Índice general

v

Índice general

Índice de figuras

VII

Índice de figuras

Índice de tablas

IX

Índice de tablas

TABLA 2.1. Clases detectadas como intruso por el módulo.

Clases detectadas como intruso	
Persona	Vaca
Camión	Bus
Automóvil	Sombrilla
Motocicleta	Mochila
Bicicleta	Maletín
Perro	Maleta
Caballo	Gato

Hay que recordar que la sola detección de una de estas clases generará un reporte positivo de intruso. El modelo no está en capacidad de identificar personas específicas en el vídeo, por lo que le resultará imposible diferenciar entre personal autorizado en un área (el mismo personal de seguridad, o la persona al mando del dron), y a intrusos en la zona.

2.2. Modelo utilizado

El modelo utilizado para el desarrollo de la totalidad del trabajo es YOLOv3. La principal ventaja que tiene este modelo es que es capaz de ejecutarse con una velocidad considerablemente mayor a la de modelos anteriores como RetinaNet-50 y RetinaNet-101.

Este modelo se basa en el uso de redes neuronales convolucionales (CNN) que, al dividir la imagen en una cuadrícula, permiten calcular la probabilidad de ocurrencia de objetos en cada celda. Con esta información, el modelo determina tres parámetros clave para su funcionamiento:

- *Intersection over Union* (IoU) [8], se trata del criterio utilizado para eliminar cajas redundantes, y asegurar que solo se mantenga una por cada objeto detectado, incluso si hay varios objetos cercanos en la imagen. De esta manera, se evita la detección de objetos duplicados y se obtiene una detección más precisa y clara de los objetos presentes en la imagen.

  Se calcula a través de la fórmula 2.1:

$$IoU = \frac{A_s}{A_u} \tag{2.1}$$

  Donde A_s representa el área de superposición entre ambas cajas y A_u el área de unión.

  Para lograr esto, cada celda es responsable de calcular la probabilidad de que haya un objeto dentro de sus fronteras, se genera una serie de cajas propuestas alrededor del objeto, y a través del uso de redes neuronales convolucionales se calcula la probabilidad de ocurrencia.

  Finalmente, el modelo elimina las cajas que no cumplen con dos criterios importantes: en primer lugar, se descartan aquellas cajas que tienen una probabilidad de ocurrencia menor que la máxima del objeto de la zona. Este algoritmo es conocido como *Non-Maximum Suppression* [9] y utiliza la probabilidad de ocurrencia, también conocida como *objectness score* [10], para

8 Capítulo 2. Introducción específica

TABLA 2.1. Clases detectadas como intruso por el módulo.

Clases detectadas como intruso	
Persona	Vaca
Camión	Bus
Automóvil	Sombrilla
Motocicleta	Mochila
Bicicleta	Maletín
Perro	Maleta
Caballo	Gato

Hay que recordar que la sola detección de una de estas clases generará un reporte positivo de intruso. El modelo no está en capacidad de identificar personas específicas en el vídeo, por lo que le resultará imposible diferenciar entre personal autorizado en un área (el mismo personal de seguridad, o la persona al mando del dron), y a intrusos en la zona.

2.2. Modelo utilizado

El modelo utilizado para el desarrollo de la totalidad del trabajo es YOLOv3. La principal ventaja que tiene este modelo es que es capaz de ejecutarse con una velocidad considerablemente mayor a la de modelos anteriores como RetinaNet-50 y RetinaNet-101.

Este modelo se basa en el uso de redes neuronales convolucionales (CNN) que, al dividir la imagen en una cuadrícula, permiten calcular la probabilidad de ocurrencia de objetos en cada celda. Con esta información, el modelo determina tres parámetros clave para su funcionamiento:

- *Intersection over Union* (IoU) [7], se trata del criterio utilizado para eliminar cajas redundantes, y asegurar que solo se mantenga una por cada objeto detectado, incluso si hay varios objetos cercanos en la imagen. De esta manera, se evita la detección de objetos duplicados y se obtiene una detección más precisa y clara de los objetos presentes en la imagen.

  Se calcula a través de la ecuación 2.1:

$$IoU = \frac{A_s}{A_u} \tag{2.1}$$

  Donde A_s representa el área de superposición entre ambas cajas y A_u el área de unión.

  Para lograr esto, cada celda es responsable de calcular la probabilidad de que haya un objeto dentro de sus fronteras, se genera una serie de cajas propuestas alrededor del objeto, y a través del uso de redes neuronales convolucionales se calcula la probabilidad de ocurrencia.

  Finalmente, el modelo elimina las cajas que no cumplen con dos criterios importantes: en primer lugar, se descartan aquellas cajas que tienen una probabilidad de ocurrencia menor que la máxima del objeto de la zona. Este algoritmo es conocido como *Non-Maximum Suppression* [8] y utiliza la probabilidad de ocurrencia, también conocida como *objectness score* [9], para

seleccionar solo las cajas más relevantes. En segundo lugar, se eliminan las cajas que tienen un *Intersection Over Union (IoU)* mayor a un valor determinado, que suele ser 0,45 aunque también se puede ajustar como parámetro en el modelo.

- *Offset*: el *offset* se refiere a la distancia medida desde la esquina superior izquierda de la celda con mayor probabilidad de contener un objeto [11]. Esta medida es una forma eficiente de representar la posición de las cajas y reduce significativamente la cantidad de cálculos necesarios, lo que aumenta la eficiencia computacional del modelo [12]. Una vez obtenido el *offset*, se utilizan estos valores para calcular las coordenadas de las cajas que no fueron descartadas.
- Tensor: hace referencia a un vector utilizado para representar la probabilidad [13] de que cada uno de los objetos incluidos en el *dataset* COCO [6] se encuentre en la caja . Esta representación no solo permite determinar la posición de diferentes objetos en la imagen, sino también discriminarlos por clases. La clase con mayor probabilidad es entonces reportada por el modelo.

