

Detección de Distanciamiento Físico vía YOLOv5

Milton Gener Palacin Grijalva

Maestría en Ciencias de Ciencia de la Computación

Universidad Nacional de Ingeniería, Perú

mpalacing@uni.pe

Abstract—La enfermedad infecciosa causada por coronavirus 2019 (COVID-19) ha traído la crisis mundial con su mortal propagación a más de 188 países, y alrededor de 27,146,371 casos confirmados junto con 889,364 muertes en todo el mundo, al 7 de setiembre del 2020¹. Mantener la distancia física, entre las personas, se ha convertido en una medida indispensable para disminuir el contagio. El presente trabajo propone un marco de trabajo basado en *Redes Neuronales Convolucionales (RNC)*² para detectar de manera automática el distanciamiento físico entre las personas, en el proceso de examinar imágenes o monitorizar videos de vigilancia. El marco de trabajo propuesto hará uso del modelo de detección de objetos YOLOv5.

Palabras Clave—COVID-19, Distanciamiento Físico, YOLO, Detección de Objetos.

1. Introducción

La enfermedad por coronavirus COVID-19 fue reportada por primera vez en Wuhan, China, el 31 de diciembre de 2019. Hasta el 11 de marzo ya se había extendido por 114 países con 118.000 casos activos y 4000 muerte. El 11 de marzo del 2019 la OMS³ declara esta enfermedad como una pandemia [1]. Al 7 de setiembre del 2020, hay más de 27,146,371 casos y 889,364 muertes en todo el mundo [2]. En la actualidad, una de las mejoras formas de evitar contraer COVID-19 es evitar exponerse al coronavirus. El Ministerio de Salud ha recomendado pautas que incluyen el distanciamiento social, distanciamiento físico, uso de máscaras, lavarse las manos con frecuencia entre otras recomendaciones. El distanciamiento físico se refiere a mantener cierta distancia entre las personas que se encuentra en la misma área. Varios grupos de investigadores demostraron que el distanciamiento social, donde incluye el distanciamiento físico, puede reducir significativamente el número total de casos infectados [3], [4].

Varias tecnologías han sido propuestas para detectar el acercamiento excesivo y la mayoría de ellos utilizan alguna forma de comunicación. Ejemplo de esta comunicación incluye WiFi, Bluetooth, seguimiento en base a la conectividad de celular, RFID, Ultra Wide Band (UWB), etc.

En el presente trabajo se propone un procedimiento inicial para detectar el distanciamiento físico de cualquier imagen analizada. Se hace uso del sistema de detección

1. Fuente: Tablero COVID-19 por el Centro de Ciencia e Ingeniería de Sistemas (CSSE) de la Universidad Johns Hopkins (JHU), <https://gisanddata.maps.arcgis.com/apps/opsdashboard/index.html#/bda7594740fd40299423467b48e9ecf6>

2. RNC: en este trabajo se entenderá como RNC a la combinación de capas convolucionales y lineales

3. OMS: Organización Mundial de la Salud

de objetos YOLOv5 que implementa un modelo robusto y moderno de *Redes Neuronales Convolucionales (RNC)* y *lineales* propuesto por YOLOv5. El resto de la estructura de presente artículo está organizado de la siguiente manera. La Sección 2 presenta los antecedentes y trabajos relaciones a la detección de objetos, incluye State-of-The-Art, y distanciamiento físico. En la Sección 3 presenta la metodología para los datos, arquitectura de RNC y principales configuraciones. En la Sección 4 se describe de forma detallada la experimentación y los resultados. En la Sección 5 presenta una breve discusión para comparar los resultados obtenidos. En la Sección 6 presenta las conclusiones y trabajos futuros.

Todo el código y cuadernos de trabajo están disponible de manera directa en:

- https://github.com/miltonpalacin/uni_03_epi/tree/master/trabajo_final o
- https://drive.google.com/drive/folders/1CvSPsLqiXOI1Tp_Cetux-L60QiwbHhId?usp=sharing

2. Antecedentes y Trabajos Relacionados

2.1. Detección de Objetos

2.1.1. Antes de RNC. Uno de los primeros algoritmos eficientes de detección de objetos fue *Viola-Jones Algorithm* del 2001 [5], los autores mostraron la detección de rostros humanos en “tiempo real”⁴ directamente de un cámara Web. El algoritmo de *Viola-Jones* consiste en extraer características simples usando transformaciones *Haar Wavelet*⁵ [6], donde cada características se representa por una “imagen integral”⁶, para luego obtener un conjunto de entrenamiento de un algoritmo de aprendizaje AdaBoost⁷.

Otra técnica más eficientes para la detección de objetos fue *Histograms of Oriented Gradients for Human Detection* del 2005 [7], el método está basado en la evaluación de histogramas locales bien normalizados de orientación de gradiente de imagen en una cuadrícula densa (píxeles simples) acumulados en bloques de descriptores normalizados HOG. La cuadrícula de descriptores de HOG combinando en un clasificador de convencional basado en SVM⁸.

4. Real-Time: Controla un entorno recibiendo datos, procesándolo y devolviendo los resultados con la suficiente rapidez (milisegundos o microsegundos) para afectar el medio ambiente en ese momento.

5. Haar Wavelet: Conocido actualmente como Haar-like; son filtros de re-escalado, proyecciones de baja frecuencia.

6. Integral Image: Tabla de áreas sumarizadas. Utilizada para extraer las características rectangulares.

7. Adaptive Boosting (AdaBoost): Es un algoritmo de Machine Learning no supervisado para clasificaciones fuertes.

8. Support Vector Machine (SVM): Son algoritmos de aprendizaje supervisado relacionado con problemas de clasificación y regresión

Después del desafío de reconocimiento visual a gran escala lanzada en el 2012 por ImageNet⁹ (ILSVRC¹⁰), donde la arquitectura RNC propuesta por Kriszhevsky¹¹ [8] tuvo uno de los mejores rendimientos. Las RNC o CNN¹², por sus siglas en inglés, en adelante se convirtieron en el estándar por defecto para la clasificación y detección de imágenes.

