt el cual ayuda al entrenamiento de los nuevos datos en base al modelo ya preentrenado. Para el entrenamiento se utiliza el framework de darknet como también el archivo de pesos preentrenados (yolo.weights)

TÉCNICA/ FRAMEWORK	MODELO	ACCURACY (mean_iou)	FUNCIONAMIENTO
SEGMENTACIÓN SEMÁNTICA			
Keras_DeepLab_v3 [60]	DeepLab V3	66.08%	Para manejar el problema de segmentar objetos a múltiples escalas, se diseñó módulos que emplean la convolución atrous en cascada o en paralelo para capturar el contexto multiescalar mediante la adopción de múltiples velocidades atrous. Además, proponen aumentar el módulo Atrous Spatial Pyramid Pooling propuesto en papers anteriores, que sondea características convolucionales a múltiples escalas, con características a nivel de imagen que codifican el contexto global y aumentan aún más el rendimiento
MASK RCNN [61]	MASK RCNN	84.44%	En base a pesos pre entrenado para MS COCO se puede obtener un punto de partida para otorgar una propia versión de entrenamiento de la red. El modelo genera cuadros delimitadores y máscaras de segmentación para cada instancia de un objeto en la imagen. Se basa en Feature Pyramid Network (FPN) y una red troncal ResNet101.
YOLACT [62]	RESNET 50	34.1 mAP	El modelo resnet50 de YOLACT++ funciona a 33.5 fps en un Titan Xp y alcanza 34.1 mAP en COCO. Esta técnica utiliza la visualización de los coeficientes de la detección de los objetos para delimitar de donde y hasta donde empieza y termina dicho objeto. Para evaluar el modelo, coloque el archivo de pesos correspondiente en el directorio y ejecute uno de los siguientes comandos. El nombre de cada configuración es todo antes de los números en el nombre del archivo (por ejemplo, ../weightsyolact_baseyolact_base_54_800000.pth
Autoría propia César Taco [63]	CT	86%	En base a pesos pre entrenado para MS COCO se puede obtener un punto de partida para otorgar una propia versión de entrenamiento de la red. El modelo genera cuadros delimitadores y máscaras de segmentación para cada instancia de un objeto en la imagen. Se basa en Feature Pyramid Network (FPN) y una red troncal ResNet101. Gracias a la selección y tratamiento de las imágenes en donde solo contienen personas, y sus respectivas etiquetas se entrena la red en sus primeras capas para que se efectúe una transferencia de aprendizaje del modelo Mask-RCNN.

2.4.1 Familiarización con el modelo pre-entrenado (Mask R-CNN)

Esta red neuronal se usa como punto inicial, por tal motivo, el primer paso es descargarse el modelo desde [43]. En donde se analiza a fondo todos los ficheros contenidos en el repositorio ya que algunos de ellos son modificados y se añade código en donde corresponde para cumplir con los objetivos de este proyecto integrador. Un ejemplo de su implementación se visualiza en la figura 22.

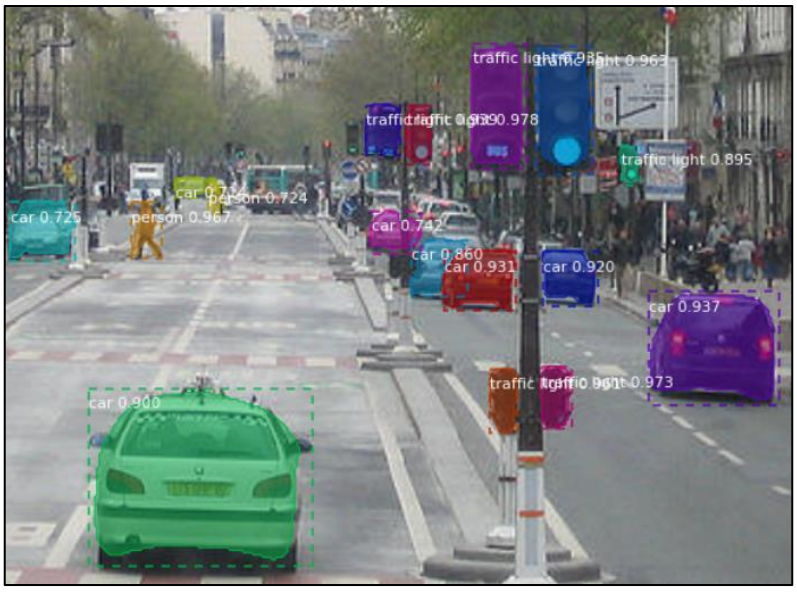


Figura 22. Aplicación de Mask R-CNN sobre una imagen [43].

Tras la validación de algunos ejemplos como la de la figura 22 se observa que, para una imagen dada, el modelo devuelve por cada objeto detectado su cuadro delimitador interlineado, la clase a la que pertenece con su probabilidad de ocurrencia, y la máscara a nivel de pixel. De esto se puede entender los atributos de un objeto detectado que aporta la información devuelta por Mask R-CNN que son:

- **ID**: es un identificador único para cada uno de los objetos detectados que permite procesar el seguimiento en una secuencia de imágenes o video.
- **ID_CLASE**: Indica la clase perteneciente del objeto detecto (persona, carro, moto, …). Esto facilita a un mejor alcance de los resultados.
- **COLOR**: Color expresado en RGB que es asociado a cada ID.
- **PUNTUACIÓN** – **SCORE**: Dictamina en base a la probabilidad de que el objeto detectado sea proveniente de la clase mostrada por el id_clase.
- **BBOX**: Se refiere al cuadro delimitador o del inglés Bounding Box, el cual es una caja que delimita la detección del objeto y que se simboliza por cuatro valores (x1,x2,y1,y2). La esquina la inferior es (x2,y2) y la superior izquierda es (x1,y1).

- **MÁSCARA**: Arreglo de pixeles (matriz) en base a la dimensión de cada frame que se va ajustando alrededor de cada detección de los objetos; en la figura 23 se muestran los atributos/características de un objeto en concreto.

Figura 23. Atributos mediante la implementación de Mask R-CNN sobre un objeto.

Para un mejor entendimiento de los resultados en las siguientes imágenes se omitirán el score y el id de la clase puesto que al tener únicamente personas no es necesaria esta información visual en los resultados. Por otro lado, el bbox únicamente se hará visible en cuanto cumplan las condiciones de distanciamiento que se lo analiza en futuras secciones.

Punto fuerte y punto débil del modelo

Existe un punto fuerte en donde el modelo es altamente confiable, en la figura 24, se puede apreciar que sin la existencia de superposición excesiva de objetos el modelo detecta, y encapsula de una manera idónea.

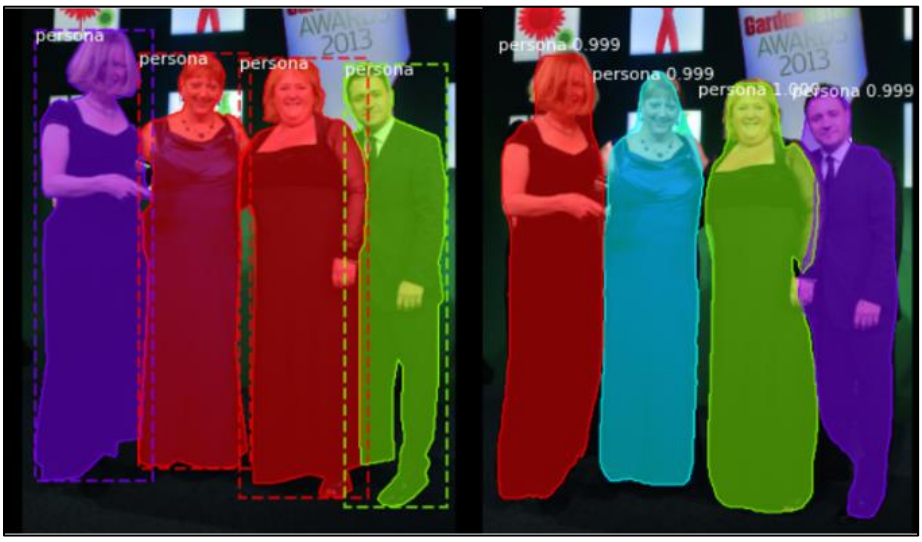


Figura 24. Ejemplo de seguimiento ideal.

Sin embargo, el punto débil es cuando existe una superposición exagerada de objetos, es ahí en donde existe un límite de seguimiento por parte del modelo, como se observa en la figura 25, el comportamiento del modelo sobre una imagen con alta concurrencia de personas es limitado, es por esto que se debe utilizar alguna otra clase de información como el color, la forma o textura de los objetos para llegar a obtener un sistema mucho más robusto y que detecte en su totalidad al 100% de personas existentes en una instancia.

Figura 25. Ejemplo de seguimiento limitado.