Es importante tener en cuenta que estos no son los parámetros finales arrojados por el modelo, sino que más adelante, la red calcula las coordenadas de la caja con la que se rodeará al objeto. Es decir, para cada objeto detectado, el modelo va a devolver (en ese orden):

- pc: *objectness score*, la probabilidad de que haya un objeto.
- bx: coordenada x del centro de la caja propuesta.
- by: coordenada y del centro de la caja propuesta.
- bh: altura de la caja propuesta.
- bw: ancho de la caja propuesta.
- c: tensor con la probabilidad de ocurrencia de las 80 clases que forman parte del *dataset* COCO.

Esto se logra a través del uso de *Anchor Boxes*: cajas de predicción predefinidas que el modelo va refinando a medida que analiza la imagen y calcula las métricas de cada uno de los objetos encontrados. A través de este método es posible eliminar la necesidad de una "ventana deslizante", lo que a su vez permite analizar la totalidad de la imagen en un tiempo mucho menor que los métodos basados en aquel método, y, en consecuencia, se hace posible llevar a cabo un análisis en tiempo real.

Dadas las bondades explicadas anteriormente, en conjunto con su facilidad de implementación en Python, se escogió partir del modelo YoloV3 y modificar su código fuente para lograr generar y extraer la información pertinente para completar el propósito del trabajo. Cabe aclarar que la decisión de utilizar este modelo se vio reforzada una vez se ejecutó la primera versión en tanto que el modelo está optimizado para funcionar con la tarjeta gráfica (NVIDIA Titan X), el código se pudo ejecutar satisfactoriamente, sin dejar a un lado los requisitos del trabajo en CPU (AMD Ryzen 5).

2.2. Modelo utilizado 9

seleccionar solo las cajas más relevantes. En segundo lugar, se eliminan las cajas que tienen un *Intersection Over Union (IoU)* mayor a un valor determinado, que suele ser 0,45 aunque también se puede ajustar como parámetro en el modelo.

- *Offset*: el *offset* se refiere a la distancia medida desde la esquina superior izquierda de la celda con mayor probabilidad de contener un objeto [10]. Esta medida es una forma eficiente de representar la posición de las cajas y reduce significativamente la cantidad de cálculos necesarios, lo que aumenta la eficiencia computacional del modelo [11]. Una vez obtenido el *offset*, se utilizan estos valores para calcular las coordenadas de las cajas que no fueron descartadas.
- Tensor: hace referencia a un vector utilizado para representar la probabilidad [12] de que cada uno de los objetos incluidos en el *dataset* COCO [6] se encuentre en la caja . Esta representación no solo permite determinar la posición de diferentes objetos en la imagen, sino también discriminarlos por clases. La clase con mayor probabilidad es entonces reportada por el modelo.

Es importante tener en cuenta que estos no son los parámetros finales arrojados por el modelo, sino que más adelante, la red calcula las coordenadas de la caja con la que se rodeará al objeto. Es decir, para cada objeto detectado, el modelo va a devolver (en ese orden):

- pc: *objectness score*, la probabilidad de que haya un objeto.
- bx: coordenada x del centro de la caja propuesta.
- by: coordenada y del centro de la caja propuesta.
- bh: altura de la caja propuesta.
- bw: ancho de la caja propuesta.
- c: tensor con la probabilidad de ocurrencia de las 80 clases que forman parte del *dataset* COCO.

Esto se logra a través del uso de *Anchor Boxes*: cajas de predicción predefinidas que el modelo va refinando a medida que analiza la imagen y calcula las métricas de cada uno de los objetos encontrados. A través de este método es posible eliminar la necesidad de una "ventana deslizante", lo que a su vez permite analizar la totalidad de la imagen en un tiempo mucho menor que los métodos basados en aquel método, y, en consecuencia, se hace posible llevar a cabo un análisis en tiempo real.

Dadas las bondades explicadas anteriormente, en conjunto con su facilidad de implementación en Python, se escogió partir del modelo YoloV3 y modificar su código fuente para lograr generar y extraer la información pertinente para completar el propósito del trabajo. Cabe aclarar que la decisión de utilizar este modelo se vio reforzada una vez se ejecutó la primera versión en tanto que el modelo está optimizado para funcionar con la tarjeta gráfica (NVIDIA Titan X), el código se pudo ejecutar satisfactoriamente, sin dejar a un lado los requisitos del trabajo en CPU (AMD Ryzen 5).

del alcance del trabajo y por lo tanto, cuya implementación es necesaria a través de este método mientras que hacerlo a través de software integrado en el código fuente (como FFMpeg) será parte del alcance de trabajos posteriores.

El paquete de Python ya viene instalado por defecto en la instalación de Ubuntu, por lo que no se requerirá su instalación. El resto de los paquetes, sin embargo, deberán ser instalados de acuerdo con el manual de instalación que se entrega al cliente, en el que se detalla de manera minuciosa cuál es el procedimiento que se debe seguir a fin de instalar correctamente todos los paquetes.

De igual manera, se debe tener en cuenta que dadas las restricciones de la máquina en la que se desarrolló el trabajo, la totalidad del desarrollo fue llevado a cabo en máquinas virtuales a través del uso de programas de virtualización como VMware Workstation [14]. Dentro de los requerimientos mínimos de hardware del trabajo, se tienen los siguientes:

- Procesador: AMD Ryzen 5
- Tarjeta gráfica: no requerida
- Memoria RAM: 6 GB.