2.1.2. La Era de RNC. La tarea de clasificar y detectar objetos dentro de las imágenes ha resultado compleja para el ojo humano cuando se trata de observar todo el panorama. La tarea de detectar objetos incluye: identificar (clasificar), detectar los bordes o cuadros limitadores, diferenciar y encontrar las relaciones de todos los objetos dentro de una imagen que representa el panorama. Las RNC han demostrado ser una herramienta efectiva que ayuda a superar esta tarea compleja.

Existen varias RNC para clasificar y detectar objetos, así como VGGNet [9] o Inception [10]. Estas RNC tienen dos características principales, el uso abundante GPU y entrenamientos con grandes conjuntos de datos. El método básico utilizado es recorrer cuadro a cuadro (Small Window) toda la imagen; en cada recorrido la RNC (el clasificador) obtiene una predicción de qué tipo de objeto está dentro del cuadro. Este método resulta demasiado lento y es necesario ejecutar varias veces el clasificador (reutilizar el clasificador).

En el 2015 la propuesta R-CNN¹³ [11] brinda un mejor enfoque:(1) propone alrededor de 2000 regiones (de abajo hacia arriba) de la imagen de entrada, (2) calcula las características de cada propuesta utilizando una gran RNC, y luego (3) clasifica cada región utilizando SVM con kernel lineal.

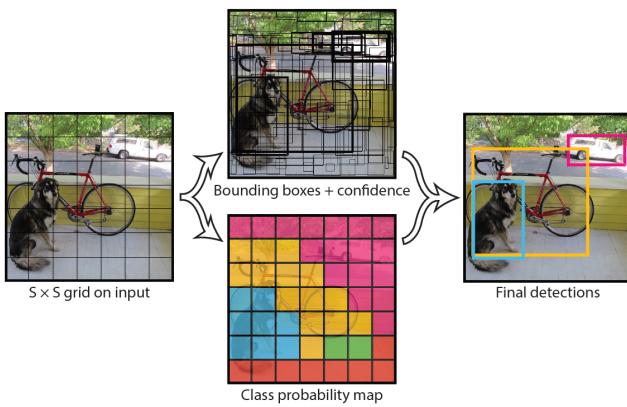


Figure 1. Modelo de detección como un problema de regresión.

2.1.3. You Only Look Once (YOLO) . YOLO es el State-of-The-Art para la detección de objetos, con un enfoque diferente; requiere examinar cualquier imagen una sola vez para predecir que clases o tipos de objetos están presentes y dónde están.

El sistema base de detección de objetos propuesta por YOLOv1 [12] consiste en dividir la imagen en una cuadrícula de $S \times S$ celdas, si el centro de un objeto

9. ImageNet: Es una base de datos de imágenes (14,197,122 imágenes), usa la base de datos léxica WordNet para clasificar las imágenes.

10. ImageNet Large Scale Visual Recognition Challenge (ILSVRC)

11. AlexNet: Red neuronal convolucional compuesta por cinco capas convolucionales y tres capas totalmente conectadas

12. Convolutional Neural Network (CNN)

13. Regions with CNN features (R-CNN): Compuesta por trece capas convolucionales y tres capas totalmente conectadas.

cae en una celda, esa celda es responsable de detectar ese objeto. Cada celda predice B cuadros delimitadores ($P(\text{Object})$) y C probabilidades del tipo o clase objeto ($P(\text{Class}(i)|\text{Object})$). La predicción del cuadro delimitador tiene 5 componentes: $(x, y, w, h, \text{confianza})$. El (x, y) representa las coordenadas del centro cuadro delimitador relativo a las dimensiones de la imagen. El (w, h) representa el ancho y la altura (normalizado a $[0, 1]$) del cuadro delimitador, relativo a las dimensiones de la imagen. La confianza representa el $P(\text{Object}) * IoU^{14}$ ($\text{pred}, \text{truth}$) entre el cuadro delimitador predicho y el real. Estas predicciones se codifican como un tensor de $S \times S \times (B * 5 + C)$. Como se observa en la figura 1, YOLO modela la detección como un problema de regresión.

1) **Diseño de la Red:** YOLOv1 implementa el modelo base como un RNC; donde las capas convolucionales iniciales extraen las características (*claseficación*) mientras que las capas totalmente conectadas predicen las probabilidades y coordenadas de los cuadro delimitadores (*detección*). La arquitectura de la RNC es inspirada en el modelo de GoogLeNet para clasificación de imágenes. La RNC tiene 24 capas convolucionales y 2 capas totalmente conectadas, utiliza filtros de convolución de 1×1 seguidos por filtros de 3×3 . La salida final de la RNC es un tensor de predicciones $S \times S \times (B * 5 + C)$.

2) **Entrenamiento:** YOLOv1 entrena previamente las capas convolucionales con imágenes de ImageNet⁹ (1000 tipos o clases) a resolución de 224×224 y luego duplica la resolución para la detección. En el entrenamiento la red se optimiza siguiendo una función de pérdida, compuesta por la *suma de los errores cuadráticos* de todos los elementos que intervienen en el modelo. En la ecuación 1 los parámetros $\lambda_{coord} = 5$ y $\lambda_{noobj} = 0.5$ son respectivamente el aumento de pérdida para la predicción de las coordenadas del cuadro delimitador y la disminución de pérdida para la predicción de confianza para cuadros que no contienen objetos. El parámetro $\mathbb{1}_i^{obj} \in (0, 1)$ indica si la celda es parte de un objeto y $\mathbb{1}_{ij}^{obj} \in (0, 1)$ indica que el $j^{\text{ésimo}}$ predictor de cuadros delimitadores en la celda $i^{\text{ésima}}$ es responsable de detectar el objeto.

3) **Inferencia:** Al igual que el entrenamiento, solo se necesita una RNC para detectar objetos en la imágenes a evaluar. El diseño de cuadrícula refuerza la diversidad espacial en la predicción de los cuadros delimitadores: Si una celda contiene a un objeto, la red predice un cuadro para el objeto en la celda, sin embargo, si la celda contiene “parte” del objeto; la red predice un cuadro delimitador por cada una de las celdas que forman todas las partes del objeto. Para seleccionar el mejor cuadro, YOLO usa “Supresión No Máxima”¹⁵, que consiste en calcular todos los “cocientes de solapamiento (IoU)” y para luego eliminar los que tengan valores no significativos.