2.4.2 Configuración y entrenamiento del modelo

2.4.2.1 Configuraciones

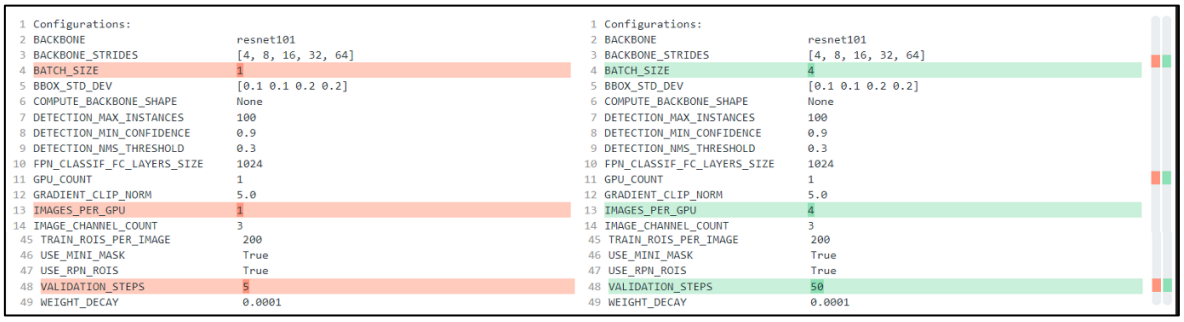
Cada implementación en base a técnicas de aprendizaje automático está asociada un número limitado de configuraciones tanto de entrenamiento e inferencia, por tal razón a continuación se detallan algunas de las configuraciones en el entrenamiento más relevantes a criterio en base al modelo pre entrenado realizado en [43]:

- **BACKBONE**: resnet101; se refiere a la red de extracción de características que se utiliza dentro de la arquitectura Mask R-CNN.
- **BATCH_SIZE**: 4; se refiere al número de ejemplos de entrenamiento utilizados en cada iteración.
- **IMAGES_PER_GPU**: 4; imágenes que se utilizaron por cada dispositivo GPU.
- **LEARNING_RATE**: 0.001; es un parámetro de ajuste en el algoritmo de optimización que determina el tamaño del paso en cada iteración mientras se mueve hacia un mínimo de una función de pérdida.
- **NUM_CLASSES**: 2; Se refiere al número de clases existentes dentro del conjunto de datos, en este caso hay dos (una para el background, y otra para personas).

- **POOL_SIZE**: 7; no es otra cosa que el muestreo descendente de una imagen, en este caso de tamaño 7x7.
- **ROI_POSITIVE_RATIO**: 0.33; se refiere al umbral mínimo que deben tener todas las regiones de interés positivas etiquetadas y detectadas.
- **STEPS_PER_EPOCH**: 500; se refiere al número de pasos totales que deben existir en una sola época.
- **VALIDATION_STEPS**: 50; es el número mínimo de pasos para que la época sea validada.
- **WEIGHT_DECAY**: 0.0001; se refiere a la técnica de regularización en aprendizaje profundo. El decaimiento de peso funciona agregando un término de penalización a la función de costo de una red neuronal que tiene el efecto de reducir los pesos durante la retro propagación. Esto ayuda a evitar que la red sobreajuste los datos de entrenamiento.

Para un desglose completo de las configuraciones tanto de entrenamiento como de inferencia, véase anexo II y III respectivamente.

Existen mínimas diferencias entre las configuraciones de entrenamiento e inferencia, lo cual se ajusta ya que para el entrenamiento se hace uso de la herramienta Google Colab, y para la inferencia se utiliza un ambiente local controlado con miniconda, por eso la figura 26 detalla las diferencias entre las distintas configuraciones realizadas.



Configurations:	
BACKBONE	resnet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	1
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.9
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5.0
IMAGES_PER_GPU	1
IMAGE_CHANNEL_COUNT	3
TRAIN_ROIS_PER_IMAGE	200
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	5
WEIGHT_DECAY	0.0001

Configurations:	
BACKBONE	resnet101
BACKBONE_STRIDES	[4, 8, 16, 32, 64]
BATCH_SIZE	4
BBOX_STD_DEV	[0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE	None
DETECTION_MAX_INSTANCES	100
DETECTION_MIN_CONFIDENCE	0.9
DETECTION_NMS_THRESHOLD	0.3
FPN_CLASSIF_FC_LAYERS_SIZE	1024
GPU_COUNT	1
GRADIENT_CLIP_NORM	5.0
IMAGES_PER_GPU	4
IMAGE_CHANNEL_COUNT	3
TRAIN_ROIS_PER_IMAGE	200
USE_MINI_MASK	True
USE_RPN_ROIS	True
VALIDATION_STEPS	50
WEIGHT_DECAY	0.0001

Figura 26. Diferencias entre archivos de configuración.

No esta demás mencionar, que al hacer uso de Google Colab para entrenamiento y el ambiente local para inferencia las configuraciones deben ser al menos en su mayoría de atributos iguales, puesto que los pesos obtenidos por el entrenamiento, archivo de extensión h5 el cual se discute en la siguiente sección, contienen la ponderación de los pesos en base a cada atributo relacionado en la configuración de entrenamiento, lo que da como resultado que en inferencia se adapte entorno a las limitaciones de la GPU utilizada la cual, para conocimiento fue una gráfica NVIDIA GTX 1660 SUPER.

2.4.2.2 Entrenamiento del modelo

La generación de un modelo idóneo en cuanto a detección y segmentación se refiere, debe afinarse hasta alcanzar un umbral de detección ideal, mientras más se aproxime al 100% en cuanto a detección y segmentación, será considerado un modelo de alta confiabilidad para que se use en fines pertinentes. Por esto se considera la siguiente información en base al número de imágenes que contendrán cada uno de los sub conjuntos de datos que son:

- Número de imágenes totales = 200
- Número de imágenes de entrenamiento = 151
- Número de imágenes de validación = 18
- Número de imágenes de testeo = 31

Con esta información y basado en las configuraciones del apartado anterior se convoca el modelo en modo de entrenamiento, lo cual se determina en base a la siguiente línea de comandos, la cual es una función propia de [43] presente en la figura 27.

```
# Create model in training mode
model = modellib.MaskRCNN(mode="training", config=config, model_dir=
MODEL_DIR)
```

Figura 27. Creación del modelo en modo de entrenamiento.

En base a los hiper parámetros configurados y el modelo en modo de entrenamiento se procede a entrenarlo con la ayuda de la función model.train, de la manera en que la figura 28 lo expresa.

```
model.train(dataset_train,dataset_val,
            learning_rate=config.LEARNING_RATE,
            epochs=30,
            layers="heads")
```

Figura 28. Entrenamiento del modelo.

De esta función los parámetros vienen dados por:

- **dataset_train**: Se refiere al conjunto de datos de entrenamiento.
- **dataset_val**: Se refiere al conjunto de datos de validación.
- **learning_rate**: Se refiere a la tasa de aprendizaje, la cual toma el valor directamente del archivo de configuración previamente detallado.
- **epochs**: Se refiere al número total de épocas (30) en que se entrenará el modelo.

- **layers**: Se refiere a las capas seleccionadas para el entrenamiento, en este caso se selecciona las capas designadas en [43] como cabezas o superiores.

Con esta información se realizan diversos entrenamientos en base a la modificación de sus épocas, que rondan los valores de 5, 10, 20 y 30, el número de imágenes pertenecientes dentro del conjunto de datos desde 50, 100 y 200 imágenes en total, así como otras consideraciones las cuales se analizaran en el apartado de pruebas y resultados.

El tiempo en que concluye cada uno de los entrenamientos varían conforme a los parámetros establecidos en la figura 28, dando así una ponderación de al menos 45 minutos de entrenamiento por cada 5 épocas configuradas teniendo en cuenta que únicamente se configuran las capas de la cabeza de la red, puesto que, si se entrenase toda la red con todas sus capas, el tiempo estimado se vería afectado drásticamente debido a la profundidad de la red.

Lo importante a considerar dentro del entrenamiento es que debido a la herramienta usada que es Google Colab el cual cuenta con un plan gratuito limitado en ciertos aspectos se debe asegurar que el procesamiento de entrenamiento no se detenga debido a que Colab tiene un tiempo de sesión activo de alrededor de 60 a 90 minutos, por lo que si el entrenamiento sobrepasa este tiempo se debe asegurar todo lo necesario para salvar todo el proceso de entrenamiento y así poder descargar los archivos de pesos de extensión h5 presentes al finalizar cada época.

Por cada época entrenada el modelo tiene un directorio de logs el cual recupera la información de los pesos de entrenamiento; la estructura de dicho directorio es el siguiente, véase figura 29 donde se observa el alojamiento de los archivos h5 que son de interés.

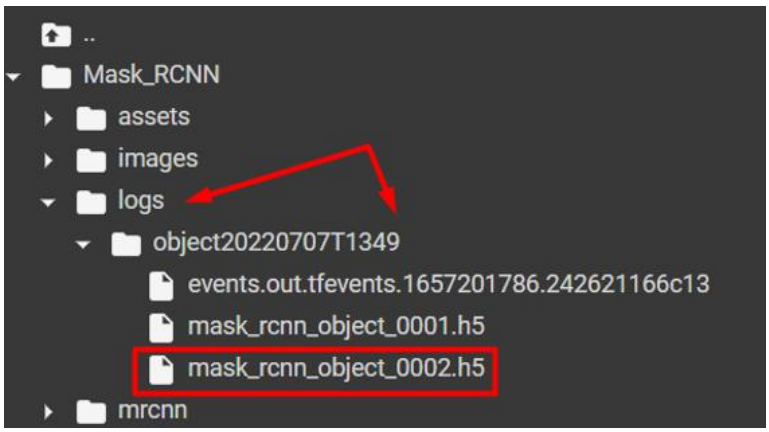


Figura 29. Árbol de directorios de pesos entrenados.

Dentro de este directorio denominado logs, se crea una carpeta de forma automática con el nombre en base a la etiqueta madre de la configuración seguido de un timestamp, el cual es una representación del instante en cuanto a fecha y hora aproximada.

Como se observa en la figura 29 por cada época entrenada se crea un archivo h5 con los pesos con el siguiente formato: mask_rcnn_object_0030.h5, en donde el número corresponde a cada una de las épocas configuradas en el entrenamiento. Con esta información, como ejemplo si se tienen un total de 30 épocas se procede a descargar el archivo h5 de la época 30 correspondiente para obtener y asegurar la existencia de los pesos para un entrenamiento de 30 épocas y 500 pasos.

2.4.3 Inferencia del modelo

Para la inferencia del modelo se emplea la construcción de un ambiente local, en donde se tiene la configuración desde la instalación de miniconda que se puede seguir a más detalle en [53] hasta su implementación total. Al usar un ambiente local se debe trabajar con versiones estables ya que al probarse con diferentes versiones a las mostradas a continuación puede arrojar errores a nivel de implementación por parte del interesado.