Es importante tener en cuenta que los listados anteriormente son los requisitos mínimos del sistema, que van a garantizar que el trabajo se ejecute cumpliendo con los requerimientos del cliente. Sin embargo, se recomiendan las siguientes características del sistema para un rendimiento óptimo a la hora de ejecutar el módulo.

- Procesador: Intel Core i7
- Tarjeta gráfica: NVidia Tesla T4 o NVidia Titan X.
- Memoria RAM: 8 GB.

El modelo YoloV3, sobre el que está basado el desarrollo del trabajo utiliza de manera extensiva la API Cuda de NVIDIA. Por este motivo, tratar de ejecutar el módulo en tarjetas gráficas de AMD resultará en el modelo siendo llevado al procesador por Tensorflow. En consecuencia, se afirmará que el trabajo resulta incompatible con estas tarjetas gráficas. Esto fue comprobado con el uso de una tarjeta AMD Radeon Vega 8.

12 Capítulo 2. Introducción específica

del alcance del trabajo y por lo tanto, cuya implementación es necesaria a través de este método mientras que hacerlo a través de software integrado en el código fuente (como FFMpeg) será parte del alcance de trabajos posteriores.

El paquete de Python ya viene instalado por defecto en la instalación de Ubuntu, por lo que no se requerirá su instalación. El resto de los paquetes, sin embargo, deberán ser instalados de acuerdo con el manual de instalación que se entrega al cliente, en el que se detalla de manera minuciosa cuál es el procedimiento que se debe seguir a fin de instalar correctamente todos los paquetes.

De igual manera, se debe tener en cuenta que dadas las restricciones de la máquina en la que se desarrolló el trabajo, la totalidad del desarrollo fue llevado a cabo en máquinas virtuales a través del uso de programas de virtualización como VMware Workstation [13]. Dentro de los requerimientos mínimos de hardware del trabajo, se tienen los siguientes:

- Procesador: AMD Ryzen 5
- Tarjeta gráfica: no requerida
- Memoria RAM: 6 GB.

Es importante tener en cuenta que los listados anteriormente son los requisitos mínimos del sistema, que van a garantizar que el trabajo se ejecute cumpliendo con los requerimientos del cliente. Sin embargo, se recomiendan las siguientes características del sistema para un rendimiento óptimo a la hora de ejecutar el módulo.

- Procesador: Intel Core i7
- Tarjeta gráfica: NVidia Tesla T4 o NVidia Titan X.
- Memoria RAM: 8 GB.

El modelo YoloV3, sobre el que está basado el desarrollo del trabajo utiliza de manera extensiva la API Cuda de NVIDIA. Por este motivo, tratar de ejecutar el módulo en tarjetas gráficas de AMD resultará en el modelo siendo llevado al procesador por Tensorflow. En consecuencia, se afirmará que el trabajo resulta incompatible con estas tarjetas gráficas. Esto fue comprobado con el uso de una tarjeta AMD Radeon Vega 8.

Capítulo 3

Diseño e implementación

En este capítulo se detallan los aspectos técnicos del desarrollo del trabajo. Incluye las consideraciones tomadas en cuenta durante el desarrollo, el detalle de las modificaciones realizadas al modelo de partida, así como el detalle del papel que juega el módulo en el sistema. De igual manera, se hace una descripción cronológica del desarrollo del módulo.

3.1. Consideraciones generales

El módulo de inteligencia artificial propuesto es una implementación del modelo YOLOv3, especialmente modificada para cumplir con las necesidades específicas del cliente, así como una serie de requisitos funcionales. Es por esto que fue necesario hacer modificaciones a los archivos originales de la biblioteca.

(Esta sección aún no se ha terminado)

3.2. Arquitectura del proyecto

3.3. Esquema del módulo

3.4. Esquema de modificaciones

3.5. Ajustes para cumplir con el rendimiento

3.6. Entrega de resultados al resto del software

13

Capítulo 3

Diseño e implementación

En este capítulo se detallan los aspectos técnicos del desarrollo del trabajo. Incluye las consideraciones tomadas en cuenta durante el desarrollo, el detalle de las modificaciones realizadas al modelo de partida, así como el detalle del papel que juega el módulo en el sistema. De igual manera, se hace una descripción cronológica del desarrollo del módulo.

3.1. Consideraciones generales

El módulo de inteligencia artificial propuesto es una implementación del modelo YOLOv3, especialmente modificada para cumplir con las necesidades específicas del cliente, así como una serie de requisitos funcionales. Es por esto que fue necesario hacer modificaciones a los archivos originales de la biblioteca.

Como se puede observar en el capítulo 1, no todos los requerimientos funcionales propuestos para el inicio del trabajo. Recordando que esta versión del trabajo representa un mínimo producto viable, a continuación se presenta una lista de todos los requerimientos funcionales que fueron cubiertos:

- el módulo se ejecuta en un computador de uso de hogar.
- el módulo es capaz de reconocer n clases como intruso.
- se reporta al usuario de manera visual.
- se reporta al usuario a través de un archivo de fácil análisis.
- resulta muy sencillo iniciar la ejecución del módulo a través de su interfaz grafica.

De igual manera, el módulo está construido de manera que realizar modificaciones, o bien, expandir su funcionalidad en un futuro resulte muy sencillo. Es de vital importancia recalcar que esto se hizo teniendo en cuenta que se trata de un módulo que, a fin de salir de este estatus de mínimo producto viable, deberá sufrir un proceso de optimizacion muy fuerte. Algunos ejemplos de puntos en los que se puede aplicar estos trabajos son:

- el módulo está diseñado para filtrar las etiquetas que no hacen parte de las clases dadas en la definición de intruso. Esto implica que, a pesar de que no se reporte en ninguna de las dos metodologías, el módulo aún reconoce otras clases. En otras palabras, si se puede observar un modelo en el *frame*, este será satisfactoriamente detectado y son los métodos encargados del reporte los que no lo entregarán como un resultado positivo.