14. IoU: Intersection over Union, también llamado “cociente de solapamiento”, es igual a la intersección de los cuadro entre la unión de los cuadro; $IoU = \text{Overlap}/\text{Union}$

15. Non-maximal Supression (NMS)

$$\begin{aligned}
& \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] \\
& + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} [(\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] \\
& + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{obj} (C_i - \hat{C}_i)^2 \\
& + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{noobj} (C_i - \hat{C}_i)^2 \\
& + \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \quad (1)
\end{aligned}$$

YOLOv2 [13] realiza las siguientes mejoras: (1) Agrega BachNormalization¹⁶, lo que permite mejorar en más del 2% la mAP¹⁷. (2) Ajusta la red de clasificación a imágenes con resolución de 448×448 por 10 épocas en ImageNet⁹, lo que permite incrementar en casi 4% la mAP¹⁷. (3) Remueve las capas "totalmente conectadas" para utilizar cuadros de anclaje (*Anchor Box*) para predecir cuadros delimitadores, lo que permite predecir más de mil cuadros comparados con solo 98 por imagen en YOLOv1. (4) Ejecuta *K-Means*¹⁸ *Clustering* en el entrenamiento de los cuadros delimitadores, variando la distancia Euclidiana por $d(box, centroid) = 1 - IoU(box, centroid)$. (5) Ajustes en la localización de los cuadros delimitadores, junto a (4) permite incrementar en casi 5% mAP¹⁷. (6) Agrega una capa de paso (*passthrough layer*) de resolución 26×26 , lo que permite incrementar en casi 1% la mAP¹⁷, (7) Entrenamiento a multiescala. (8) Nuevo modelo para la clasificación basado en DarkNet¹⁹⁻¹⁹: 19 capas convolucionales y 5 maxpooling.

YOLOv3 [14] realiza las siguientes mejoras: (1) Uso de clasificación *multilabel*; reemplaza la función *softmax* con una *logistic*, usa *cross-entropy-loss* para cada label. (2) Uso de regresión logística para la predicción de cuadros delimitadores. (3) Predicción de cuadros delimitadores a tres diferentes escala el cual usa 9 grupos de *Anchor Box*. (4) Nuevo modelo para la clasificación basado en DarkNet¹⁹⁻⁵³.

YOLOv4 [15] realiza cambios importantes orientados sistemas en producción y computación paralela. (1) Usa una arquitectura moderna: Backbone (CSPDarknet53 [16]) orientado a mejorar la clasificación, Neck (PANet path-aggregation) orientado a mejorar la predicción, Head (YOLOv3) orientado a mejorar la detección. (2) Usa Bag of Freebies (Data Augmentation for trained offline, CutMix, Genetic Algorithms, Self-Adversarial Training, CIOU-loss, etc.) y Bag of Special (post-processing methods, Cross-Stage, SPP-block, DIoU-NMS, etc.) para Backbone y Head para mejorar la detección de objetos en el entrenamiento.

YOLOv5 [17] lanzada el 25 de junio del 2020 aún no tiene un artículo académico relacionado. Es una extensión natural del repositorio YOLOv3 Pytorch además de

16. BachNormalizacion: método estadístico para normalizar o escalar los datos de entrada o los mapas de activación.

17. mean Average Precisión (mAP): métrica para medir la precisión en RNC para la detección de objetos. Calcula el área bajo la curva (AUC) determinado por la "Precision vs Recall".

18. K-Means: Método de agrupamiento de un conjunto de n observaciones en k grupos.

19. DarkNet: Red neuronal Open Source construida en *C*.

rescartar los principales aportes de YOLOv4. (1) Formula un nuevo modelo de configuración *.yaml*. (2) Migración de Darknet, implementado en *C*, a PyTorch.

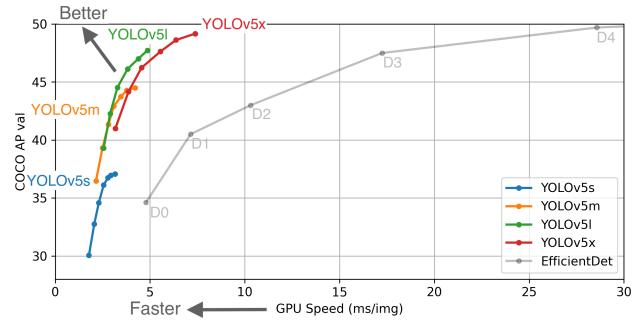


Figure 2. Precisión de YOLOv5 en datos de validación sobre COCO val2017. La comparación es entre los modelos pre-entrenados en YOLOv5 (s=small, m=medium, l=large, x=extra large) y EfficientDet (modelo State-Of-The-Art para la detección de objetos).

2.2. Distanciamiento Físico

Como parte de las recomendaciones de *distanciamiento social* que dan todas las organizaciones de salud, el distanciamiento físico hace referencia a la mayor o menor lejanía entre las personas que puede medirse en metros o pies.

La OMS recomienda al público mantener como mínimo un metro (1) (tres pies) de distancia con respecto a otras personas [18]. KidsHealth²⁰ recomienda a los padres asegurar como mínimo dos (2) metros (seis pies) de distancia de otras personas [19].

Un estudio realizado por Talib Dbouk y Dimitris Drikakiss [20] demuestran que en determinadas condiciones ambientales como la velocidad del viento las gotas de la saliva que se desprenden de la tos pueden propagar a más de 5 metros de distancia.

En la actualidad existen varios trabajos relacionados al monitoreo del distanciamiento social para lugares públicos y centros de trabajo. El artículo *Monitoreo Covid-19 con YOLOv3* [21], el cual es la base del presente trabajo, propone el uso de YOLOv3 y DeepSort²¹. EL artículo *COVID-Robot* [22] propone el uso de YOLOv3 y de un robot móvil. Deepak Birla [2] propone una calibración manual para simular la perspectiva de "vista de pájaro" (de arriba hacia abajo): consiste en seleccionar cuatro puntos que definen "la región de interés", esta propuesta hace uso de YOLOv3.

Existen otros artículos relacionados a la detección del distanciamiento físico, entre los más importantes: [23], [24], [25], [26].