2.4.3.1 Instalación miniconda y entorno físico

Los pasos para la instalación se evidencian en [53], en base a la consola de comandos por parte de miniconda una vez instalado se sigue con el siguiente listado de comandos los cuales se utilizaron en concreto para un entorno con python 3.7.7, Tensorflow 2.1.0 y keras 2.3.1, la información se detalla en la figura 30.

```
$ conda create -n MaskRCNN anaconda python=3.7.7
$ conda activate MaskRCNN
$ conda install ipykernel
$ python -m ipykernel install --user --name MaskRCNN --display-name "MaskRCNN"
$ conda install tensorflow-gpu==2.1.0 cudatoolkit=10.1
$ pip install tensorflow==2.1.0
$ pip install jupyter
$ pip install keras
$ pip install numpy scipy Pillow cython matplotlib scikit-image opencv-python h5py imgaug IPython[all]
```

Figura 30. Preparación del entorno.

La correcta instalación y el tiempo de la misma dependerá de la conexión a internet, la pericia a la hora del conocimiento sobre Python, tensorflow, y keras, así también como el manejo de comandos por consola y la adquisición de dispositivos físicos compatibles.

Esta instalación se la realizó en base a los siguientes dispositivos físicos:

- CPU: Intel core I5 10400f
- GPU: Nvidia GTX 1660 Super
- MOTHERBOARD: Asus B650 Plus

Para asegurar que las instalaciones están correctas y que los dispositivos físicos están correctamente configurados la figura 31 contiene la información que de manera específica devuelve entorno a los dispositivos físicos anteriormente mencionados.

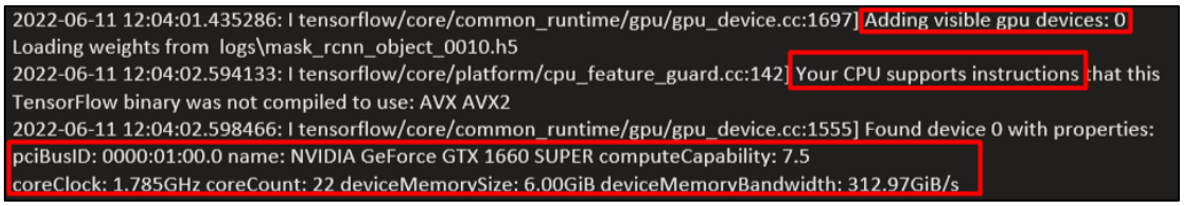


Figura 31. Detalle de dispositivos físicos configurados.

2.4.3.2 Instalación de Mask R-CNN

Al igual que en el entrenamiento se debe conseguir una versión estable de Mask R-CNN la cual se la puede descargar desde [43] como se menciona anteriormente, incluyendo que en el entorno local se añadan los comandos enlistados en la figura 32 a continuación.

```
$ python setup.py install
$ pip install git+https://github.com/philferriere/cocoapi.git#subdirectory=PythonAPI
```

Figura 32. Instalación Mask R-CNN en entorno local.

Esto se lo realiza con la finalidad de contar con un ambiente confiable y robusto de manera local para proceder con las modificaciones para la inferencia del modelo entorno a un conjunto de videos como se lo especifica en el alcance de este proyecto integrador. Dichos videos de prueba están disponibles en el Anexo V.

2.4.3.3 Modificaciones para inferencia en video

Tomando en consideración el script en el Anexo IV, para la inferencia en video se debe tener en cuenta la modificación de los siguientes parámetros:

- **model_filename = "mask_rcnn_object_0030.h5"**
  Aquí se debe cargar el modelo entrenado con el conjunto de datos a su elección creando una carpeta con el nombre “logs” y cargar el archivo h5 dentro de la misma.
- **class_names = ['BG', 'persona']**
  Las clases relacionadas con su modelo BG (BackGround) + clases customizadas.

- **min_confidence = 0.86**
  Nivel mínimo de confianza para aceptar un hallazgo como positivo dentro de la inferencia acorde a las limitaciones de este proyecto integrador.
- **camera = cv2.VideoCapture(0)**
  Si se desea procesar con webcam.
- **camera = cv2.VideoCapture("video.mp4")**
  Si se desea correr un video cargándolo desde la PC

El comando: python personaVideoDistancia.py se utiliza para realizar la detección de los videos existentes en los enlaces del anexo V, con la finalidad de recabar la información para concluir de manera satisfactoria los objetivos de este proyecto integrador. Para mayor información sobre el script utilizado véase anexo IV.

2.5 Incorporación del algoritmo de distanciamiento

2.5.1 Consideraciones iniciales

Para la incorporación de un algoritmo de distanciamiento se deben tener ciertas consideraciones, una de ellas es el posicionamiento de la cámara que da imagen al video, en este caso a la aglomeración de personas, puesto que la distancia se la calcula en base a los pixeles que existen de distancia desde un punto a otro dentro de un mismo frame. Por tal razón el ángulo y la distancia debe ser el mismo en todos los videos sujetos a análisis para poder tener una precisión lo más cercana al cálculo de la distancia como si se la realizara en la vida real.

Otra de las consideraciones es el factor profundidad, puesto que no es lo mismo tener un metro cuadrado desde el foco de la cámara hacia la mitad del frame y otra casi al final del frame, es decir, en términos de pixeles no se puede determinar con exactitud la profundidad a la que un objeto está en relación a otro más cerca con las técnicas descritas y mencionadas anteriormente, un ejemplo se lo puede visualizar en la figura 33, la cual detalla la cantidad de pixeles aproximados entre dos personas en diferentes profundidades.

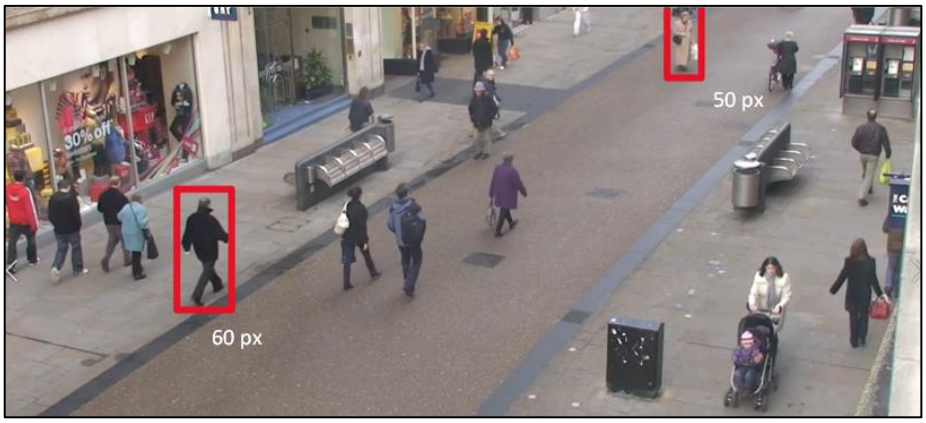


Figura 33. Ejemplo de profundidad en un determinado frame.

Como se observa en los cuadros rojos se tienen a dos personas, por lo que a nivel de pixeles el metro cuadrado que le rodea a la persona del lado izquierdo es mayor al metro cuadrado que le rodea a la persona situada en el lado derecho del frame, claro está, que en términos netamente físicos es tema de perspectiva pero que en temas de programación haría que el modelo se torne lento lo cual sería perjudicial para el comportamiento del mismo.

Es por esto que se analiza el comportamiento de los cuadros delimitadores en base al sustento científico y al estado del arte mencionado en apartados anteriores.

2.5.2 Relación entre cuadros delimitadores

Como se menciona en [3] la localización de los cuadros delimitadores viene dado por el conjunto de puntos (x,y,x+w,y+h), en donde para hallar su punto medio se realiza el cálculo en base a la ecuación 3.

$$(Cx, Cy) = (\left(x + \frac{w}{2}\right), \left(y + \frac{h}{2}\right))$$

Ecuación 3. Punto medio de cuadro delimitador.

Para que exista esta ecuación se debe encontrar los siguientes grupos de puntos:

(x,y): El cual se obtiene mediante la iteración de cada una de los cuadros en dimensión 0 y 1 respectivamente, un ejemplo se observa e la ecuación 4:

$$(x, y) = (boxes[i][0], boxes[i][1])$$

Ecuación 4. Obtención del punto superior izquierdo de cada bbox.

(w,h): El cual se obtiene mediante la iteración de la diferenciación de los cuadros en dimensión 2 y 0, y la dimensión 3 y 1 respectivamente como se observa en la ecuación 5:

$$(w, h) = (boxes[i][2] - boxes[i][0], boxes[i][3] - boxes[i][1])$$

Ecuación 5. Obtención de w,h de cada bbox.

Tanto para la ecuación 4 y 5 se debe iterar en base a todos los índices obtenidos de las detecciones en base a la función que se tiene por defecto para el manejo de redes neuronales en aprendizaje profundo; los mismos se obtienen como indica la figura 34.

```
# Indices de los cuadros delimitadores.
indices = cv2.dnn.NMSBoxes(boxes, scores, MIN_CONF, NMS_THRESH)
```

Figura 34. Índices de los cuadros delimitadores.

En donde:

- **boxes**: Se refiere al conjunto en que se alberga la información de cada uno de los cuadros delimitadores obtenidos por los resultados del modelo.
- **scores**: Se refiere al conjunto en que se guarda la información sobre la probabilidad de cada una de las detecciones obtenidas por la detección del modelo.
- **MIN_CONF**: Se refiere al parámetro de confianza mínima en el que el modelo esta testeado.
- **NMS_THRESH**: Se refiere al parámetro de supresión no máxima en correlación con el valor de confianza mínima.

Supresión no-máxima

Es una técnica utilizada en numerosas tareas de visión artificial. Es una clase de algoritmos para seleccionar una entidad (por ejemplo, para el caso, cuadros delimitadores) entre muchas entidades superpuestas [55]. Los criterios son más comúnmente alguna forma de número de probabilidad y alguna forma de medida de superposición (por ejemplo, Intersección sobre Unión) para este caso se utiliza un valor del 86%.