- la precisión del modelo debe ser mejorada: dado que ocasionalmente se hacen detecciones de objetos que no corresponden con lo visto en el *frame*. Se detectó que, en particular, el módulo es especialmente susceptible a generar falsos positivos.
- los cuadros se devuelven en espacio de color bgr. Esto, en combinación con que el color de los rectángulos no se puede cambiar desde la interfaz grafica.
- el color de los rectángulos es igual para todas las clases de objeto.
- se debe mejorar significativamente el manejo que da el módulo de las excepciones que se generen, tanto a nivel de manejo de problemas originados en el código mismo, como recuperación cuando se generen cortes en ma recepcion del *streaming* de video.
- a pesar de que la interferir grafica se realizó pensando en la facilidad de uso para el usuario, su comportamiento podría resultar extraño dado que el diseño de experiencia de usuario (UX) quedó fuera del alcance de los trabajos durante su etapa de concepción.

Finalmente, resulta de especial interés tener en mente también las limitaciones del modelo:

- los modelos YOLO se ven limitados cuando se les pide detectar objetos que se encuentran muy cerca de una fuente de luz muy brillante, especialmente el sol, se puede generar dificultades para detectar los objetos. Esto es parcialmente contrarrestando añadiendo las clases adicionales detalladas en la tabla n.
- los objetos deben encontrarse a una distancia prudencial del dron. A pesar de que mas pruebas de vuelo no se han enfocado en el cálculo de este valor, sí se hace evidente que al estar el dron demasiado lejos de ellas, no se hace una detección oportuna.
- el modelo no cuenta con funciones de *auto-healing*, que deben ser tenidas en cuenta a fin de lograr mantener la integridad del modelo si sus pesos son actualizados constantemente por el cliente a fin de mejorar la predicción.

3.2. Arquitectura del módulo

Se propone como mínimo producto viable de este módulo, un código altamente modular. Esto significa que la manera en la que está desarrollado (a excepción del método principal, en el archivo Utils), está planteada en pequeños métodos que podrán ser modificados y mantenidos con facilidad. Esto supone, sin embargo, que sea de vital importancia entender la arquitectura del módulo a fin de poder realizar las modificaciones pertinentes en el punto adecuado del código. Así como se hizo en el literal anterior con el flujo de información entre el Sistema y el Módulo, a continuación, se describe el flujo de la información en el interior del módulo.

El módulo está compuesto por cuatro capas funcionales:

1. funciones originales de YOLO: encargadas de la funcionalidad base de detección de los objetos en la totalidad de las clases que componen el *dataset*

Capítulo 4

Ensayos y resultados

A pesar de que esto se estipuló en la concepción inicial del proyecto, conforme se dio su avance no se hizo posible llevar a cabo un proceso formal de pruebas al módulo. Sin embargo, el cliente estuvo involucrado directamente en el proceso de verificación de la funcionalidad del módulo a través de reuniones quincenales.

4.1. Descripción del proceso de pruebas

A pesar de que esto se estipuló en la concepción inicial del proyecto, conforme se dio su avance no se hizo posible llevar a cabo un proceso formal de pruebas al módulo. Sin embargo, el cliente estuvo involucrado directamente en el proceso de verificación de la funcionalidad del módulo a través de reuniones quincenales.

4.2. Pruebas en Raspberry Pi

Desde la concepción inicial del proyecto, se estableció que se tendría dos caminos para poder desarrollarlo: Por un lado, se planteó que el módulo debía ser capaz de reconocer intrusos en varios streamings de vídeo simultáneos (camino que finalmente se vio favorecido), o bien, hacer que el módulo estuviese en capacidad de correr en equipos a bordo de los drones. Para esto, se hizo una serie de pruebas en equipos Raspberry Pi. Más concretamente, las especificaciones de los equipos sobre los que se hicieron las pruebas son:

- Modelo: Raspberry Pi
- Sistema Operativo: Raspberry Pi OS (Debian 11 – Bullseye)
- CPU quad-core ARM Cortex-A7 900MHz
- Memoria ram: 1 Gb.
- Overclocking: Desactivado

Concretamente, el modelo con el que se tuvo los problemas más importantes, especialmente a la hora de la instralación fue la versión completa de Tensorflow, requerida para la ejecución del módulo. Cabe mencionar que las placas Raspberry Pi son compatibles con la versión Lite de Tensorflow. Dicho esto, se intentaron 5 métodos para la instalación de esta biblioteca. Ninguno de ellos fue exitoso. El proceso llevado a cabo en cada una de las cinco instalaciones fue el siguiente:

- Máquina 1: Aquí irá una descripción de la instalación de la máquina
- Máquina 2

COCO. Incluye funciones como nms (encargada del *non-max supression*) y la función IoU.

2. funciones híbridas: normalmente funciones qué coordinan el llamado de las demás funciones. A pesar de que ofrecen una funcionalidad muy similar a las originales de YOLO, funcionan en un orden diferente, de forma que se puedan obtener los datos requeridos en el momento preciso para las necesidades del cliente.
3. funciones propias: Que permiten agregar al módulo la funcionalidad requerida por el cliente. Ejemplos de estas funciones incluyen la generación de archivos en formato CSV y la carga de etiquetas válidas.
4. interfaz gráfica: Que permite al usuario interactuar con el módulo para iniciar su ejecución fácilmente.