3. Metodología

De acuerdo al numeral 2.1.3, el método de detección de objetos de YOLOv5 utiliza cuadros delimitadores sobre las personas detectadas dentro de la imagen analizada. El proceso base para el marco de trabajo a proponer consiste: eliminar los cuadros delimitadores, los cuales serán reemplazados por coordenadas centrales (centroídes). Con la ayuda de una tercera coordenada de profundidad se

20. KidsHealth: Organización sin fines de lucro, fundada por el MD Neil Izemberg en 1995.

21. DeepSort: Deep Learning to Track Custom Object in a Video.

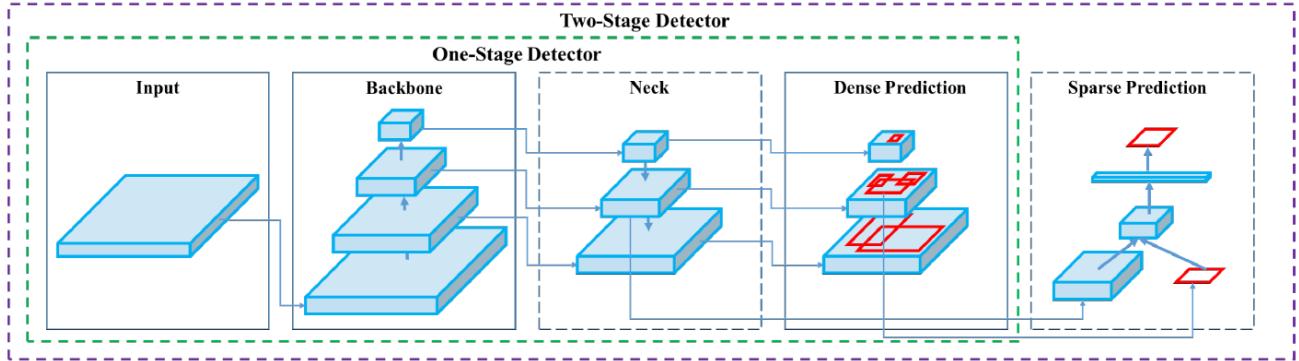


Figure 3. Arquitectura YOLOv5.

logra determinara la “norma $L2$ ²² —tridimensional— entre todas las personas detectadas dentro de la imagen.

La metodología utilizada en el presente trabajo consiste en el siguiente procedimiento:

- **Preprocesamiento:** YOLOv5 dispone con modelos pre-entrenados [27], para detectar 80 clases de objetos, en con el conjunto de datos “COCO²³ train 2017 (117,263 images)”. Se volverá a entrenar YOLOv5 con el conjunto de datos OID (Open Image DataSet), con el objetivo que solo aprenda a detectar personas. Con este fin, se pre-procesará el conjunto de datos OID para generar los datos de entrenamiento y validación en el formato requerido por YOLOv5.
- **Entrenamiento:** luego de descargar la última versión del repositorio de YOLOv5 se procederá al entrenamiento. La red YOLOv5 a entrenar utilizará el modelo “yolov5x” descrita con más detalle en el numeral 3.1.
- **Inferencia:** realizar una rápida prueba de la detección de objetos y su visualización.
- **Distanciamiento Físico:** realizar los cambios en YOLOv5 para que pueda detectar y alertar el correcto distanciamiento físico entre las personas.
- **Resultado:** realizar pruebas de detección del distanciamiento físico en imágenes y videos.

3.1. Arquitectura del Modelo Propuesto

YOLOv5 es un detector de objeto en un sola etapa, la arquitectura completa se visualiza en la figura 3:

El modelo de red seleccionado “yolov5x” esta conformados por 407 capas entre convolucionales y lineales que están distribuidos en tres piezas principales, descritas a continuación:

- 1) **Backbone:** Tiene como objetivo extraer las características más importantes de la imagen de entrada. En YOLOv5 usa la red CSPNet²⁴ [16] como Backbone el cual permite agregar y generar características en diferentes granularidades.
- 2) **Neck:** Son una serie de capas para mezclar y combinar características de la imagen para luego enviarlas a la predicción. En YOLOv5 usa la red

PANet²⁵ [28] para generar, de manera óptima, pirámides de características a diferentes escalas y tamaños.

- 3) **Head:** Consuma todas las características generadas en Neck y realiza los pasos necesarios para la predicción de clases y cajas delimitadoras. Realiza la parte final de la detección de objetos; aplica los *Anchor Boxes* en las características y genera los vectores de salida con la probabilidad de la clase de objeto, puntuación de objeto (si es un objeto) y cuadros delimitadores (bounding boxes).

Otras características de la arquitectura YOLOv5:

- **Función de Activación:** usa las funciones de activación *ReLU* [29] y *Sigmoid* [30].
- **Función de Optimización:** usa SGD [31] o Adam [32], por defecto usa SGD.
- **Función de Perdida:** es una función compuesta por las funciones de pérdida de la puntuación de objeto, el puntaje de la probabilidad de clase de objeto y el puntaje de la regresión del cuadro delimitador, ver la fórmula 1. En el caso especial de la función de pérdida del puntaje de la probabilidad de clase y puntuación objeto, YOLOv5 usa *Binary Cross-Entropy with Logits Loss* [33].
- **Métrica:** usa *mAP* [34] [35] (precisión media de la media), métrica similar a AUC (área bajo de curva).
- **Hiperparámetros:** los hiperparámetros disponibles en YOLOv5 se pueden visualizar en la tabla 3.1.

3.2. Medida de Calidad y Ajuste de Hiperparámetros

Para medir la calidad del modelo generado se utilizará la métrica mAP. La métrica mAP incluye la medición de la precisión en la predicción de la clase de objeto y la ubicación del objeto, basado en IoU de cuadro delimitador original.

No se modificará los hiperparámetros por defecto, debido a que estos valores corresponden a los valores óptimos derivados de los pre-entrenamientos realizados por YOLOv5. Cabe precisar que dichos pre-entrenamientos: $s = \text{small}$, $m = \text{medium}$, $l = \text{large}$ y $x = \text{extra large}$ han sido entrenados por un intervalos de tiempo: 2, 4, 6 y 8 días respectivamente, usando GPU V100 de 16GB.