Con toda la información recolectada hasta el momento en base a los puntos de cada cuadro delimitador, su centro, los índices asociados, y si nivel de confianza mínima, se procede con el análisis de la distancia en base al punto centro de cada cuadro delimitador enfocado en la distancia euclidiana.

2.5.3 Algoritmo de distanciamiento mediante Scipy

Distancia Euclidiana

La distancia euclidiana es un número positivo que indica la separación que tienen dos puntos en un espacio donde se cumplen los axiomas y teoremas de la geometría de Euclides. La distancia entre dos puntos A y B de un espacio euclidiano es la longitud del

vector AB perteneciente a la única recta que pasa por dichos puntos [56]. Un ejemplo del cálculo de dicha distancia lo contiene la figura 35.

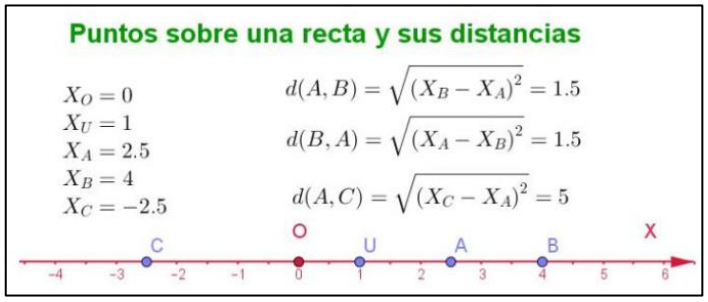


Figura 35. Espacio euclidiano unidimensional conformado por la recta (OX)

De la figura 35 se determina la ecuación utilizada para el cálculo de la distancia que la expresa la ecuación 6.

$$\boldsymbol{d_E(P_1, P_2) = \sqrt{(x1 - x2)^2 + (y1 - y2)^2}}$$

Ecuación 6. Fórmula de la distancia euclidiana.

Una apreciación más visual se observa en la figura 36, en donde cada par de punto hace referencia al centro de cada cuadro delimitador detectado por parte del modelo.

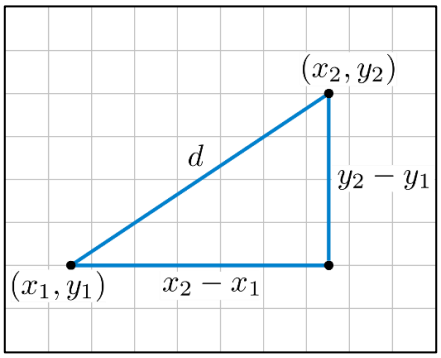


Figura 36. Ejemplo visual distancia euclidiana.

Scipy

Es un software de código abierto para matemáticas, ciencias e ingeniería. Dentro de este software tiene una serie de funciones para el cálculo de distancias computacionales, de las cuales se escoge dentro del apartado de distancias, la denominada CDIST, la cual calcula la distancia entre cada par de las dos colecciones de entradas y devuelve una matriz de distancia acorde a todos los puntos de entrada [54]. El formato de dicha función es el siguiente:

$$\boldsymbol{cdist}\ (XA, XB\ [, metric, out])$$

Concretamente para efectos de este proyecto se utiliza la siguiente ecuación 7:

$$\boldsymbol{dst} = dst.cdist\ (centers, centers, metric = "euclidean")$$

Ecuación 7. Distancia euclidiana entre los cuadros delimitadores detectados.

En donde:

- **dst**: variable en la cual se almacenará la matriz de distancias.
- **dst.cdist**: llamada a la función cdist existente en scipy.
- **centers**: conjunto de los puntos medios de cada cuadro delimitador calculados en base a la ecuación 3.
- **metric="euclidean":** se refiere a la utilización de la métrica euclidiana.

El resultado de aplicar la ecuación 7 devuelve los puntos dispuestos como m vectores fila n-dimensionales en la matriz dst en base al número de detecciones existentes en cada frame de inferencia. Observando el resultado de la figura 37, se aprecia que la matriz contiene ceros en su diagonal principal y que los valores de la matriz triangular superior e inferior son los míos, por lo que, en términos de procesamiento computacional basta con iterar dicha matriz superior o inferior para no procesar dos veces el mismo resultado y así no cargarle de tanto procesamiento al modelo.

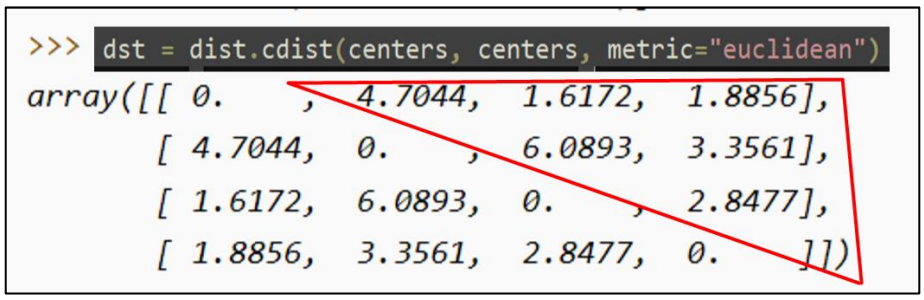


Figura 37. Matriz resultado del uso de cdist.

Una vez obtenido los resultados de las distancias existentes en cada frame acorde al número de detecciones que da como resultado la implementación del modelo, se tienen las siguientes consideraciones para determinar la distancia mínima a nivel de pixel que debe existir para que la regla del metro cuadrado, en la cual no deben existir más de dos personas no se infrinja por la aglomeración de personas:

- La distancia mínima configurada a criterio es de 50 pixeles, esto se determina en base al ángulo y posición de la cámara que proyecta los videos.

- Los cuadros delimitadores se harán visibles siempre y cuando la distancia existente entre un cuadro y otro sea menor a la distancia mínima configurada lo cual genera una alerta visual al espectador.
- Cada infracción que se cometa y se verifique en el algoritmo se guarda los índices de los centros o puntos medios de cada cuadro delimitador en un conjunto de datos con el nombre “violate”.
- Finalmente se tiene el conteo total de personas, como también el número de personas en riesgo por infringir la regla del metro cuadrado situados en la equina inferior izquierda.

Las pruebas y resultados de esta implementación se analizan en la siguiente sección.

3 PRUEBAS, RESULTADOS, CONCLUSIONES Y RECOMENDACIONES

3.1 Pruebas

Para las pruebas de la implementación del modelo basado en la red neuronal Mask R-CNN se dividen en dos grupos, el primero en base al entrenamiento y validación realizado en Google Colab, y el segundo en la inferencia realizada en un ambiente local.

Entrenamiento

Para el entrenamiento se realizaron alrededor de seis procesos de entrenamiento, esto considerando la siguiente información proporcionada en la tabla 6, en base a la misma configuración, la cual puede verse a mayor detalle en el Anexo II.

TABLA 6. Procesos de entrenamiento

Número de Imágenes	Número de épocas
50	5
100	5
200	5 / 10 / 30 / 50

Se establece en base al número de imágenes y en base al número de épocas, para obtener el óptimo resultado en el entrenamiento, sin embargo, se lo fue incrementando de la manera que indica en la tabla 6 que, en base a las 200 imágenes etiquetadas, se entrena

cuatro veces, en 5, 10, 30, y 50, en donde se visualiza que para el entrenamiento con 50 épocas la herramienta de Google Colab se interrumpe debido a la sobre limitación de recursos por lo que se llega al resultado de un archivo h5 correspondiente a la época 40, la cuál fue la última en entrenarse dentro del proceso de 50 épocas.

Es en base al archivo h5 de 30 épocas que se analizan los resultados de dichas pruebas, como se detalla en la siguiente sección de resultados, tomando en cuenta que se tendrán varios ejemplos de detección en imágenes, como también, sus respectivas gráficas de precisión en contra del recall, el ground of truth, términos que se explicaron en secciones anteriores, y finalmente los gráficos o figuras pertenecientes al valor de perdida tanto en entrenamiento como en validación de todo el proceso ejecutado para el entrenamiento de la red neuronal basado en el modelo pre entrenado Mask R-CNN.

Validación

Para las pruebas de validación se escoge mediante código Python, dentro del conjunto de datos de validación, concretamente, imágenes al azar para validar en el proceso de detección y segmentación. Dicho proceso de igual manera se evidencia en el apartado de resultados.

Inferencia

Para las pruebas de inferencia en base a lo mencionado en secciones anteriores, se tiene la configuración de un ambiente local, en donde en base a videos se procede a realizar dichas pruebas. Por un lado, se tiene entradas de videos con baja densidad de personas, es decir, que no existe un cumulo considerable de persona; de manera neutra se tiene entradas de video con densidad media y finalmente con densidad baja.

Un ejemplo de las densidades antes mencionadas se lo puede visualizar en la siguiente figura 38, en donde el recuadro con el número 1 se refiere a densidad baja, el recuadro con el número 2 se refiere a densidad media y el recuadro con el número 3 hace énfasis en la densidad alta.

Figura 38. Densidades

Se tiene estas consideraciones ya que es en base a la densidad de las aglomeraciones de personas lo cual determinará si el estudio realizado es eficaz y puede ser ejecutado con las técnicas mencionadas, las cuales se basan en segmentación e instanciación semántica.

3.2 Resultados

Para los resultados de la implementación del modelo se dividen en dos grupos, el primero en base al entrenamiento y validación realizado en Google Colab, y el segundo en la inferencia realizada en un ambiente local.

Entrenamiento y validación

Como se menciona en el apartado anterior, en base al archivo generado del entrenamiento proveniente del proceso de 30 épocas, se tienen los resultados del entrenamiento y validación.

A continuación, se observan dos ejemplos para evaluar la efectividad del modelo en base a dos imágenes escogidas aleatoriamente. La figura 39 y 42 hacen referencia a las imágenes escogidas aleatoriamente, las figuras 40 y 43 hacen referencia a la precisión y el recall de las figuras 39 y 42 respectivamente, en las figuras 41 y 44 se observa el ground truth de las figuras 39 y 42 respectivamente.