El diagrama de boques correspondiente a estas cuatro capas es el siguiente:

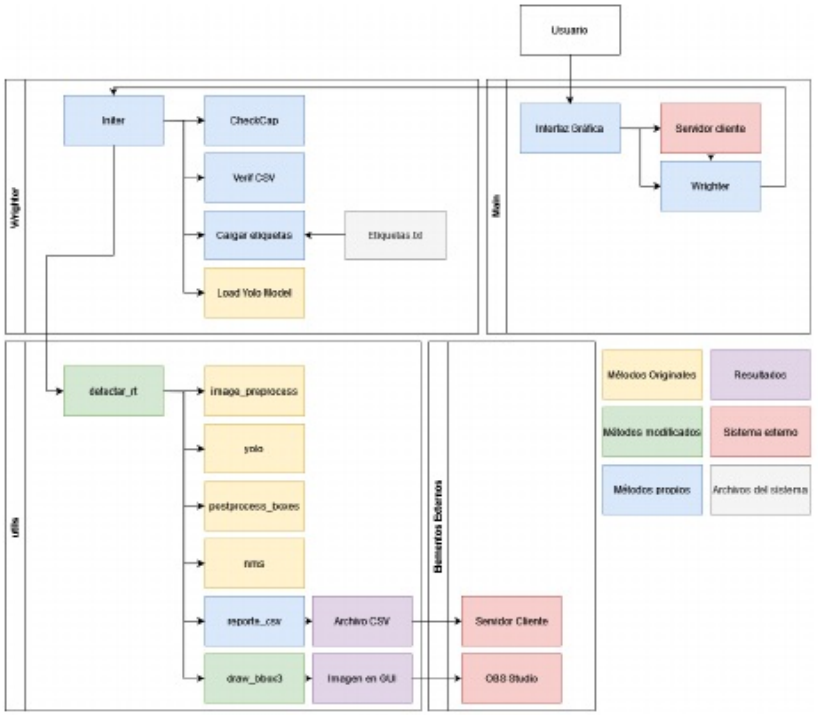

FIGURA 3.1. Diagrama de bloques del módulo.

- Máquina 3
- Máquina 4
- Máquina 5

Es importante destacar que, dado que el módulo se desarrolló tomando como base el modelo YOLOv3 original (no el modelo Lite), no se cumplía con las especificaciones requeridas por este tipo de equipos, y una segunda implementación del modelo utilizando la versión YOLOv3 Lite hubiese implicado en sí, el trabajo equivalente a un proyecto completo adicional.

De igual manera atrae particular atención mencionar que la placa con la que se contaba era un modelo antiguo de Raspberry Pi (de febrero de 2015), que no cuenta con una compatibilidad tan completa como la que se da con una Raspberry Pi 4. Es, entonces, de vital importancia que, para iteraciones posteriores del desarrollo del trabajo, y en caso de que se decida continuar con el desarrollo de módulos a bordo de los drones, considerar placas más avanzadas, como Raspberry Pi 4.

4.3. Caso de Uso

3.3. Esquema detallado del módulo en el interior del sistema

Este módulo es una parte de un trabajo más grande desarrollado directamente por el cliente. Es decir, deberá estar en capacidad de comunicarse con otras partes del sistema, de forma que estos otros módulos puedan tomar decisiones y recibir órdenes de controladores humanos. A pesar de que se desconoce el funcionamiento de la totalidad del sistema.

Para lograr esto, durante la fase de concepción del trabajo se estipuló que el módulo deberá ser capaz de conectarse a los servidores del cliente, en los que se aloja la transmisión de vídeo. Esta conexión entre ambos equipos se logra cuando el usuario indica al módulo, a través de la interfaz gráfica (visible en la imagen). Una vez se presiona el botón "Iniciar ejecución". Esta interfaz está pensada para ser tan sencilla como sea posible, permitiendo que el usuario inicie la ejecución del módulo de la manera más sencilla posible.

FIGURA 3.2. Interfaz gráfica del módulo.

Por otro lado, el módulo deberá reportar al resto del sistema si se detectó a algún intruso. Para esto, se definieron dos metodologías, que serán tocadas más en detalle más adelante en este capítulo:

1. reporte visual
2. reporte en formato CSV

Dada esta descripción, desde afuera, el módulo tendrá el siguiente aspecto:

Cabe mencionar que, dada la naturaleza confidencial de este módulo, así como de la totalidad del sistema, se buscó que el módulo requiriera la menor cantidad de información para funcionar como fuese posible. Esto también permite que los parámetros dados del vídeo y la cantidad de transmisiones que se procesan a la vez se encuentren bajo total control del cliente.

De igual manera es de vital importancia tener en cuenta que la totalidad del módulo se desarrolló en total desconocimiento del funcionamiento del resto del sistema. Es por esto que para iniciar su ejecución se debe ejecutar el código de manera manual. El sistema no puede detectar cuándo se ha iniciado una transmisión, y por lo tanto, se requiere una persona encargada de dar inicio cuando sea necesario.

Finalmente, este módulo está diseñado como un mínimo producto viable que permita a las personas tomar decisiones informadas, con soporte en vídeo. Esta versión no es de ninguna manera un sistema autónomo, sino que constituye un sistema de apoyo a la decisión.

Capítulo 5

Conclusiones

5.1. Resultados Obtenidos

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Tiempos de ejecución

Acá se indica cómo se podría continuar el trabajo más adelante.