22. Norma L2: es la distancia euclíadiana o norma euclíadiana

23. Common Object in Context (COCO): gran conjunto de datos de segmentación, detección de objetos y etiquetado.

24. Cross Stage Partial Network (CSPNet): Red neuronal que optimiza el costo de los recursos computacionales.

25. Path Aggregation Network (PANet): Red neuronal que permite la generación de pirámides de características.

TABLE 1. TABLA DE HIPERPARÁMETROS

Hyper	Valor	Descripción
lr0	0.01	Tasa de aprendizaje inicial
lrf	0.2	Tasa de aprendizaje final OneCycleLR ($lr0 * lrf$)
momentum	0.937	momentum para SGD / Adam
weight_decay	0.0005	Caída del peso del optimizador
giou	0.05	Ganancia de la pérdida de predicción de cuadro.
cls	0.5	Ganancia de pérdida de predicción clase.
cls_pw	1.0	Peso Binary Cross Entropy Loss de la clase.
obj	1.0	Ganancia de la pérdida de predicción de objeto.
obj_pw	1.0	Binary Cross Entropy Loss del objeto.
iou_t	0.20	Umbral de entrenamiento de IoU.
anchor_t	4.0	Umbral de Anchor Box múltiple
fl_gamma	0.0	Pérdida focal
hsv_h	0.015	HSV-Hue (ángulo) de la imagen para "Augmentation".
hsv_s	0.7	HSV-S (saturación) de la imagen para "Augmentation".
hsv_v	0.4	HSV-V (brillo) de la imagen para "Augmentation".
degrees	0.0	Rotación de imagen.
translate	0.1	Traducción de imagen.
scale	0.5	Escala de la imagen.
shear	0.0	Corte de la imagen.
perspective	0.0	Perspectiva de la imagen.
flipud	0.0	Imagen volteada hacia arriba-abajo (probabilidad).
fliplr	0.5	Imagen voltear izquierda-derecha (probabilidad).
mixup	0.0	Desorden de la imagen (probabilidad).

3.3. Proceso de Generación del Marco de Trabajo

Para generar el “marco de trabajo” que permita la detección de distanciamiento físico, se seguirá el siguiente proceso, ver la figura 4:

- 1) Pre-procesar el conjunto de datos OID: en este paso primero se descargará el conjunto de imágenes de solo personas, luego se procesará para la generación del conjunto de datos de entrenamiento y validación, y las etiquetas de todas las imágenes.
- 2) Entrenar la red YOLOv5: con el conjunto de datos generados en el paso anterior y los pesos pre-entrenados “yolov5x.pt” (transfer learnig).
- 3) Validar el aprendizaje: probar que modelo generado (best.pt) con un pequeño conjunto de datos. Debe poder visualizarse solo cuadros delimitadores para las personas.
- 4) Realizar ajustes a YOLOv5: con el objetivo que el modelo detecte el distanciamiento físico.
 - a) Cada persona será asociada con una característica espacial tridimensional (x, y, d) , donde (x, y) corresponde a las coordenadas del centroide del cuadro delimitador y d define la profundidad desde un punto de observación (desde la cámara) [25] [36].

$$d = \left(\frac{(2 \times 3.14 \times 180)}{(w + h \times 360)} \times 1000 + 3 \right) \quad (2)$$

Donde w es el ancho y h es el alto del cuadro delimitador de la persona.

- b) Para el conjunto de cuadros delimitadores, se calcula la “norma L2” por cada par de coordenadas (centroides de los cuadros o personas) —vectorización por pares—, dado por la siguiente fórmula:

$$D_F = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (d_2 - d_1)^2} \quad (3)$$

- Donde D_F es el **distanciamiento físico**.
c) La distancia D_F ayudará a determinar si existe el distanciamiento mínimo, de **un metro**, acorde a lo recomendado por OMS (ver el numeral 2.2). Para el cálculo de “un metro” de un par de personas (cuadros) dentro de la imagen se utilizará la siguiente fórmula:

$$U_M = \frac{(h_1 + h_2)}{2 \times \kappa} \quad (4)$$

y

$$D_{min} = \eta \times U_M \quad (5)$$

Donde U_M es la **unidad metro** en píxeles de la imagen, κ es la estatura promedio global de valor 1.6 [37], η en la cantidad en metros de valor uno (1), y D_{min} son los píxeles de la distancia mínima, para esto caso “un metro”.

Si la distancia está por debajo de la distancia mínima $D_F < D_{min}$ se alertará como **Alto** y se tranzará una línea de color **rojo**, si está entre $D_F \in [D_{min}, 1.5 \times D_{min}]$ se alertará como **Bajo** y se tranzará una línea de color **amarillo** y se dejará en **verde** el centroide en caso supere $D_F > 1.5 \times D_{min}$.

- d) El indicador general de distanciamiento I_D ayudará a determinar si es seguro entrar a la zona delimitada por la imagen, se utilizará la siguiente fórmula:

$$I_D = \%aforo \times \%alerta \quad (6)$$

Donde:

$$\%aforo = \frac{\text{total}_{\text{personas}}}{\text{capacidad}_{\text{zona}}} \quad (7)$$

$$\%alerta = \frac{\text{total}_{\text{alto}} + 0.5 \times \text{total}_{\text{bajo}}}{\text{total}_{\text{grupos}}} \quad (8)$$

$$\text{capacidad}_{\text{zona}} = \frac{\text{Area}_{\text{imagen}}}{(\text{altura}_{\text{promedio}} + D_{min})^2} \quad (9)$$

$$\text{total}_{\text{grupos}} = \binom{\text{total}_{\text{personas}}}{2} \quad (10)$$

EL valor de I_D oscilará entre $[0, 2]$, donde 2 = 200% indica que se está incumpliendo el distanciamiento y la zona está encima del aforo²⁶.

26. Aforo:Capacidad total de la zona o local

- 5) Pruebas finales: en este paso se realiza las pruebas en imágenes y videos. También se realiza comparación del modelo pre-entrenado de YOLOv5 y el modelo generado.