Primer resultado:

Figura 39. Imagen escogida al azar para evaluación.

Se puede observar que la figura 39 detecta y segmenta de manera idónea en la mayoría de las personas que existen dentro de la imagen. Por su lado en el lado izquierdo de la mencionada imagen se observa el score y su segmentación; y que por el lado derecho se tiene el cuadro delimitador como recurso adicional.

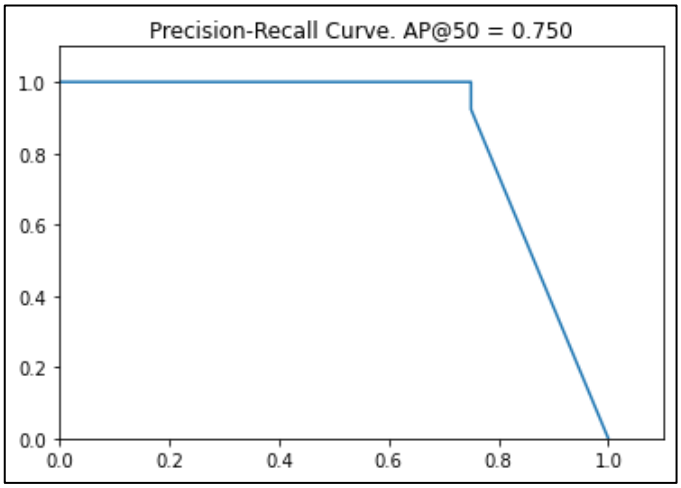


Figura 40. Precisión vs recall de la figura 39.

Se puede evidenciar que en la figura 40 se tiene un resultado de precision en contra del recall de un 75% de efectividad en base a una precisión media del 50%.

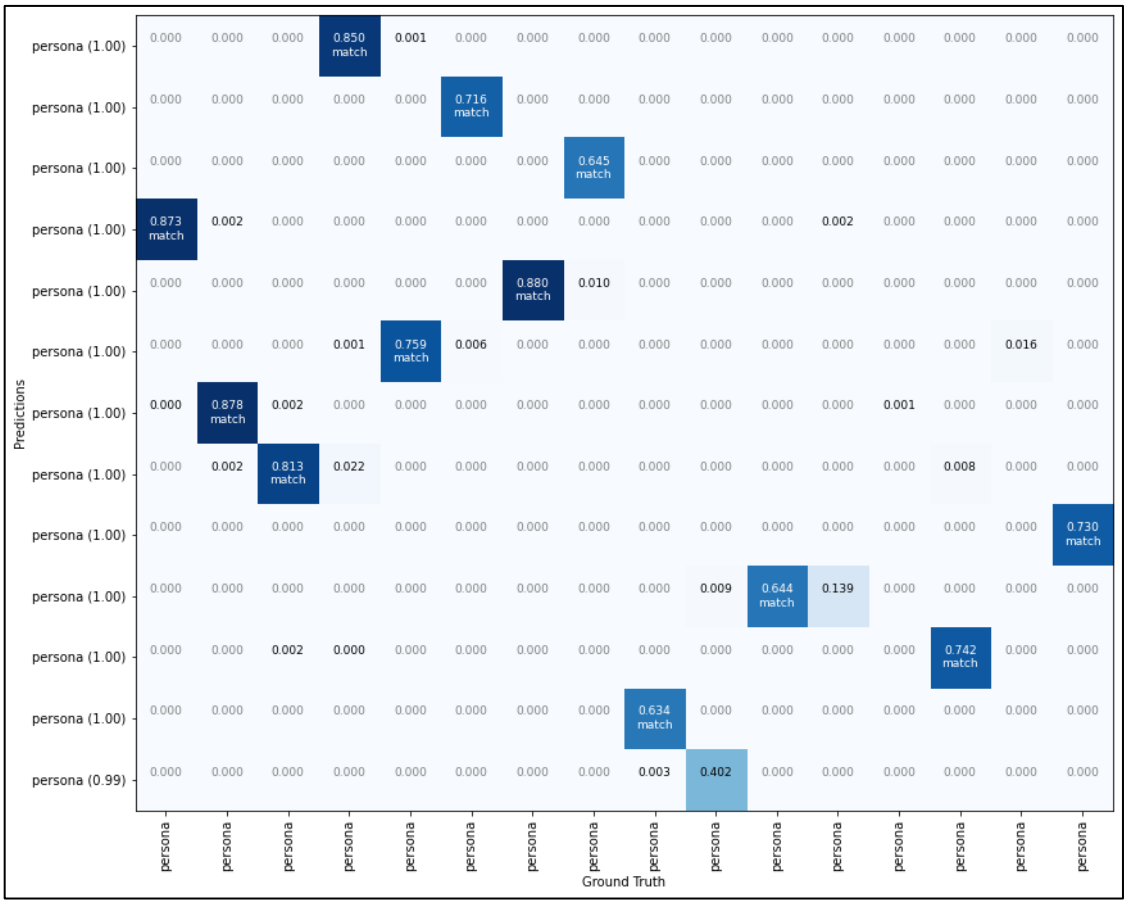


Figura 41. Ground Truth de la figura 39.

En la figura 41 se puede apreciar los matches que existen en relación a las predicciones que el sistema detectó en base a las etiquetas que tiene como campo real de detección.

Segundo resultado:

Figura 42. Segunda imagen escogida al azar para análisis.

Se puede observar que la figura 42 detecta y segmenta de manera idónea en la totalidad de las personas que existen dentro de la imagen. En el lado izquierdo de la mencionada imagen se observa el score y su segmentación; y que por el lado derecho se tiene el cuadro delimitador como recurso adicional, de igual manera cada instancia presenta un diferente color para mayor apreciación.

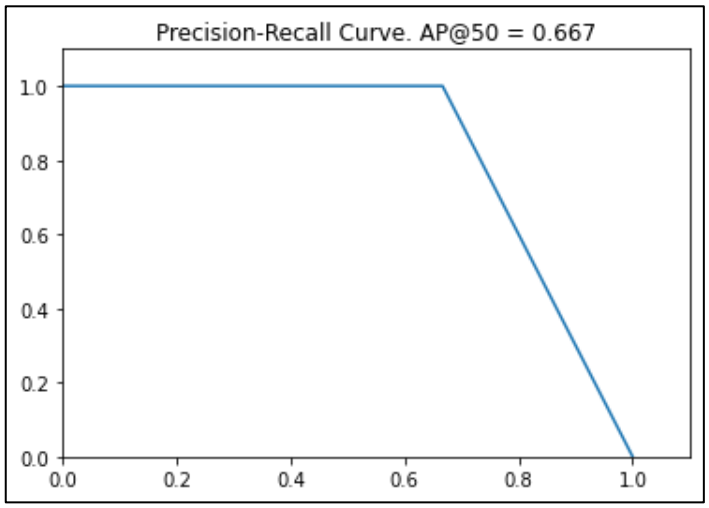


Figura 43. Precisión vs recall de la figura 42.

Se puede evidenciar que en la figura 43 se tiene un resultado de precision en contra del recall de un 66.7% de efectividad en base a una precisión media del 50%.

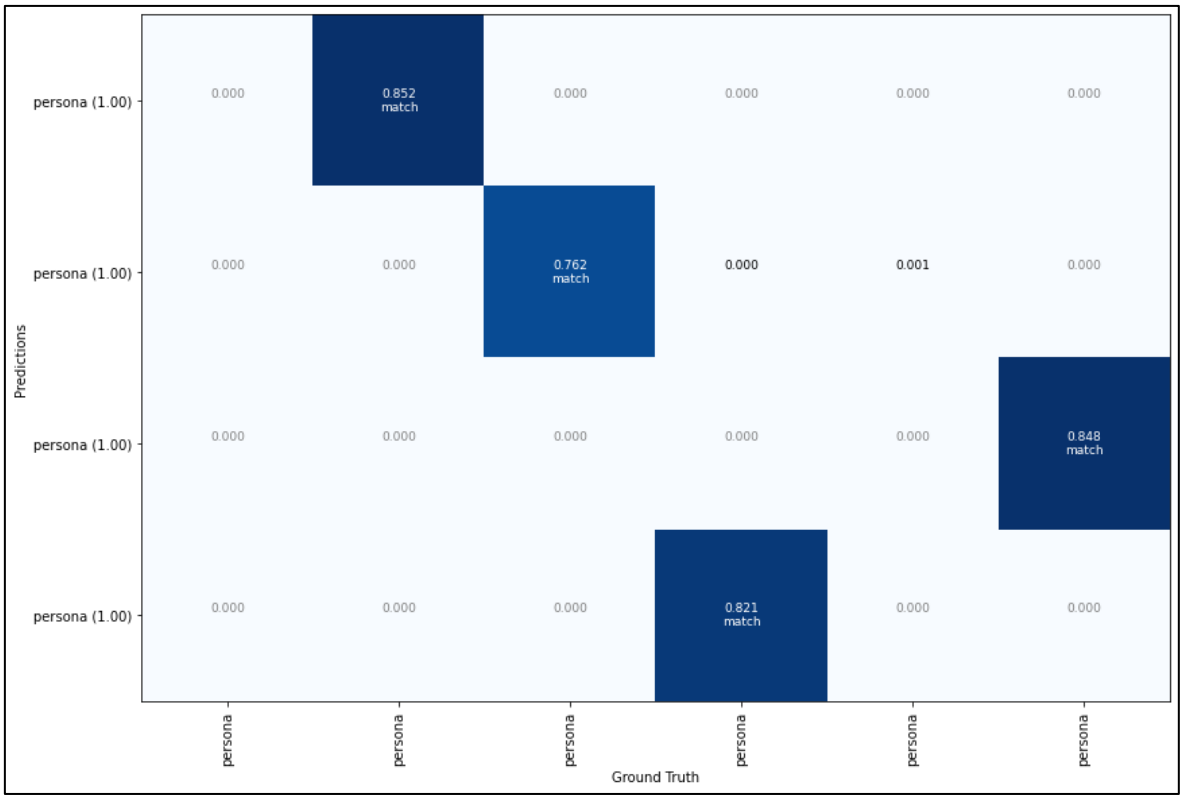


Figura 44. Ground Truth de la figura 42.

En la figura 41 se puede apreciar los matches que existen en relación a las predicciones que el sistema detectó en base a las etiquetas que tiene como campo real de detección que fueron asignadas en el proceso de entrenamiento.

Finalmente, en base a todo el proceso de entrenamiento se obtienen las siguientes figuras con los resultados de la perdida en el entrenamiento y la validación; por su lado la figura 45 hace referencia a la perdida en base a las 30 épocas ejecutadas.

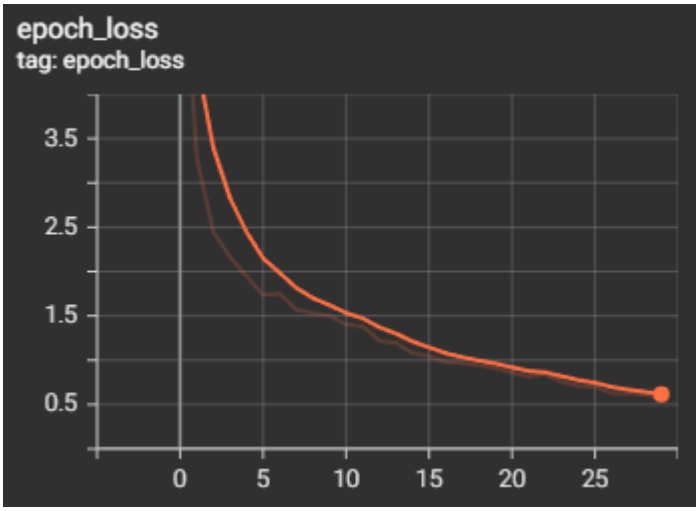


Figura 45. Perdida en base a las épocas de entrenamiento.

La figura 46 hace referencia a la perdida en base a los cuadros delimitadores entrenados en las 30 épocas.

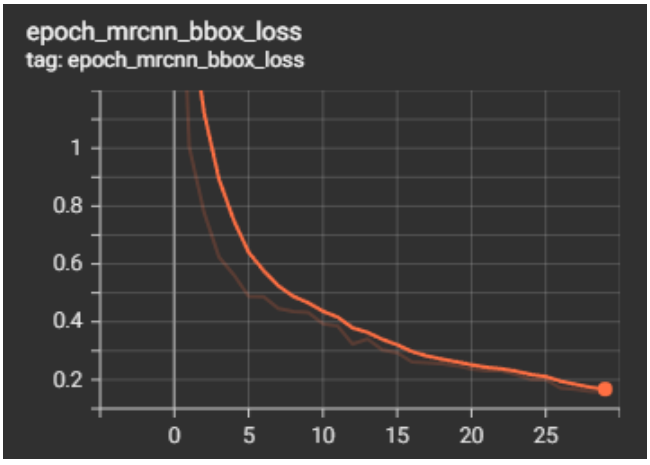


Figura 46. Perdida en base a los cuadros delimitadores.

Y la figura 47 se refiere a la perdida en el proceso de validación de las 30 épocas entrenadas.

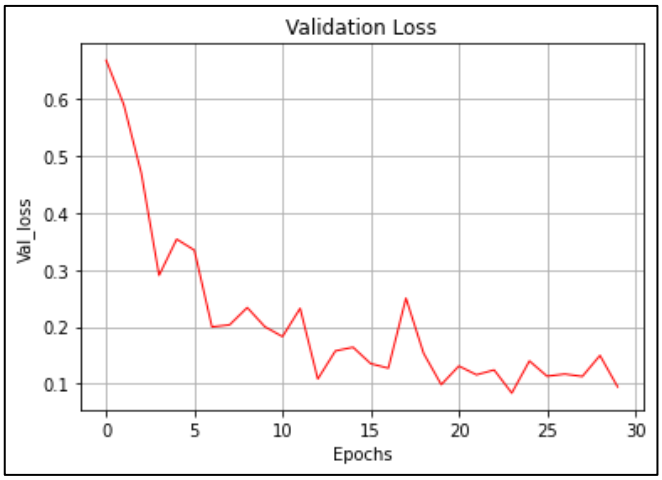


Figura 47. Perdida en la validación del entrenamiento.

En conclusión, se tiene la siguiente Tabla 7 que muestra los valore de precisión media en base al valor de intersección sobre la unión del 50%, de las épocas entrenadas con un conjunto de datos de 200 imágenes.

TABLA 7. Resultados de mAP@IoU=50.

Épocas	mAP
10	0.925
20	0.882
30	0.857

Inferencia

Los resultados de la inferencia del modelo, como se menciona en la sección anterior está basado en torno a las densidades de la aglomeración, es de tal manera que se busca un video en el cual existan estas tres densidades y así determinar que tan eficaz y confiables es el modelo. Por tal razón los resultados a continuación están sujetos a las limitaciones mencionadas en el alcance de este trabajo integrador.

Las siguientes figuras son fragmentos tomados del video resultante del Anexo V, en donde se puede apreciar de mejor manera los resultados de la implementación del modelo entrenado.

La figura 48 se puede observar el resultado de la implementación del modelo, incluyendo el factor distancia que determina si un conjunto de personas está en riesgo o no, en base a una aglomeración de personas con densidad media.

Figura 48. Inferencia en video de media densidad.

En las figuras 49 y 50, de igual manera se tienen los resultados en base a las condiciones ya mencionadas, con densidades media y baja-media respectivamente.

Figura 49. Inferencia en video de media densidad.

Figura 50. Inferencia en video de baja-media densidad

Finalmente, la figura 51 muestra los resultados en base a las condiciones ya mencionadas, pero de una densidad alta de personas considerable.

Figura 51. Inferencia en video de media-alta densidad

3.3 Conclusiones

- En base a todos los resultados obtenidos dentro de este trabajo integrador se concluye el cumplimiento de los objetivos tanto general como específicos, ya que se puede evidenciar el correcto proceso de entrenamiento basado en un modelo previamente entrenado, como también su proceso de validación y testeo para que el mismo detecte y segmente personas en base a las aglomeraciones existentes en diferentes instancias acorde a las limitaciones de este trabajo integrador.

- El trabajo otorga buenos resultados en base a la entrada de un video posicionado en una ubicación especifica, ya que al alterar la posición y ubicación de la cámara se producen inconsistencias en el modelo, por lo que, se concluye que para otro determinado sitio o ubicaciones diferentes a las planteadas en este trabajo se debe etiquetar un mayor número de imágenes y que el proceso de entrenamiento también sea mucho más robusto, intentando ver cuál es la mejor opción para las épocas, pasos de validación y pasos entre épocas para que dicho entrenamiento no presente ni sobre ajuste ni sub ajuste en sus resultados.

- Se concluye que el modelo entrenado resultante cumple con el 86% de precisión media a un valor del 50% de intersección sobre la unión, esto quiere decir que el modelo no presenta ni sobre ajuste ni sub ajuste entorno y en base a la constancia en sus gráficas de perdida en entrenamiento y validación, lo cual determina el correcto proceso de entrenamiento del modelo.

- El modelo es sumamente limitado, sin embargo, esto no quiere decir que no cumpla con el objetivo y alcance de este trabajo integrador, es decir, el modelo no detecta ni segmenta instancias provenientes a un video ubicado a más de 8 metros de altura, debido a que las personas cada vez se van haciendo más pequeñas a medida que la altura crece, esto crea un problema ya que cada vez es más difícil determinar el conjunto de pixeles característicos de una determinada persona; por tal motivo sus resultados son óptimos dentro de los parámetros y limitaciones establecidos en el alcance de este trabajo de integración.

3.4 Recomendaciones

Para trabajos futuros, el autor de este proyecto de integración recomienda:

- Se recomienda tener una solidez y un vasto conocimiento en al menos lo mencionado dentro del marco teórico, con la libertad de adentrar en conocimientos más profundos para poder entender y asimilar lo que el estado del arte puede proporcionar y de ahí ponerlo en práctica con una metodología como bien es mencionada la de Action Research.

- Para el entrenamiento de modelos similares, tomando como referencia este proyecto, es importante acotar que dispositivos de uso diario y estudiantil, entre otros, demandaría un consumo de recursos excesivos y hasta incluso en la mayoría de casos innecesario, por lo que se recomienda utilizar la nube de Google Colab para entrenar los modelos dentro de las limitaciones que tiene su plan gratuito.

- Debido al uso de frameworks y demás librerías de desarrollo es importante tener cuidado y poner énfasis en las versiones a lo largo del tiempo en cada uno de los archivos de código, puesto que un mayor uso de versiones estables y compatibles con la mayoría de sistemas asegura en conjunto un modelo más estable y robusto.

- Se debe considerar dentro del entrenamiento que Google Colab, al contar con un plan gratuito limitado, en ciertos aspectos hay que asegurar que el procesamiento de entrenamiento no se detenga debido a que tiene un tiempo de sesión activo de alrededor de 60 a 90 minutos, por lo que si el entrenamiento sobrepasa este tiempo es importante tener en cuenta que el IDE de Colab no se congele ni se pierdan los avances hasta poder finalizar con la etapa de entrenamiento por completo.

- Analizar la viabilidad del proyecto en su totalidad, puesto que en ocasiones si se requiere realizar un proyecto completamente desde cero referentes a visión computacional, demandaría un sinfín de consumo de recursos, por lo que optar por modelos pre entrenados y de ahí realizar cualquier modificación en base a los requerimientos del nuevo proyecto consumiría menor tiempo y recursos.

4 REFERENCIAS BIBLIOGRÁFICAS

[1] Ramík, D. M., Sabourin, C., Moreno, R., & Madani, K. (2014). A machine learning based intelligent vision system for autonomous object detection and recognition. Applied intelligence, 40(2), 358-375.

[2] Bolya, D., Zhou, C., Xiao, F., & Lee, Y. J. (2019). Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF international conference on computer vision (pp. 9157-9166).

[3] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).

[4] Liu, C., Tao, Y., Liang, J., Li, K., & Chen, Y. (2018, December). Object detection based on YOLO network. In 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC) (pp. 799-803). IEEE.

[5] Dick, B. (2011). Action research literature 2008—2010: Themes and trends. Action Research, 9(2), 122-143.

[6] Papadopoulos, D., Uijlings, J., Keller, F., & Ferrari, V. (2016). We don't need no bounding-boxes: Training object class detectors using only human verification. Retrieved 9 March 2022, from https://arxiv.org/abs/1602.08405

[7] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014, September). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

[8] O. Russakovsky, L.-J. Li, and L. Fei-Fei. Best of both worlds: human-machine collaboration for object annotation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 2121–2131, 2015.

[9] J. Redmon and A. Farhadi. Yolo9000: Better, faster, stronger. In Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on, pages 6517–6525. IEEE, 2017.

[10] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi. Iqa: Visual question answering in interactive environments. arXiv preprint arXiv:1712.03316, 2017.

[11] G. Papandreou, T. Zhu, N. Kanazawa, A. Toshev, J. Tompson, C. Bregler, and K. Murphy. Towards accurate multiperson pose estimation in the wild. arXiv:1701.01779, 2017.

[12] J. Dai, Y. Li, K. He, and J. Sun. R-FCN: Object detection via region-based fully convolutional networks. In NIPS, 2016.

[13] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. arXiv:1611.08050, 2016.

[14] S. Bell, C. L. Zitnick, K. Bala, and R. Girshick. Insideoutside net: Detecting objects in context with skip pooling and recurrent neural networks. In CVPR, 2016.

[15] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In CVPR, 2016.

[16] S. Xie, R. Girshick, P. Doll´ar, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. arXiv:1611.05431, 2016.

[17] Climent-Pérez, P. (2018). Videovigilancia inteligente de personas: métodos con cámaras fijas, aéreas o múltiples. Memoria explicativa de la tesis.

[18] Meivel, S., Sindhwani, N., Anand, R., Pandey, D., Alnuaim, A. A., Altheneyan, A. S., ... & Lelisho, M. E. (2022). Mask Detection and Social Distance Identification Using Internet of Things and Faster R-CNN Algorithm. Computational Intelligence and Neuroscience, 2022.

[19] Gad, A., ElBary, G., Alkhedher, M., & Ghazal, M. (2020, December). Vision-based approach for automated social distance violators detection. In 2020 International Conference on Innovation and Intelligence for Informatics, Computing and Technologies (3ICT) (pp. 1-5). IEEE.

[20] Cuestas, E. (2020). La pandemia por el nuevo coronavirus COVID-19.

[21] Iglesias-Osores, S. (2020). Importancia del aislamiento social en la pandemia de la COVID-19. Revista Medica Herediana, 31(3), 205-206.

[22] Prado-Ortega, M. X., & Grunauer-Robalino, G. R. (2020). Salud pública: detección de concentración de personas aplicando Big data para evitar brotes epidemiológicos covid-19. Identidad Bolivariana, 4(2), 5-19.

[23] Castillo-Sánchez, L. A. (2022). Una mirada al derecho de acceso a los servicios públicos en tiempos de COVID en Ecuador. Revista Jurídica Crítica y Derecho, 3(4), 17-28.

[29] Aurélien Gerón. “Chapter 10: Introduction to Artificial Neural Networks with keras.” Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow. O´Reilly

[30] Sánchez Martínez, M. (2018). Detección de personas mediante técnicas de aprendizaje automático: SVM y CNN.

[31] Aurélien Gerón. Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow. O´Reilly

[32] "Make Sense" · GitHub Pages. https://skalskip.github.io/make-sense/ (accedido el 1 de agosto de 2022).

[33] Coursera specialization Deep learning

[34] Lihat Prijono “Student Notes: Convolutional Neural Networks (CNN) Introduction”

[35] Dey, S., 2018. Hands-On Image Processing With Python. Brimingham: Packt Publishing Ltd.

[36] Sharma, S., Sharma, S., & Athaiya, A. (2017). Activation functions in neural networks. towards data science, 6(12), 310-316. (accedido el 1 de agosto de 2022).

[37] Ruiz-Santaquiteria, J., Bueno, G., Deniz, O., Vallez, N., & Cristobal, G. (2020). Semantic versus instance segmentation in microscopic algae detection. Engineering Applications of Artificial Intelligence, 87, 103271. (accedido el 1 de agosto de 2022).

[38] Rahman, M. A., & Wang, Y. (2016, December). Optimizing intersection-over-union in deep neural networks for image segmentation. In International symposium on visual computing (pp. 234-244). Springer, Cham. (accedido el 1 de agosto de 2022).

[39] Sampieri, R., Fernández, C., & Baptista, L. (2014). Definiciones de los enfoques cuantitativo y cualitativo, sus similitudes y diferencias. RH Sampieri, Metodología de la Investivación, 11-1.

[40] Stringer T. Ernest, *Action Research Third Edition*. 2000

[41] M. A. Nasrollahi, “A CLOSER LOOK AT USING STRINGER´S ACTION RESEARCH MODEL IN IMPROVING STUDENT´S LEARNING * Mohammad Ali Nasrollahi,” Int. J. Curr. Res., no. November, 2015

[42] "FiftyOne — FiftyOne 0.16.5 documentation". Voxel51 // Developer tools for ML. https://voxel51.com/docs/fiftyone/ (accedido el 1 de agosto de 2022).

[43] "GitHub - matterport/Mask_RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow". GitHub. https://github.com/matterport/Mask_RCNN (accedido el 1 de julio de 2022).

[44] "The Open Images Dataset V4: Unified image classification, object detection, and visual relationship detection at scale", Kuznetsova et al., arXiv:1811.00982 2018.

[45] Benenson, R., Popov, S., & Ferrari, V. (2019). Large-scale interactive object segmentation with human annotators. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (pp. 11700-11709).

[46] Pyrhon, “What is Python? Executive Summary | Python.org.” https://www.python.org/doc/essays/blurb/ (accedido el 4 de julio de 2022)

[47] "Google Colab". Google Research. https://research.google.com/colaboratory/intl/es/faq.html (accedido el 4 de julio de 2022).

[48] "Miniconda — Conda documentation". Conda — Conda documentation. https://docs.conda.io/en/latest/miniconda.html (accedido el 4 de julio de 2022).

[49] "JSON". JSON. https://www.json.org/json-en.html (accedido el 4 de julio de 2022).

[50] "Loading Datasets From Disk — FiftyOne 0.16.5 documentation". Voxel51 // Developer tools for ML. https://voxel51.com/docs/fiftyone/user_guide/dataset_creation/datasets.html#cocodetectiondataset (accedido el 4 de julio de 2022).

[51] Fleischhacker, D., & Stuckenschmidt, H. (2010, February). A practical implementation of semantic precision and recall. In 2010 International Conference on Complex, Intelligent and Software Intensive Systems (pp. 986-991). IEEE. (accedido el 6 de julio de 2022).

[52] Sigman, M. E., Williams, M. R., Thurn, N., & Wood, T. (2021). Validation of ground truth fire debris classification by supervised machine learning. Forensic Chemistry, 26, 100358. (accedido el 6 de julio de 2022).

[53] "Miniconda — Conda documentation". Conda — Conda documentation. https://docs.conda.io/en/latest/miniconda.html#installing (accedido el 7 de julio de 2022).

[54] "Distance computations (scipy.spatial.distance) — SciPy v1.8.1 Manual". Numpy and Scipy Documentation — Numpy and Scipy documentation. https://docs.scipy.org/doc/scipy/reference/spatial.distance.html (accedido el 7 de julio de 2022).

[55] Hosang, J., Benenson, R., & Schiele, B. (2017). Learning non-maximum suppression. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4507-4515). (accedido el 8 de julio de 2022).

[56] Lele, S. (1993). Euclidean distance matrix analysis (EDMA): estimation of mean form and mean form difference. Mathematical Geology, 25(5), 573-602. (accedido el 8 de julio de 2022).

[57] GitHub - weiliu89/caffe at ssd. (s. f.). GitHub. https://github.com/weiliu89/caffe/tree/ssd

[58] GitHub - david8862/keras-YOLOv3-model-set: end-to-end YOLOv4/v3/v2 object detection pipeline, implemented on tf.keras with different technologies. (s. f.). GitHub. https://github.com/david8862/keras-YOLOv3-model-set

[59] GitHub - pjreddie/darknet: Convolutional neural networks. (s. f.). GitHub. https://github.com/pjreddie/darknet

[60] GitHub - bonlime/keras-deeplab-v3-plus: Keras implementation of Deeplab v3+ with pretrained weights. (s. f.). GitHub. https://github.com/bonlime/keras-deeplab-v3-plus

[61] GitHub - matterport/Mask_RCNN: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. (s. f.). GitHub. https://github.com/matterport/Mask_RCNN

[62] GitHub - dbolya/yolact: A simple, fully convolutional model for real-time instance segmentation. (s. f.). GitHub. https://github.com/dbolya/yolact

[63] "GitHub - CesarTaco1007/TIC_SegmentacionSemantica_AglomeracionPersonas_CPTA: Implementación de un modelo de aprendizaje profundo para la detección de personas provenientes de un video, basado en el modelo pre entrenado Mask R-CNN". GitHub. https://github.com/CesarTaco1007/TIC_SegmentacionSemantica_AglomeracionPersonas_CPTA (accedido el 1 de agosto de 2022).

5 ANEXOS

ANEXO I. Anotaciones (JSON) mediante herramienta MakeSenseAI en formato COCO

ANEXO II. Configuración de entrenamiento

ANEXO III. Configuración de inferencia

ANEXO IV. Script Detección de Personas

ANEXO V. Enlaces

ANEXO I

Anotaciones (JSON) mediante herramienta MakeSenseAI en formato COCO

{"info":
{"description":"my-project-name"},
"images":[
{"id":1,"width":640,"height":480,"file_name":"1061388561_90a69fb345_z.jpg"},{"id":2,"width":512,"height":384,"file_name":"21021041_43564e03cc_z.jpg"},{"id":3,"width":637,"height":422,"file_name":"2208466227_4135cdd37c_z.jpg"},{"id":4,"width":480,"height":640,"file_name":"2353543124_8c79cc1b9d_z.jpg"},{"id":5,"width":520,"height":640,"file_name":"2456235787_0d637e59c1_z.jpg"},{"id":6,"width":480,"height":640,"file_name":"247156454_e89caddd47_z.jpg"},{"id":7,"width":480,"height":640,"file_name":"2524858550_77fa484932_z.jpg"},{"id":8,"width":480,"height":640,"file_name":"2616366588_0f7eb53aa5_z.jpg"},{"id":9,"width":640,"height":480,"file_name":"276307105_94562222e2_z.jpg"},{"id":10,"width":640,"height":480,"file_name":"2767824906_570e58e95e_z.jpg"},{"id":11,"width":427,"height":640,"file_name":"2805578619_e1befc8e35_z.jpg"},{"id":12,"width":403,"height":640,"file_name":"2915311228_4153dc0abb_z.jpg"},…..]
"annotations":[
{"id":0,"iscrowd":0,"image_id":1,"category_id":1,"segmentation":[[274.6293085190682,265.7092840993048,281.2842997625008,264.6584960082365,286.53824021784226,258.7040301588495,292.8429687642521,248.19614924816648,296.34559573447973,243.6427341868705,296.69585843150253,238.03853103450624,292.8429687642521,235.93695485236964,295.6450703404342,225.42907394168662,299.49796000768464,217.37303191016298,305.1021631600489,214.2206676369581,304.05137506898063,211.41856606077596,300.1984854017302,210.01751527268488,299.49796000768464,205.81436290841168,296.69585843150253,202.66199863520677,299.1476973106619,198.1085835739108,300.1984854017302],"bbox":[274.6293085190682,186.19965187513674,72.50437828371275,187.3905429071803],"area":6018.7522428160855},…],
"categories":[{"id":1,"name":"persona"}]}

Archivo completo:
https://drive.google.com/file/d/1z2_8NMewQsMpYTPGxEkifs3GvS7ED1xl/view?usp=sharing

ANEXO II

Train Configurations:

BACKBONE: resnet101

BACKBONE_STRIDES: [4, 8, 16, 32, 64]

BATCH_SIZE: 4

BBOX_STD_DEV: [0.1 0.1 0.2 0.2]

COMPUTE_BACKBONE_SHAPE: None

DETECTION_MAX_INSTANCES: 100

DETECTION_MIN_CONFIDENCE: 0.9

DETECTION_NMS_THRESHOLD: 0.3

FPN_CLASSIF_FC_LAYERS_SIZE: 1024

GPU_COUNT: 1

GRADIENT_CLIP_NORM: 5.0

IMAGES_PER_GPU: 4

IMAGE_CHANNEL_COUNT: 3

IMAGE_MAX_DIM: 512

IMAGE_META_SIZE: 14

IMAGE_MIN_DIM: 512

IMAGE_MIN_SCALE: 0

IMAGE_RESIZE_MODE: square

IMAGE_SHAPE: [512 512 3]

LEARNING_MOMENTUM: 0.9

LEARNING_RATE: 0.001

LOSS_WEIGHTS:
{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}

MASK_POOL_SIZE: 14

MASK_SHAPE: [28, 28]

MAX_GT_INSTANCES: 100

MEAN_PIXEL: [123.7 116.8 103.9]

MINI_MASK_SHAPE: (56, 56)

NAME: object

NUM_CLASSES: 2

POOL_SIZE: 7

POST_NMS_ROIS_INFERENCE: 1000

POST_NMS_ROIS_TRAINING: 2000

PRE_NMS_LIMIT: 6000

ROI_POSITIVE_RATIO: 0.33

RPN_ANCHOR_RATIOS: [0.5, 1, 2]

RPN_ANCHOR_SCALES:
(32, 64, 128, 256, 512)

RPN_ANCHOR_STRIDE: 1

RPN_BBOX_STD_DEV: [0.1 0.1 0.2 0.2]

RPN_NMS_THRESHOLD: 0.7

RPN_TRAIN_ANCHORS_PER_IMAGE:
256

STEPS_PER_EPOCH: 500

TOP_DOWN_PYRAMID_SIZE: 256

TRAIN_BN: False

TRAIN_ROIS_PER_IMAGE: 200

USE_MINI_MASK: True

USE_RPN_ROIS: True

VALIDATION_STEPS: 50

WEIGHT_DECAY: 0.0001

ANEXO III

Inference Configurations:

BACKBONE: resnet101

BACKBONE_STRIDES: [4, 8, 16, 32, 64]

BATCH_SIZE: 1

BBOX_STD_DEV: [0.1 0.1 0.2 0.2]

COMPUTE_BACKBONE_SHAPE: None

DETECTION_MAX_INSTANCES: 100

DETECTION_MIN_CONFIDENCE: 0.9

DETECTION_NMS_THRESHOLD: 0.3

FPN_CLASSIF_FC_LAYERS_SIZE: 1024

GPU_COUNT: 1

GRADIENT_CLIP_NORM: 5.0

IMAGES_PER_GPU: 1

IMAGE_CHANNEL_COUNT: 3

IMAGE_MAX_DIM: 512

IMAGE_META_SIZE: 14

IMAGE_MIN_DIM: 512

IMAGE_MIN_SCALE: 0

IMAGE_RESIZE_MODE: square

IMAGE_SHAPE: [512 512 3]

LEARNING_MOMENTUM: 0.9

LEARNING_RATE: 0.001

LOSS_WEIGHTS:
{'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'mrcnn_mask_loss': 1.0}

MASK_POOL_SIZE: 14

MASK_SHAPE: [28, 28]

MAX_GT_INSTANCES: 100

MEAN_PIXEL: [123.7 116.8 103.9]

MINI_MASK_SHAPE: (56, 56)

NAME: object

NUM_CLASSES: 2

POOL_SIZE: 7

POST_NMS_ROIS_INFERENCE: 1000

POST_NMS_ROIS_TRAINING: 2000

PRE_NMS_LIMIT: 6000

ROI_POSITIVE_RATIO: 0.33

RPN_ANCHOR_RATIOS: [0.5, 1, 2]

RPN_ANCHOR_SCALES:
(32, 64, 128, 256, 512)

RPN_ANCHOR_STRIDE: 1

RPN_BBOX_STD_DEV: [0.1 0.1 0.2 0.2]

RPN_NMS_THRESHOLD: 0.7

RPN_TRAIN_ANCHORS_PER_IMAGE:
256

STEPS_PER_EPOCH: 500

TOP_DOWN_PYRAMID_SIZE: 256

TRAIN_BN: False

TRAIN_ROIS_PER_IMAGE: 200

USE_MINI_MASK: True

USE_RPN_ROIS: True

VALIDATION_STEPS: 5

WEIGHT_DECAY: 0.0001

ANEXO IV

Script Detección de personas (cálculo de distancia)

```
def calcular_dist(boxes, indexes):
  if len(indexes) > 2:
    idf = indexes.flatten()
    centers = list()
    status = list()
    violate = set()
    safe = list()
    low_risk = list()
    high_risk = list()
    for i in idf:
    #The mask RCNN bounding box format demands the top left and bottom right coordinate of the box which is given by: [x, y, x+w, y+h].
    #El formato de la bbox de mask RCNN arroja el borde superior izquierda, y el borde inferior derecho en base a las coordenada dadas por: [x, y, x+w, y+h]
    # x, y, x+w, y+h
      (x,y) = (boxes[i][0],boxes[i][1]) # top-left position
      (w,h) = (boxes[i][2]-boxes[i][0],boxes[i][3]-boxes[i][1])

      centers.append([int(y+(h/2)),int(x+(w/2))])
      #print("centros ", centers)

    dst = dist.cdist(centers, centers, metric="euclidean")
    #print("matriz distancia \n", dst)
```

```
    # loop over the upper triangular of the distance matrix
    # Recorrer la matriz de distancia contando unicamente su traingularidad superior
    for i in range(0, dst.shape[0]):
        for j in range(i + 1, dst.shape[1]):
            # check to see if the distance between any two centroid pairs is less than the configured number of pixels
            # revisar si la distancia que existe entre dos distintos centroides es menor que la cantidad de pixeles configurados inicialmente.
            if dst[i, j] < MIN_DISTANCE:
                # update our violation set with the indexes of the centroid pairs
                # actualizamos el conjunto de "violate" con los indices de los centroides que cumplen esta condicion
                violate.add(i)
                violate.add(j)
                high_risk.append([centers[i], centers[j]])
                #y,  x, y+h x+w
                y1, x1, y2, x2 = boxes[i]
                #print("coordenadas ", boxes[i])
                cv2.rectangle(frame_obj, (x1, y1), (x2, y2),(0,10,255), 2)
    person_count = len(centers)
    safe_count = len(centers)-len(violate)
    high_risk_count = len(violate)

    return high_risk_count, safe_count, idf, high_risk
```

ANEXO V

Enlaces:

Repositorio:
https://github.com/CesarTaco1007/TIC_SegmentacionSemantica_AglomeracionPersonas_CPTA

Dataset:

https://drive.google.com/drive/folders/1Vt20XO5ZDVh6_u6fi_fn55izbw9XcbXq?usp=sharing

Resultado Video 1:

https://www.youtube.com/watch?v=y39gL0RWYtQ

Resultado Video 2:

https://www.youtube.com/watch?v=eMcebxnjVKg