5.3. Monitoreo de resultados

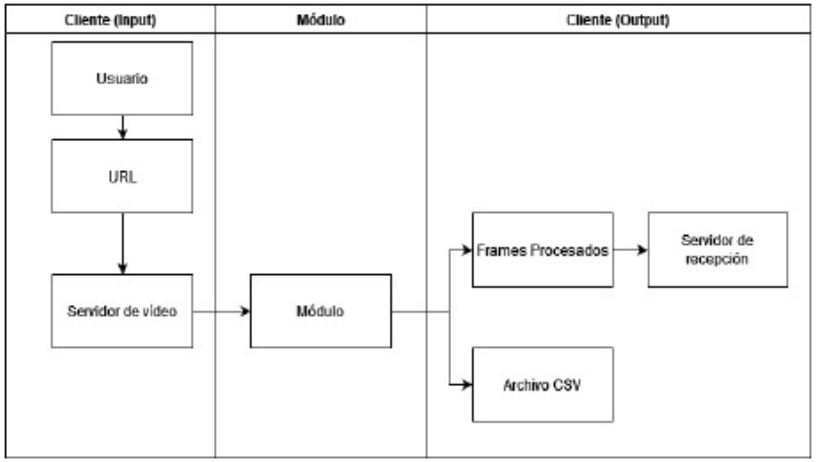

FIGURA 3.3. Diagrama de bloques del módulo en el interior del sistema.

3.4. Esquema detallado de las modificaciones hechas al modelo

Dado que se trata de una implementación personalizada, existen varias diferencias entre el módulo desarrollado y la implementación original del modelo YOLOv3. Dicho esto, las modificaciones realizadas generales al modelo, en orden cronológico son las siguientes:

- se preparó el código para que se ejecute desde un *streaming* de vídeo, en vez de un archivo local en el equipo. Esto implica que se debe detener la ejecución antes de pasar imágenes al método de detección si no es posible establecer conexión con el servidor.
- se filtraron las etiquetas a fin de lograr que sólo sean detectadas las clases definidas en la definición de intruso.
- se crearon los métodos encargados de generar una respuesta que se pasará al resto del módulo. Particularmente, se le permitió al programa generar dos tipos de respuesta.
- se estableció un método que permite descartar un número determinado de *frames*, de forma que se permita aliviar la carga computacional sobre el equipo.
- durante el desarrollo del módulo se estableció que el hacer que el módulo sea capaz de procesar varios *streamings* de vídeo de manera paralela está fuera del alcance de las temáticas cubiertas en el postgrado, a fin de cubrir satisfactoriamente este requisito, se informó al cliente que sería posible este análisis en caso de que se reciba el *streaming* con varias cámaras combinadas

en un único cuadro. Esto, por ser una decisión relacionada con la forma en la que el cliente maneja su *streaming*, no afectó el desarrollo del trabajo.

Cabe resaltar que cada una de estas modificaciones implicó una serie de tareas más pequeñas, a las que Manta Beach, como cliente, hizo un seguimiento cercano.

3.5. Ajustes para cumplir con el rendimiento requerido

Dadas las limitantes del equipo en el que se realizó el desarrollo del módulo (especialmente teniendo en cuenta que este desarrollo se hizo utilizando máquinas virtuales), fue necesario llevar a cabo un análisis detallado que, en teniendo en mente las necesidades del cliente, permitiera reducir la carga computacional sobre la máquina. De esta manera se intentó una serie de ajustes al modelo de forma que se lograra mantener los requisitos dados en el proceso de establecimiento de los trabajos:

- eliminación de un número determinado de *frames*. Si bien, este es un número fácil de establecer, y que el sistema recibirá como parámetro, se recomienda utilizar un valor de N=10, de forma que, para un fps de 30, se analicen 3 cuadros por segundo. Cabe resaltar que 30 será el número máximo para cumplir con el requisito de hacer un análisis en un tiempo no mayor a 1 segundo, mientras que con N=120, se cumplirá el requisito de hacer un análisis en un tiempo no mayor a 4 segundos.
- reducción de la calidad del *streaming* de vídeo a 200. A pesar de que esta reducción afecta muy notoriamente la calidad del vídeo, el módulo es capaz de detectar objetos. Se pierden, sin embargo, las propiedades probatorias del trabajo. En la imagen se puede ver un ejemplo del vídeo con calidad reducida. Como se mencionó en secciones anteriores este es un parámetro qué el módulo ya recibe en el *streaming* de vídeo, y cuyo control recae únicamente en el cliente.

Finalmente, la solución escogida fue descartar uno de cada N *frames*, de forma que no se procese la totalidad de ellos, y se requiera una mejor cantidad de operaciones. Cabe mencionar que, en este tipo de modelos, es común que conforme se aumente el rendimiento, también se reducirá la precisión del modelo. Es por esto que, para contrarrestar estos efectos, se decidió incluir algunas clases que permitieran hacer una detección indirecta. Estas clases corresponden a objetos que las personas cargan con frecuencia. Estas clases adicionales son:

TABLA 3.1. Clases auxiliares para detección indirecta.

Clases auxiliares	
Camión	Bicicleta
Automóvil	Bus
Motocicleta	Sombrilla
Mochila	Maletín

Dada la naturaleza del trabajo, cabe aclarar que, en este punto, el módulo no podrá identificar discernir que al detectar simultáneamente la clase mochila y la clase persona en un espacio muy cercano, podrá tratarse de una única detección. Estas serán registradas y reportadas por el módulo como dos detecciones separadas.