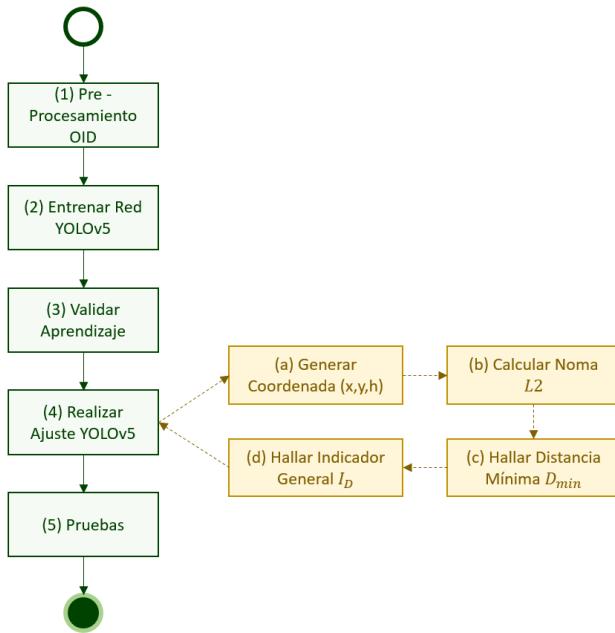


Figure 4. Proceso de Generación del Marco de Trabajo.

4. Experimentación y Resultados

4.1. Entorno

Para el realizar la clasificación binaria (persona o no persona) en la red del modelo YOLOv5 se utilizó Nvidia-SMI GPU, 16GB de memoria como hardware base. Se uso el servicio cloud gratuito de Google, Colab, donde se desarrolló, implementó y probó los Notebooks de Jupyter (cuadernos de trabajo). Junto a dos cuentas de Google Colab y recursos de una máquina local: Nvidia GFORCE GPU de 4GB de memoria, se logró completar todos el proceso descrito en la sección 3.

4.2. Datos

- Conjunto de datos orientados al entrenamiento:
 - Origen:** conjunto de datos obtenidos de forma gratuita del repositorio *Open Image Dataset (OID)* <https://opensource.google/projects/open-images-dataset>. Descargados con la librería *Open Images Downloader* [38].
 - Clase:** descarga de imágenes de la categoría “Person”. Las etiquetas (anotaciones) que corresponden a cada imagen tiene las coordenadas de los cuadros delimitadores de las personas que se encuentran dentro de la imagen.
 - Dimensiones:** la resolución de las imágenes es variada no fue necesario restringir esta variable.
 - Muestra:** el conjunto de datos obtenido fue de 1000 imágenes dividido en muestras de entrenamiento y validación en un ratio de 8:2 (800 y 200).

- Videos de prueba: 10 videos seleccionados, los que fueron descargados del repositorio *Multiple Object Tracking Benbchmark* <https://motchallenge.net/>.
- Imágenes de prueba: 10 imágenes aleatorias descargas de sitios de noticias e imágenes de Google Seach.

Para generar las etiquetas de entrenamiento en el formato requerido de YOLOv5 se realizó las siguientes tareas:

- Generar un archivo *.txt* por imagen.
- Cada objeto que existe en la imagen debe estar representado en una fila dentro del archivo *.txt*.
- Cada fila tiene el formato siguiente: *clase centro_x centro_y ancho alto*. Todos los datos están normalizados entr [0, 1].
- Las imágenes fueron ubicadas dentro de la carpeta “/images/*.jpg” y las etiquetas en la carpeta “/labels/*.txt”. Ejemplo:

```

# Imágenes
..../persons_ds/images/000000109622.jpg
# Etiquetas
..../persons_ds/labels/000000109622.txt
  
```

4.3. Entrenamiento

Para realizar el entrenamiento con YOLOv5; requiere los archivos de configuración del conjunto de datos y de la arquitectura de la red, los que se describen a continuación:

- Configuración de ubicación del conjunto de datos, ver figura 5. Donde *nc* es el número de clases, el valor de *nc* = 1 debido a que solo detecta personas

```

train: ..../persons_ds/images/train/
val: ..../persons_ds/images/val/

# number of classes
nc: 1

# class names
names: ['Person']
  
```

Figure 5. Configuración de conjunto de datos para entrenamiento.

- Configuración de la arquitectura del modelo propuesto, ver figura 6. De acuerdo al numeral 3.1, se considera como modelo base la configuración “yolov5x.yaml” el cual está conformado por 24 submodelos entre BottleneckCSP, Convolucionales, Lineales, otros. El modelo llega a tener 407 capas.

Con esta información entrenamos YOLOv5:

- Transfer Learning:** se utiliza los pesos pre-entrenados “yolov5x.pt”. Esta práctica ayuda a generar mejores resultados con un conjunto pequeño, en la figura 7 se pude visualizar el resultado dela inferencia de personas.
- Redimensionar:** en el procesamiento interno de YOLOv5 las imágenes son escaladas a la dimensión fija de 640x640.

```

# parameters
nc: 1 # number of classes
depth_multiple: 1.33 # model depth multiple
width_multiple: 1.25 # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23] # P3/8
  - [30,61, 62,45, 59,119] # P4/16
  - [116,90, 156,198, 373,326] # P5/32

# YOLOv5 backbone
backbone:
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]], # 0-P1/2
  [-1, 1, Conv, [128, 3, 2]], # 1-P2/4
  [-1, 3, BottleneckCSP, [128]],
  [-1, 1, Conv, [256, 3, 2]], # 3-P3/8
  [-1, 9, BottleneckCSP, [256]],
  [-1, 1, Conv, [512, 3, 2]], # 5-P4/16
  [-1, 9, BottleneckCSP, [512]],
  [-1, 1, Conv, [1024, 3, 2]], # 7-P5/32
  [-1, 1, SPP, [1024, [5, 9, 13]]],
  [-1, 3, BottleneckCSP, [1024, False]], # 9
]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 6], 1, Concat, [1]], # cat backbone P4
  [-1, 3, BottleneckCSP, [512, False]], # 13

  [-1, 1, Conv, [256, 1, 1]],
  [-1, 1, nn.Upsample, [None, 2, 'nearest']],
  [[-1, 4], 1, Concat, [1]], # cat backbone P3
  [-1, 3, BottleneckCSP, [256, False]], # 17 (P3/8-small)

  [-1, 1, Conv, [256, 3, 2]],
  [[-1, 14], 1, Concat, [1]], # cat head P4
  [-1, 3, BottleneckCSP, [512, False]], # 20 (P4/16-medium)

  [-1, 1, Conv, [512, 3, 2]],
  [[-1, 10], 1, Concat, [1]], # cat head P5
  [-1, 3, BottleneckCSP, [1024, False]], # 23 (P5/32-large)

  [[17, 20, 23], 1, Detect, [nc, anchors]], # Detect(P3, P4, P5)
]

```

Figure 6. Configuración de conjunto de datos para entrenamiento.