Apéndice A

Tablas extendidas

TABLA A.1: Software requerido para el desarrollo y la ejecución del módulo

Paquete	Tipo	Licencia	Versión
PIP	Gestor de bibliotecas	MIT	22.0.2
Numpy	Biblioteca de Python	BSD	1.21.5
OpenCV	Biblioteca de Python	Apache 2	4.5.4
Matplotlib	Biblioteca de Python	BSD	3.6.2
Tensorflow	Biblioteca de Python	Apache 2	2.10.1
Pycharm	Software completo	Apache 2	2022.3.3 Community edition
OBS Studio	Software completo	GPLv2	28.1.2 (64 Bit)
Python	Lenguaje de programación	Python Software Foundation License Version 2	3.10.6
Ubuntu	Sistema operativo	Creative Commons CC-BY-SA version 3.0 UK	22.04.1 LTS (64 Bit)

Esto, sin embargo, es preferible a eliminar detecciones, teniendo en mente que se prefiere que el módulo pueda detectar a tantas personas como sea posible, dada la sensibilidad de las consecuencias en caso de una falla del modelo.

3.6. Entrega de resultados obtenidos al resto del software y retransmisión del vídeo procesado

Una vez se ha analizado el *frame*, el módulo deberá estar en capacidad de devolver resultados al resto del sistema, de forma que tanto las personas encargadas, como el sistema mismo estén en capacidad de tomar decisiones rápidas, e informadas de cómo proceder. Dados estos dos actores a los que se les debe hacer llegar la información, en conjunto con el cliente se decidió utilizar dos métodos diferentes a través de los que el módulo dará su respuesta para cada uno de los *frames*:

- entrega visual: Imagen procesada con sus cajas identificando los objetos detectados. Estos *frames* ya procesados son retransmitidos al servidor original para su visualización en la plataforma designada por el cliente. Es importante tener en cuenta que, en esta versión del módulo, esta retransmisión no se hace directamente a través del módulo propuesto, sino que se requerirá el uso de software externo, que en esta ocasión es OBS Studio.
- entrega para análisis de datos: Archivo en formato CSV con la información de los objetos detectados. De igual manera, el sistema estará en capacidad de reportar el número del dron desde el que se hizo la detección, a pesar de que es una característica que no se ha probado, teniendo en cuenta que aún no se ha hecho vuelos con varios drones de manera simultánea. A diferencia de la imagen preprocesada, este archivo no es retransmitido. Se guarda por defecto en la carpeta de documentos de la maquina en la que se está ejecutando. Dado que esta versión corresponde a un mínimo producto viable, este archivo no se retransmite. Se guarda, en cambio, en la carpeta de documentos dr la máquina en la que se ejecuta el módulo.

Es de vital importancia tener en cuenta, sin embargo, que como se mencionó en el capítulo 1, no se espera que haya una persona encargada constantemente de la revisión de las cámaras, y en consecuencia, es posible deshabilitar la retransmisión del cuadro preprocesado. Por otro lado, y buscando que el sistema sea tan autónomo como resulte posible, no se podrá apagar el reporte a través de archivos JSON.

Por otro lado, los archivos JSON deberán contener los siguientes datos:

- fecha y hora de detección.
- coordenadas de detección dentro del *frame*.
- número de dron en el que ocurre la detección.

Los datos serán entregados en el orden en el que han sido mencionados. Esto es de vital importancia de forma que el resto del sistema sea capaz de interpretar correctamente los datos entregados. Por otro lado, se debe recordar que el sistema desconoce la ubicación del dron en su recorrido de vuelo. Es por esto que el módulo no podrá reportar la ubicación en la que detectó el objeto. Esta deberá ser

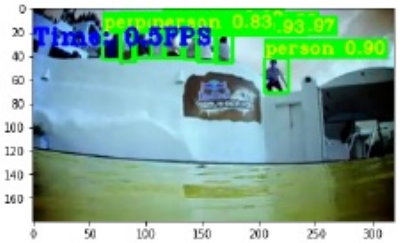

FIGURA 3.4. Ejemplo de resultados generados por el módulo.

calculada por el sistema a partir de los datos de posición, altura y posición en el *frame* (información proporcionada por el módulo).

Bibliografía

[1] Drone AI Services. *Drones de seguridad privada ¿Cómo funcionan?* Último acceso el 28 de marzo de 2023. 2020. URL: https://www.droneservices.com.ar/industria-4-0/drones-de-seguridad-privada/.

[2] Jordan Henrio y Tomoharu Nakashima. «Anomaly Detection in Videos Recorded by Drones in a Surveillance Context». En: oct. de 2018, págs. 2503-2508. DOI: 10.1109/SMC.2018.00429.

[3] Joseph Redmon y col. «You Only Look Once: Unified, Real-Time Object Detection». En: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[4] Joseph Redmon y Ali Farhadi. «YOLOv3: An Incremental Improvement». En: *CoRR* abs/1804.02767 (2018). arXiv: 1804.02767. URL: http://arxiv.org/abs/1804.02767.

[5] Heng Zhao y col. «Mixed YOLOv3-LITE: A Lightweight Real-Time Object Detection Method». En: *Sensors (Basel)* 20.7 (2020), pág. 1861. ISSN: 1424-8220. DOI: 10.3390/s20071861. URL: https://doi.org/10.3390/s20071861.

[6] Common Objects in Context. *COCO*. Último acceso el 28 de marzo de 2023. URL: https://cocodataset.org/#home.

[7] Tae-Hoon Yong. *Automatic diagnosis for cysts and tumors of both jaws on panoramic radiographs using a deep convolution neural network*. Último acceso el 11 de abril de 2023. 2020. URL: https://www.researchgate.net/profile/Tae-Hoon-Yong/publication/342402762/figure/fig2/AS:957322327257088@1605254837269/A-CNN-architecture-modified-from-YOLOv3-with-the-modified-layers-in-Bold-CNN.ppm.

[8] Sairaj Neelam. *YOLO for Object Detection, Architecture Explained!* Último acceso el 11 de abril de 2023. 2021. URL: https://medium.com/analytics-vidhya/understanding-yolo-and-implementing-yolov3-for-object-detection-5f1f748cc63a.