- Batch & Epoch:** el modelo se entrenó en 30 épocas y 4 lotes. Cabe indicar, que es necesario tener una cuenta *Colab Pro* en caso sea necesario incrementar el número de lotes o épocas. El tiempo de total de entrenamiento fue de $0.838h$.
- Monitorizar:** en el procesamiento interno de YOLOv5, durante la etapa de entrenamiento, el rendimiento del modelo es monitorizado de manera constante usando la métrica *mAP* para la localización de los cuadros delimitadores, la predicción del tipo de clase al que pertenece y la perdida en global en la detección de personas como se indica en la 8 .

4.4. Resultados

El marco propuesto produce (como se muestra en la figura 9 y la figura 10) imágenes (o fotogramas en caso de videos) procesados con las personas detectadas, identificando con un pequeño círculo central (centroide) de color verde: en caso no viole el distanciamiento físico, amarillo: alerta que este entrando en un rango límite del distanciamiento físico y rojo: alerta que está violando el distanciamiento físico. El Indicador general de distanciamiento, visible en un círculo en la parte inferior izquierda de la imagen, muestra en color: verde \rightarrow amarillo \rightarrow rojo, y un porcentaje: 0 \rightarrow 200, que permite identificar si la zona o región que representa la imagen es segura para el transito de las personas.



Figure 7. Configuración de conjunto de datos para entrenamiento.

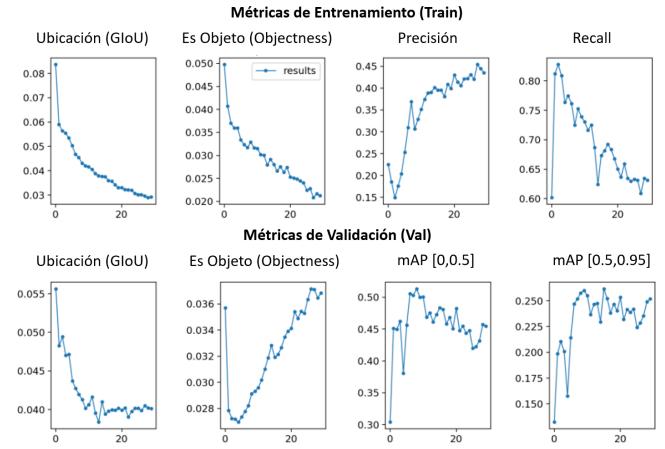


Figure 8. Métricas de Train y Val.

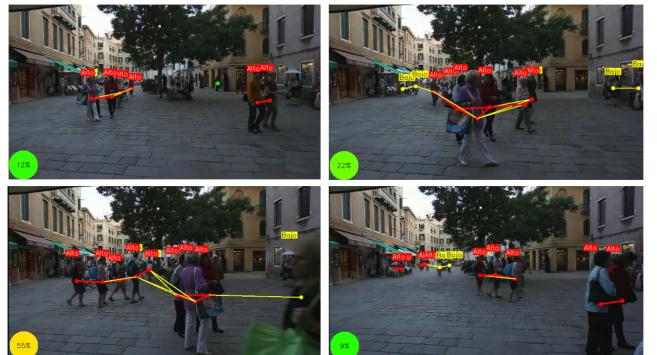


Figure 9. Resultado del marco de trabajo propuesto para la Detección de Distanciamiento Físico vía YOLOv5.

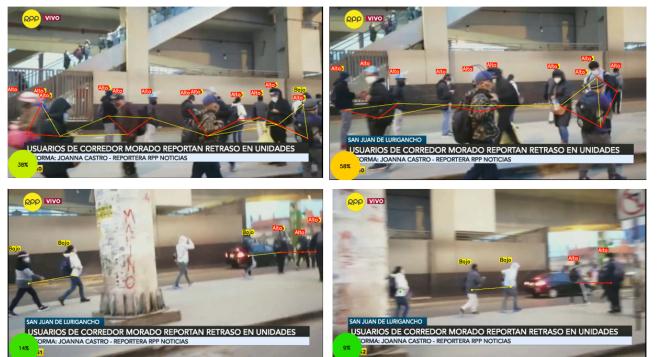


Figure 10. Resultado del marco de trabajo propuesto para la Detección de Distanciamiento Físico vía YOLOv5.

5. Discusión

Los pesos generados en el entrenamiento para la detección de personas, después de comparar las inferencias con los pesos pre-entrenados de YOLOv5 no lograron superarlo, debido al poco entrenamiento realizado. YOLOv5 pre-entrena sus modelos con lotes superiores a 16 y hasta 300 épocas, por varios días y utiliza gran cantidad de GPU. Los pesos pre-entrenados de YOLOv5 detecta 80 clase de objetos, la propuesta en el presente trabajo: es entrenar YOLOv5 en personas. Por este motivo se visualiza que ambos tipos de entrenamiento, la propuesta y de línea base por YOLOv5 no detectan todas las personas de una imagen. En referencia a la detección del distanciamiento físico, en la actualidad existen trabajos similares, pero todas tienen deficiencias en la detección de la distancia, incluso en aquello trabajos donde utilizan cámaras o dispositivos físicos, una mejora importante sería agregar:

- Una red neuronal para detecta puntos de fuga (*vanish point*)
- Entrenamiento para reconocer niños y personas de baja estatura
- Usar técnica de medición de entropía para mejora el indicador general de distanciamiento.

6. Conclusiones y Trabajos Futuros

El presente trabajo propone un marco eficiente basado en Deep Learning en tiempo real para automatizar el proceso de monitoreo del distanciamiento físico a través de enfoques de detección y distancia de personas, donde cada individuo se identifica en tiempo real con la ayuda de cuadros delimitadores, reemplazados por puntos centrales (centroídes). Los centroídes ayudan a identificar los grupos de personas que satisfacen la propiedad de proximidad calculada con la ayuda del enfoque vectorizado por pares (norma L2). El indicador general de distanciamiento se calcula multiplicando el porcentaje del aforo por el porcentaje de violaciones del distanciamiento físico: el cual se calcula con el total de grupos que violan el distanciamiento físico entre el total de grupos generados por las personas detectadas en la imagen.

Se propone como trabajo futuro: asignar un identificador a la personas detectadas y generar estadísticas que permitan realizar análisis estadísticos y poder detectar comportamientos atípicos o anómalos (*p.e. detección de regla*)

References

- [1] W. H. Organization. (2020) Who director-general's opening remarks at the media briefing on covid-19 - 11 march 2020. [Online]. Available: <https://www.who.int/dg/speeches/7>
- [2] D. Birla. (2020) Social distancing ai using python, deep learning and computer vision. [Online]. Available: <https://blog.usejournal.com/social-distancing-ai-using-python-deep-learning-c26b20c9aa4c>
- [3] L. Mao, "Agent-based simulation for weekend-extension strategies to mitigate influenza outbreaks," *BMC public health*, vol. 11, p. 522, 06 2011.
- [4] C. Jarvis, K. Zandvoort, A. Gimma, K. Prem, M. Auzenberg, K. O'Reilly, G. Medley, J. Emery, R. Houben, N. Davies, E. Nightingale, S. Flasche, T. Jombart, J. Hellewell, S. Abbott, J. Munday, N. Bosse, S. Funk, F. Sun, and W. Edmunds, "Quantifying the impact of physical distance measures on the transmission of covid-19 in the uk," *BMC Medicine*, vol. 18, p. 124, 05 2020.
- [5] P. Viola and M. Jones, "Robust real-time object detection," vol. 57, 01 2001.
- [6] C. Papageorgiou, M. Oren, and T. Poggio, "General framework for object detection," vol. 6:, 02 1998, pp. 555 – 562.
- [7] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," *IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2005)*, vol. 2, 06 2005.
- [8] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.
- [9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv 1409.1556*, 09 2014.
- [10] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," 06 2015, pp. 1–9.
- [11] A. Girshick, J. Donahue, T. Darrelland, and J. Malik, "Rich feature hierarchies for object detection and semantic segmentation," *Proc. IEEE Conf. CVPR*, pp. 1–5, 01 2014.
- [12] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," 06 2016, pp. 779–788.
- [13] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," 07 2017, pp. 6517–6525.
- [14] ———, "Yolov3: An incremental improvement," 04 2018.
- [15] A. Bochkovskiy, C.-Y. Wang, and H.-y. Liao, "Yolov4: Optimal speed and accuracy of object detection," 04 2020.
- [16] C.-Y. Wang, H.-y. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "CspNet: A new backbone that can enhance learning capability of cnn," 06 2020, pp. 1571–1580.
- [17] J. Solawetz. (2020) Yolov5 new version - improvements and evaluation. [Online]. Available: <https://blog.roboflow.com/yolov5-improvements-and-evaluation/>
- [18] W. H. Organization. (2020) Coronavirus disease (covid-19) advice for the public. [Online]. Available: <https://www.who.int/emergencies/diseases/novel-coronavirus-2019/advice-for-public>
- [19] J. M. KidsHealth. (2020) Coronavirus (covid-19): Social distancing with children. [Online]. Available: <https://kidshealth.org/en/parents/coronavirus-social-distancing.html?ref=search>
- [20] T. Dbouk and D. Drikakis, "On coughing and airborne droplet transmission to humans," *Physics of Fluids*, vol. 32, p. 053310, 05 2020.
- [21] N. Punn, S. Sonbhadra, and S. Agarwal, "Monitoring covid-19 social distancing with person detection and tracking via fine-tuned yolo v3 and deepsport techniques," 05 2020.
- [22] A. Sathyamoorthy, U. Patel, Y. Savle, M. Paul, and D. Manocha, "Covid-robot: Monitoring social distancing constraints in crowded scenarios," 08 2020.
- [23] V. Bajpai. (2020) Social distancing using yolov5. [Online]. Available: <https://github.com/ChargedMonk/Social-Distancing-using-YOLOv5>
- [24] P. Dwivedi. (2020) Using ai to detect social distancing violations. [Online]. Available: <https://medium.com/swlh/using-ai-to-detect-social-distancing-violations-4707301844be>

- [25] P. Paul. (2020) Object detection and distance measurement. [Online]. Available: <https://github.com/paul-pias/Object-Detection-and-Distance-Measurement>
- [26] A. Anwar. (2020) Using python to monitor social distancing in a public area. [Online]. Available: <https://towardsdatascience.com/monitoring-social-distancing-using-ai-c5b81da44c9f>
- [27] U. LLC. (2020) yolov5. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [28] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, “Path aggregation network for instance segmentation,” 03 2018.
- [29] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” 05 2015.
- [30] Wikipedia. (2020) Sigmoid function. [Online]. Available: https://en.wikipedia.org/wiki/Sigmoid_function
- [31] S. Ruder, “An overview of gradient descent optimization algorithms,” 09 2016.
- [32] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *International Conference on Learning Representations*, 12 2014.
- [33] pytorch.org. (2019) Bcewithlogitsloss. [Online]. Available: <https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html>
- [34] T. Shah. (2018) Measuring object detection models, what is mean average precision? [Online]. Available: <https://cutt.ly/BfW62wA>
- [35] R. J. Tan. (2019) Breaking down mean average precision (map). [Online]. Available: <https://towardsdatascience.com/breaking-down-mean-average-precision-map-ae462f623a52>
- [36] L. Xiaoming, Q. Tian, W. Chen, and Y. Xingliang, “Real-time distance measurement using a modified camera,” 02 2010.
- [37] Wikipedia. (2020) Estatura. [Online]. Available: <https://es.wikipedia.org/wiki/Estatura#:~:text=El%20var%C3%A3o%20adulta%20puede%20medir,55%20a%201%2C65%20m>
- [38] H. Patel. (2018) open images downloader. [Online]. Available: <https://github.com/harshilpatel312/open-images-downloader>