[9] Sambasivarao. K. *Non-maximum Suppression (NMS)*. Último acceso el 11 de abril de 2023. 2019. URL: https://towardsdatascience.com/non-maximum-suppression-nms-93ce178e177c.

[10] Uri Almog. *YOLO V3 Explained*. Último acceso el 11 de abril de 2023. 2020. URL: https://towardsdatascience.com/yolo-v3-explained-ff5b850390f.

[11] Weidong Kuang. *Fundamentals of deep learning: a step-by-step guide Chapter 12 Object Detection*. Último acceso el 11 de abril de 2023. 2023. URL: https://faculty.utrgv.edu/weidong.kuang/book/chapter12%20object%20detection.pdf.

[12] Manish Chablani. *YOLO — You only look once, real time object detection explained*. Último acceso el 11 de abril de 2023. 2017. URL: https://towardsdatascience.com/yolo-you-only-look-once-real-time-object-detection-explained-492dc9230006.

Capítulo 4

Ensayos y resultados

Este capítulo detalla las distintas pruebas que se hicieron sobre el módulo, así como el proceso de descarte de módulos a bordo de los drones. De igual manera, detalla los resultados obtenidos durante estos ensayos y los criterios utilizados para dar por cerrados los distintos requerimientos presentados en el capítulo 1. Finalmente, se explican los resultados mostrados durante una ejecución exitosa del módulo.

4.1. Descripción del proceso de pruebas

A pesar de que esto se estipuló en la concepción inicial del trabajo, conforme se dio su avance no se hizo posible llevar a cabo un proceso formal de pruebas al módulo. Sin embargo, el cliente estuvo involucrado directamente en el proceso de verificación de la funcionalidad del módulo a través de reuniones quincenales.

En estas reuniones fue posible hacer un seguimiento muy cercano y dar aprobación a las siguientes características para el funcionamiento del módulo. Durante estas reuniones se hizo posible dar por aprobado el código presentado para el trabajo, teniendo en cuenta que se cumplió con los siguientes criterios:

- el módulo obtiene correctamente los *frames* enviados por el cliente en el servidor que este dispone para tal fin.
- el módulo detecta intrusos en un tiempo no mayor a 4 segundos, aún cuando se ejecute en sistemas considerablemente menores a los que se espera lo ejecuten. En particular, utilizando CPU en vez de GPU durante la detección.
- la precisión del modelo es razonable.
- fue posible hacer la detección en tiempo real. Es decir, no solo se ejecutó con vídeos de prueba grabados por el cliente, sino que el módulo también se ejecutó correctamente durante vuelos transmitidos en tiempo real.
- el módulo es capaz de detectar intrusos en varios *streamings* de vídeo. Se deja la anotación que para que esto se logre, se deberá preprocesar el vídeo de forma que todos los *streamings* vengan combinados en un único *frame*. El módulo no tiene la capacidad de ejecutar varias instancias de manera paralela.
- se realizó una serie de pruebas durante estas reuniones, en las que se evidenció el funcionamiento del módulo.

[13] Jonathan Hui. *Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3*. Último acceso el 11 de abril de 2023. 2018. URL: https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088.

[14] VMware. *VMware Workstation Player*. Último acceso el 11 de abril de 2023. 2023. URL: https://www.vmware.com/products/workstation-player.html.

Para este visto bueno, no fueron considerados los siguientes requisitos, a pesar de que se trata de tareas que fueron ejecutadas:

- documentación del módulo.
- facilidad de uso del módulo.
- sistema operativo en el que se ejecutará el módulo. A pesar de que, durante la etapa de planeación, no se especificó el sistema operativo en el que se ejecutará el modelo, este fue diseñado para ejecutarse en Ubuntu 22.04.

4.2. Pruebas en Raspberry Pi

Desde la concepción inicial del trabajo, se estableció que se tendrían dos caminos para poder desarrollarlo: Por un lado, se planteó que el módulo debía ser capaz de reconocer intrusos en varios *streamings* de vídeo simultáneos (camino que finalmente se vio favorecido), o bien, hacer que el módulo fuese capaz de correr en equipos a bordo de los drones. Para esto, se llevaron a cabo una serie de pruebas y ensayos en equipos Raspberry Pi. Más concretamente, las especificaciones de los equipos sobre los que se hicieron las pruebas son:

- modelo: Raspberry Pi 2 modelo B
- sistema Operativo: Raspberry Pi OS (Debian 11 – Bullseye)
- CPU quad-core ARM Cortex-A7 900MHz
- memoria ram: 1 Gb.
- overclocking: Desactivado

Específicamente, el modelo con el que se tuvo los problemas más importantes, especialmente a la hora de la instalación fue la versión completa de Tensorflow, requerida para la ejecución del módulo. Cabe mencionar que las placas Raspberry Pi son compatibles con la versión Lite de Tensorflow. Dicho esto, se intentaron cinco métodos para la instalación de esta biblioteca. Ninguno de ellos fue exitoso. El proceso llevado a cabo en cada una de las cinco instalaciones fue el siguiente:

- instalación 1 [14]:
  - instalación de Tensorflow: utilizando PIP.
  - se clona el repositorio de GIT de Tensorflow.
- instalación 2 [15]:
  - instalación de Tensorflow: utilizando PIP y el comando –no-cache-dir.
- instalación 3 [16]:
  - instalación de Tensorflow Lite: utilizando PIP.
- instalación 4 [17]:
  - instalación de Libatlas: utilizando PIP.
  - instalación de Tensorflow: utilizando PIP.
- instalación 5 [18]: