



ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

DETECCIÓN Y CLASIFICACIÓN DE OBJETOS USANDO LA ARQUITECTURA YOLOV4

SOBRE IMÁGENES RGB MÁS UN CANAL ADICIONAL

PROYECTO INTEGRADOR

Previo a la obtención del Título de:

INGENIERÍA EN COMPUTACIÓN

Presentado por:

Axel Antonio Auza Aspiazu

Luis Antonio Carrasco Medina

Guayaquil - Ecuador

2021

AGRADECIMIENTOS

En primer lugar, queremos agradecer a Dios, por permitirnos llegar a este punto de nuestra vida profesional.

También agradecer a ESPOL por la educación tanto dentro y fuera del aula de clase, así como el CIDIS por brindarnos los recursos y herramientas necesarias para este proyecto.

Por último, agradecer al PhD. Boris Vintimilla por sus enseñanzas, paciencia y esfuerzo por obtener lo mejor, al PhD Federico Bonini por se guía al comienzo del sendero de la vida universitaria y a la MSc. Gladys Carrillo por las experiencias laborales compartidas.

DECLARACIÓN EXPRESA

“La responsabilidad del contenido de este Proyecto de Grado, nos corresponde conforme al reglamento de propiedad intelectual de la institución; Axel Auza y Luis Carrasco damos nuestro consentimiento para que la ESPOL realice la comunicación pública de la obra por cualquier medio con el fin de promover la consulta, difusión y uso público de la propiedad intelectual”



Axel Auza



Luis Carrasco

EVALUADORES

PhD. Boris Vintimilla

.....
PhD. Boris Vintimilla
PROFESOR DE LA MATERIA

PhD. Boris Vintimilla

.....
PhD. Boris Vintimilla
PROFESOR TUTOR



.....
PhD. Segio Velastin
PROFESOR CO-TUTOR

RESUMEN

El presente proyecto se centra en la modificación de una arquitectura de CNN conocida como YOLOv4, la cual funciona como detector y clasificador de objetos en imágenes, pero solo trabaja con imágenes RGB (3 canales). Por lo que se pretende modificar esta arquitectura para que trabaje sobre imágenes de más de tres canales, a esta nueva arquitectura la llamaremos YOLOv4_4C. El propósito del presente proyecto es modificar la arquitectura YOLOv4 para que pueda trabajar sobre imágenes de más de tres canales otorgándole más *features* a la red CNN en escenarios en donde las imágenes RGB no son suficientes para la detección y clasificación de objetos. El contenido de este documento inicialmente presenta un capítulo con la teoría y bases necesarias para realizar este proyecto, además de la motivación de este. Luego pasamos a nuestra propuesta de metodología en donde encontramos que el proyecto está compuesto por cuatro componentes llamados: bases de datos, pre-procesamiento, detección y clasificación de objetos sobre imágenes de cuatro canales, validación y pruebas en donde se explica que se realizó y donde se realizó para lograr que dicha arquitectura logre procesar imágenes de más de tres canales. A continuación, se muestran los resultados y análisis del tercer componente en donde se explica con gráficas los porcentajes de precisiones que alcanza el nuevo modelo YOLOv4_4C propuesto. Para esto, se analizan 4 escenarios para la validación de la arquitectura modificada (YOLOv4_4C), que luego son comparados con otros 4 escenarios usando la arquitectura original (YOLOv4). Cabe recalcar, que esta validación se realiza por medio de una comparativa por pares, dicha comparativa hace un breve análisis de cómo se comportan estas arquitecturas ante estos escenarios. Finalmente, se describen las conclusiones y recomendaciones que se obtuvieron luego de realizar la implementación de la arquitectura modificada y como cambia el comportamiento de esta al utilizar imágenes con más de tres canales. Se debe destacar que de los resultados obtenidos con YOLOv4_4C muestran una mejoría del 32.57% de precisión usando técnicas de *data*

augmentation e imágenes de cuatro canales, en donde el cuarto canal es la componente térmica (*LWIR*) en comparación con la arquitectura YOLOv3 y un 39.41% de precisión con YOLOv4.

Palabras Clave: YOLOv4, CNN, Bases de Datos, Imágenes de cuatro canales, Imágenes RGB, Imágenes térmicas, Detección de Objetos, Clasificación de Objetos.

ABSTRACT

This project focuses on modifying a CNN architecture known as YOLOv4, which works as a detector and classifier of objects in images, but only works with RGB images (3 channels). Therefore, it is intended to modify this architecture so that it works on images with more than three channels, we will call this new architecture YOLOv4_4C. The purpose of this project is to modify the YOLOv4 architecture so that it can work on images with more than three channels, granting more features to the CNN network in scenarios where RGB images are not sufficient for the detection and classification of objects. The content of this document initially presents a chapter with the theory and bases necessary to carry out this project, in addition to its motivation. Then we move on to our proposed methodology where we find that the project is composed of four components called: databases, pre-processing, detection and classification of objects on four-channel images, validation and tests where it is explained what was carried out and where it was carried out to achieve that said architecture can process images of more than three channels. Next, the results and analysis of the third component are shown, where the percentages of accuracies reached by the proposed new YOLOv4_4C model are explained with graphs. For this, 4 scenarios for the validation of the modified architecture (YOLOv4_4C) are analyzed, which are then compared with 4 other scenarios using the original architecture (YOLOv4). It's worth say, this validation is carried out by means of a comparison by pairs, this comparison makes a brief analysis of how these architectures behave in these scenarios. Finally, the conclusions and recommendations obtained after implementing the modified architecture are described and how its behavior changes when using images with more than three channels. It should be noted that the results obtained with YOLOv4_4C show a 21.21% improvement in precision using transfer learning, data augmentation techniques and four-channel images, where the fourth channel is the thermal component (LWIR) compared to the YOLOv3 architecture. and 38.05% accuracy with YOLOv4. It describes the conclusions and recommendations that were obtained after implementing the modified architecture and how its behavior changes when using images with more than three channels. It should be noted that the results obtained with YOLOv4_4C show a 21.21% improvement in precision using transfer learning, data augmentation techniques and four-channel images, where the fourth channel is the thermal component (LWIR) compared to the YOLOv3 architecture. and 38.05% accuracy with YOLOv4. It describes the conclusions and recommendations that were obtained after implementing the modified architecture and how its behavior changes when using images with more than three channels. It should be noted that the

results obtained with YOLOv4_4C show a 32.57% improvement in precision using data augmentation techniques and four-channel images, where the fourth channel is the thermal component (LWIR) compared to the YOLOv3 architecture. and 39.41% accuracy with YOLOv4.

Keywords: YOLOv4, CNN, Databases, Four Channel Images, RGB Images, Thermal Images, Object Detection, Object Classification.

ÍNDICE GENERAL

| | |
|--|-----|
| RESUMEN..... | I |
| ABSTRACT | III |
| ÍNDICE GENERAL | V |
| ÍNDICE FIGURAS | VII |
| ÍNDICE TABLAS | IX |
| CAPÍTULO 1 | 1 |
| 1. Introducción..... | 1 |
| 1.1 Descripción del problema..... | 2 |
| 1.2 Justificación del problema..... | 3 |
| 1.3 Objetivos..... | 5 |
| 1.3.1 Objetivo general | 5 |
| 1.3.2 Objetivos específicos..... | 5 |
| 1.4 Marco teórico | 5 |
| 1.4.1 Detección de objetos | 6 |
| 1.4.2 Clasificación de objetos | 8 |
| CAPÍTULO 2 | 15 |
| 2. Metodología..... | 15 |
| 2.1 Bases de datos | 15 |
| 2.1.1 Generación de la BD 3 canales y la BD 3 canales aumentada..... | 17 |
| 2.1.2 Generación de la BD 4 canales y la BD 4 canales aumentada..... | 18 |
| 2.2 Pre-Procesamiento..... | 20 |
| 2.2.1 Transformación de anotaciones sobre imágenes | 21 |
| 2.2.2 Generación de imágenes de cuatro canales..... | 23 |
| 2.3 Detección y clasificación de objetos sobre imágenes de cuatro canales | 24 |
| 2.3.1 Lectura del código fuente de YOLOv4 | 24 |

| | | |
|------------------|---|----|
| 2.3.2 | Modificación del código fuente para leer imágenes de cuatro canales | 25 |
| 2.4 | Validación y pruebas | 26 |
| 2.4.1 | Validación..... | 26 |
| 2.4.2 | Pruebas | 27 |
| CAPÍTULO 3 | | 29 |
| 3. | Resultados y análisis del componente de detección y clasificación de objetos sobre imágenes de cuatro canales | 29 |
| 3.1 | Escenario 1: Imágenes RGB (Día) | 30 |
| 3.2 | Escenario 2: Imágenes RGB + <i>Thermal</i> (Día)..... | 31 |
| 3.3 | Comparativa de imágenes RGB (Día) e imágenes RGB + <i>Thermal</i> (Día) | 32 |
| 3.4 | Escenario 3: Imágenes RGB + <i>data augmentation</i> (Día)..... | 33 |
| 3.5 | Escenario 4: Imágenes RGB + <i>Thermal</i> + <i>data augmentation</i> (Día) | 34 |
| 3.6 | Comparativa de imágenes RGB + <i>data augmentation</i> (Día) e imágenes RGB + <i>Thermal</i> + <i>data augmentation</i> (Día) | 35 |
| 3.7 | Escenario 5: Imágenes RGB (Noche) | 35 |
| 3.8 | Escenario 6: Imágenes RGB + <i>Thermal</i> (Noche) | 36 |
| 3.9 | Comparativa de imágenes RGB (Noche) e imágenes RGB + <i>Thermal</i> (Noche) | 37 |
| 3.10 | Escenario 7: Imágenes RGB + <i>data augmentation</i> (Noche)..... | 38 |
| 3.11 | Escenario 8: Imágenes RGB + <i>Thermal</i> + <i>data augmentation</i> (Noche)..... | 38 |
| 3.12 | Comparativa de imágenes RGB + <i>data augmentation</i> (Noche) e imágenes RGB + <i>Thermal</i> + <i>data augmentation</i> (Noche) | 40 |
| CAPÍTULO 4 | | 42 |
| 4. | Conclusiones y recomendaciones | 42 |
| 4.1 | Conclusiones | 42 |
| 4.2 | Recomendaciones | 43 |

ÍNDICE FIGURAS

| | |
|---|----|
| Figura 1.1 Ejemplo de pares de imágenes (RGB y <i>Thermal</i>) de una misma escena del <i>dataset</i> de KAIST [9] | 3 |
| Figura 1.2 Imagen ejemplo basado en la forma (izquierda) y basado en la apariencia (derecha) [11] | 6 |
| Figura 1.3 Imagen ejemplo basado en la extracción de fondo (izquierda) y basado en la diferencia de imágenes (derecha) [11] | 7 |
| Figura 1.4 Cross mini Batch Normalization (CBN) [8] | 9 |
| Figura 1.5 Path Aggregation Network (PAN) [8] | 9 |
| Figura 1.6 iPhone 6 [21] | 10 |
| Figura 1.7 iPhone 12 [22] | 10 |
| Figura 1.8 RedEdge-MX MicaSense [23] | 11 |
| Figura 1.9 Pares de imágenes: (arriba) NIR, (medio) imagen colorizada por la red DCGAN, (abajo) imágenes RGB reales [12] | 12 |
| Figura 1.10 Pares de imágenes: (arriba) NIR, (medio) RGB, (abajo) NDVI creado a partir de imágenes NIR y RGB [15] | 13 |
| Figura 1.11 Imagen tomada del dron [13] | 13 |
| Figura 1.12 Mosaico completo CIR [16] | 14 |
| Figura 2.1 Ejemplo de imágenes del dataset de KAIST. (Izquierda) RGB, (Derecha) LWIR de la misma escena [9] | 16 |
| Figura 2.2 Esquema de la generación de las 4 bases de datos a usarse en el proyecto | 16 |
| Figura 2.3 Ejemplo de una imagen RGB del <i>dataset</i> KAIST [9] | 17 |
| Figura 2.4 Ejemplos de imágenes RGB usando data augmentation | 18 |
| Figura 2.5 Ejemplo de imagen RGB usando data augmentation mosaic | 18 |
| Figura 2.6 Proceso de creación de imágenes de 4 canales – BD 4 canales | 19 |
| Figura 2.7 Ejemplo de imágenes cuatro canales usando <i>data augmentation</i> | 19 |
| Figura 2.8 Ejemplo de imágenes cuatro canales usando data augmentation <i>mosaic</i> | 20 |
| Figura 2.9 Proceso de transformación de anotaciones al formato YOLOV4 | 20 |
| Figura 2.10 Ejemplo de etiquetado de personas en una imagen RGB | 21 |
| Figura 2.11 Ejemplo de oclusión en imagen RGB de la base de datos KAIST [9] | 22 |
| Figura 2.12 Representación visual del formato de anotación de la arquitectura actual de YOLOv4 [9] | 23 |

| | |
|--|----|
| Figura 2.13 Proceso de creación de imagen de 4 canales | 23 |
| Figura 2.14 Archivo antes de la modificación para lectura de imágenes tres canales | 25 |
| Figura 2.15 Archivo después de la modificación para lectura de imágenes de más de tres canales | 26 |
| Figura 3.1 Escenarios de entrenamiento para ser evaluados con YOLOv4 y YOLOv4_4C | 30 |
| Figura 3.2 Entrenamiento de imágenes RGB (Día) utilizando la arquitectura de YOLOv4 | 31 |
| Figura 3.3 Entrenamiento de imágenes RGB + <i>Thermal</i> (Día) utilizando la arquitectura de YOLOv4_4C | 32 |
| Figura 3.4 Entrenamiento de imágenes RGB + <i>data augmentation</i> (Día) utilizando la arquitectura de YOLOv4..... | 33 |
| Figura 3.5 Entrenamiento de imágenes RGB + <i>Thermal</i> + <i>data augmentation</i> (Día) utilizando la arquitectura de YOLOv4..... | 34 |
| Figura 3.6 Entrenamiento de imágenes RGB (Noche) utilizando la arquitectura de YOLOv4 | 36 |
| Figura 3.7 Entrenamiento de imágenes RGB + <i>Thermal</i> (Noche) utilizando la arquitectura de YOLOv4_4C | 37 |
| Figura 3.8 Entrenamiento de imágenes RGB + <i>data augmentation</i> (Noche) utilizando la arquitectura de YOLOv4..... | 38 |
| Figura 3.9 Entrenamiento de imágenes RGB + <i>Thermal</i> + <i>data augmentation</i> (Noche) utilizando la arquitectura de YOLOv4_4c | 40 |

ÍNDICE TABLAS

| | |
|--|----|
| Tabla 3.1 Comparación de imágenes RGB (Día) y RGB + <i>Thermal</i> (Día) | 32 |
| Tabla 3.2 Comparación RGB + <i>data augmentation</i> (Día) y RGB + <i>Thermal</i> + <i>data augmentation</i> (Día) | 35 |
| Tabla 3.3 Comparación RGB (Noche) y RGB + <i>Thermal</i> (Noche)..... | 37 |
| Tabla 3.4 Comparación de imágenes RGB + <i>data augmentation</i> (Noche) y RGB + <i>Thermal</i> + <i>data augmentation</i> (Noche) | 41 |

CAPÍTULO 1

Este capítulo inicialmente presenta una breve introducción al problema de detección de objetos sobre imágenes, seguidamente describe el problema relacionado con la temática del proyecto propuesto con el propósito de entrar en contexto con el mismo. Luego, pasamos a la justificación para conocer porque se propone desarrollar el proyecto, después encontraremos los objetivos que motivan el proyecto para finalizar con una base teórica que permite soportar el trabajo a realizar.

1. INTRODUCCIÓN

La detección de objetos sobre imágenes, que se desarrolla a partir de la inteligencia artificial, es una metodología que posee múltiples usos dentro de la vida diaria. Por ejemplo, algunas de las aplicaciones que toman ventaja de su uso son: el control de tráfico, el reconocimiento facial en sistemas de vigilancia, la clasificación de objetos en imágenes, entre otros. Las implementaciones convencionales de los sistemas de detección de objetos obligan a realizar configuraciones en la nube, clústeres de procesamiento de imágenes que requieren una gran capacidad computacional.

La mayoría de las implementaciones de detección de objetos sobre imágenes utiliza modelos basados en *Redes Neuronales Convolucionales* (CNN), que son esenciales para obtener un balance tanto en precisión como en eficiencia. La mejora de estos detectores de objetos permite reducir el costo computacional, así como mejorar su predicción. Para ello, los modelos de detección utilizan una mayor cantidad de características (*features*) presentes en la imagen, lo que les permite identificar un objeto similar a como lo realiza el ojo humano. Sin embargo, estos modelos CNN por lo general están limitados a trabajar sobre imágenes del espectro visible con máximo tres canales y no permiten trabajar con un número mayor de canales, como es el caso de YOLOv4 cuya detección de objetos lo realiza sobre imágenes de tres canales RGB como lo describen los autores Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao en su arquitectura [1].

Cabe mencionar que YOLOv4 es un detector y clasificador de objetos basados en CNN, que se encarga de tareas de detección y clasificación de objetos en imágenes, usando la arquitectura

propuesta por los autores, en donde recibe imágenes a color (RGB) con sus respectivas anotaciones para crear un modelo el cual puede identificar en otras imágenes donde se encuentra el objeto.

Nuestra propuesta de proyecto se centra en modificar la actual arquitectura de YOLOv4, que trabaja sobre imágenes de 3 canales, para que trabaje sobre imágenes de 4 canales. Para esto, inicialmente añadiremos un canal adicional a los datos de entrada (es decir, generaremos imágenes de 4 canales a partir de las imágenes de entrada) lo cual proporcionará nuevas características a los datos de entrada con el propósito de obtener un detector de objetos con una mayor precisión y con mejores resultados. Además, basados en la idea de Alex García y Peter König [5] sobre el uso de modelos entrenados con un aumento de datos (*data augmentation*) donde concluyen que esta técnica aumenta la robustez de los modelos frente a la variabilidad de la entrada sin reducir su efectividad, este proyecto propone también el uso de aumento de datos para su propósito.

1.1 Descripción del problema

En los últimos años se han propuesto una gran cantidad de implementaciones que pueden usarse para la detección y clasificación de objetos que hacen uso de CNN y mucho más reciente se ha propuesto el uso de modelos neuronales basados en *Transformers* [6]. Del mismo modo, existen también implementaciones convencionales o clásicas que trabajan directamente sobre los datos de la imagen sin el uso de CNN o *Transformers* tales como las técnicas de *Super Vector Machine (SVM)*, *Random Forest*, entre otras. Estas técnicas tradicionales representaban un arduo trabajo para el investigador ya que eran computacionalmente muy costoso por la cantidad de recursos que consumía, además era primordial tener un conjunto de datos (*dataset*) muy grande y bien anotado. Pero sus resultados no eran los óptimos porque no se obtenía una precisión alta [7].

En este proyecto el problema de la detección y clasificación de objetos se basa en el uso de la arquitectura YOLOv4, el cual es una CNN que trabaja sobre imágenes RGB de tres canales. Esto es una limitante ya que existen características presentes en la escena de la imagen que se podrían aprovechar si se usan imágenes de más de tres canales, las cuales pueden ser generadas combinando por ejemplo imágenes del espectro visible con imágenes de otros espectros.

Para esto, este proyecto propone modificar el código original de la arquitectura YOLOv4 para que permita recibir como entrada imágenes con cuatro canales (en lugar de imágenes de tres

canales), se busca con esto incrementar la precisión en tareas de detección y clasificación de objetos. El proyecto propone usar como base de datos el conjunto de datos públicos provistos por KAIST Multispectral Pedestrian dataset [9] cuyos autores son Soonmin Hwang, Jaesik Park, Namil Kim y Yukyung Choi, el cual consta de pares de imágenes de una misma escena (imágenes RGB e imágenes térmicas) de personas tal como se muestra en Figura 1.1. Estos pares de imágenes pueden ser aprovechados para generar imágenes de cuatro canales incluyendo información del espectro visible (RGB) y térmicas (LWIR) [9]. En la sección 2.2.3 se describe el procedimiento propuesto para generar este tipo de imágenes.

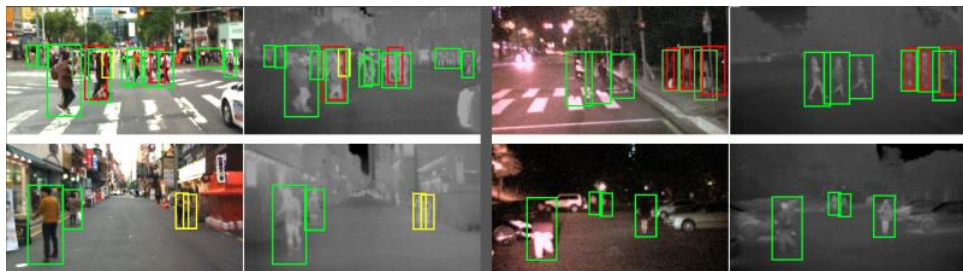


Figura 1.1 Ejemplo de pares de imágenes (RGB y *Thermal*) de una misma escena del *dataset* de KAIST [9]

Una vez modificado la arquitectura de YOLOv4 este será evaluado para medir el comportamiento en las tareas de detección y clasificación de objetos sobre las imágenes de cuatro canales.

1.2 Justificación del problema

En la actualidad la detección y clasificación de objetos juega un papel muy importante en varios sectores tales como en: alimentos, supermercados, tránsito vehicular, video vigilancia, entre otros. Por ejemplo, en el sector de supermercados donde se puede encontrar una gran variabilidad tanto de productos como de marcas se estudian los retos que presenta la detección y clasificación de objetos en este entorno.

Por otro lado, existen implementaciones de modelos de arquitecturas de redes tales como *Faster R-CNN*, *Single Shot MultiBox Detector* (SSD), YOLOv2 [3] que permiten detectar y clasificar objetos de forma automática. No obstante, la precisión de estos modelos se ve afectada cuando existe una gran variabilidad de objetos en la escena de las imágenes que estos procesan.

En [3] se evalúan los resultados obtenidos sobre el conjunto de datos (imágenes de tres canales) MS COCO con 80 clases donde *Faster R-CNN* obtuvo una precisión del 42.7%, SSD una precisión del 46.5% y YOLOv2 un 44.0%. Se observa que a medida que la variabilidad o cantidad de objetos en la imagen incrementa, el detector empieza a perder precisión en comparación con el conjunto de datos con menor cantidad de clases. [8] analiza como la precisión de la arquitectura de YOLOv4 se ve afectada cuando el tamaño de la imagen a procesar varía, reportando que sobre la base de datos de MS COCO se obtuvo que para imágenes de tamaño 416x416 una precisión en promedio del 41.2%, mientras que para imágenes de tamaño 512x512 una precisión del 43.0%, mientras que para imágenes de 608x608 que son las más recomendables a usar, se logra una precisión del 43.5%.

Considerando que la precisión generada por los modelos se ve afectada cuando la cantidad de objetos a identificar es considerablemente grande o el tamaño de la imagen a procesar varía, la precisión podría mejorarse aumentando la cantidad de información contenida en la imagen. Una forma de hacer esto sería usando imágenes de más de tres canales, esto es imágenes de 4 canales.

En la actualidad, la aparición de nuevas tecnologías para la adquisición de imágenes con más de tres canales permite obtener una mayor cantidad de características de la escena a procesar. Estos nuevos tipos de datos incluyen una mayor cantidad de características que podrían ser aprovechadas por las actuales arquitecturas que no las han considerado logrando obtener una mejor precisión en su labor de detección y clasificación al no estar limitadas únicamente al uso de imágenes 3 canales. Sin embargo, el disponer de imágenes de más de tres canales es esencial para obtener más características presentes en la escena de la imagen lo cual permitirá a las arquitecturas que se basan en el uso de CNN mejorar su precisión en tareas de detección y clasificación de objetos. Además, la mayoría de las arquitecturas han sido implementadas para usar solo imágenes de tres canales, sin considerar otros canales que pueden contener información importante al momento de realizar el entrenamiento de la red.

1.3 Objetivos

Los objetivos del proyecto generales y específicos se describen a continuación:

1.3.1 Objetivo general

Adaptar la arquitectura de YOLOv4 para que trabaje sobre imágenes RGB más un canal (imágenes de 4 canales) permitiendo la detección y clasificación de objetos.

1.3.2 Objetivos específicos

1. Generar una base de datos de imágenes de 4 canales (RGB + un canal) a partir de la base de datos de KAIST [9].
2. Aplicar técnicas de aumento de datos sobre la base de datos de KAIST y el conjunto de datos de imágenes de 4 canales generado previamente.
3. Adaptar el funcionamiento de YOLOv4 para que trabaje con imágenes de cuatro canales, esto generará una nueva arquitectura YOLOv4_4C.
4. Evaluar la arquitectura de YOLOv4 y YOLOv4_4C sobre las bases de datos generadas en los objetivos específicos 1 y 2.

1.4 Marco teórico

Empezaremos definiendo dos términos importantes para la investigación de este proyecto, como son la detección y clasificación de objetos que son dos tareas claves para comprender la imagen. Por un lado, la tarea de detección de objetos tiene como objetivo predecir la ubicación del objeto dentro de la imagen, mientras que la tarea de clasificación de objetos se encarga de predecir la existencia de determinados objetos dentro de la imagen [10]. Luego de conocer los términos principales que definen el proyecto, conoceremos las técnicas actuales que existen tanto para la detección como para la clasificación que se dividen en dos grandes grupos respectivamente [11], como se describe a continuación:

1.4.1 Detección de objetos

Según la literatura las técnicas de detección de objetos sobre imágenes se clasifican en: detección de objetos basado en características y detección de objetos basado en movimientos.

1. Basado en características

Esta técnica se basa en características de forma y apariencia de la imagen, se encuentra dividida en dos grandes grupos que se describen abajo.

1.1. Basado en la forma:

La detección de objetos basados en la forma depende significativamente de la detección de fondo, entre los elementos que se utilizan tenemos puntos, líneas, rectángulos, circunferencia y contornos. Donde cada una cumple una función específica como, por ejemplo: los puntos pueden ser usados para rastrear objetos.

1.2. Basado en la apariencia:

Este tipo de detección se realiza mediante varias secciones que se combinan, utilizando un modelo como una combinación de unas cuantas líneas para determinar donde se encuentra un objeto.

En la Figura 1.2 se muestran ejemplos de la detección de objetos basado en la forma y en la apariencia.



Figura 1.2 Imagen ejemplo basado en la forma (izquierda) y basado en la apariencia (derecha) [11]

2. Basado en movimiento

El método basado en movimiento se centra en el contexto de la imagen, es decir, características de fondo y diferencias entre imágenes como se describe posteriormente:

2.1. Extracción de fondo:

Este método tradicional y universal para detectar objetos que se encuentran en movimiento en una secuencia de imágenes, en donde solo se enmarca el objetivo para detectar si existe una diferencia entre la imagen anterior y la actual.

2.2. Diferencia de imágenes:

La diferencia de imágenes es parecida a la anteriormente descrita, excepto por el fondo que no es excluido de la imagen, donde se puede analizar el movimiento de puntos que sería el objeto para detectar si existe movimiento.

En la Figura 1.3 se muestran ejemplos de la detección de objetos basado en la extracción de fondo y en la diferencia de imágenes.



Figura 1.3 Imagen ejemplo basado en la extracción de fondo (izquierda) y basado en la diferencia de imágenes (derecha) [11]

1.4.2 Clasificación de objetos

En la literatura de clasificación de objetos las técnicas de clasificación de objetos sobre imágenes se dividen en: aprendizaje supervisado y aprendizaje no supervisado.

1. Aprendizaje de maquina (ML)

Esta técnica muy usada en el aprendizaje automático se divide en dos grupos: aprendizaje supervisado y aprendizaje no supervisado.

1.1. Aprendizaje supervisado

En el aprendizaje supervisado el sistema considerado caja negra toma un conjunto de datos de entrenamiento anotados, cuyo propósito es obtener parámetros o características a partir de los datos, así como las relaciones entre ellos. De modo que, luego podrá predecir la anotación de datos no observados previamente.

1.2. Aprendizaje no supervisado

En este método el sistema o caja negra toma un conjunto de datos sin anotaciones, el objetivo principal es hacer que el modelo se ajuste a los datos que va observando de forma automática. Por lo que cada dato que observa se convierte en conocimiento.

2. Clasificadores

Los clasificadores encargados de anotar un objeto en la imagen, de forma que nos permite conocer que objetos se encuentran en ella, se dividen en:

2.1. K vecinos más cercanos

Este clasificador trabaja de tal forma que va almacenando todas las observaciones en el entrenamiento y asigna la clase vecina más cercana a los datos en la fase de evaluación, su principal ventaja es encontrar la categoría más cercana de forma rápida.

2.2. Perceptrón multicapa (MLP)

EL (MLP) se caracteriza por resolver problemas que no son linealmente separables, rompiendo la limitación del perceptrón simple que solo consta de una neurona. Además, requiere de un largo proceso para su aprendizaje.

Considerando las técnicas anteriormente descritas, se escogió usar la arquitectura YOLOv4 que está basada en CNN que es un modelo de clasificación y detección de imágenes haciendo uso de aprendizaje supervisado, cuyas ventajas y desventajas se comparan con versiones anteriores.

El modelo de detección que utiliza YOLOv4, es considerado uno de los más eficientes y potentes ya que permite que con unidades de procesamiento gráficas asequibles a cualquiera se pueda entrenar un detector de objetos rápido y preciso, en comparación con implementaciones anteriores. Por otro lado, incluye en su estado del arte CBN (*Cross mini Batch Normalization*) y PAN (*Path Aggregation Network*), como se muestra en la Figura 1.4 y en la Figura 1.5 respectivamente, que en esta implementación son más eficientes y adecuadas para entrenamientos con una sola GPU [8].

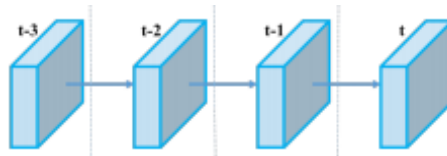


Figura 1.4 Cross mini Batch Normalization (CBN) [8]

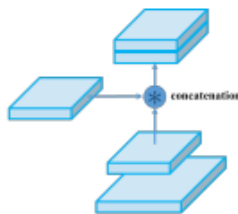


Figura 1.5 Path Aggregation Network (PAN) [8]

Sin embargo, esta arquitectura no es perfecta, dado que tiene una probabilidad muy alta de combinar varios objetos que se encuentran en los límites de los bloques procesados en una región y omitirlo debido a su fragmentación de bloques. Además, cuando se varía la información respecto a la base de datos tal como tamaño o número de clases su precisión se ve afectada negativamente.

Otra de las razones que motivan este proyecto, son las nuevas tecnologías de adquisición de imágenes o configuraciones en donde un claro ejemplo son los dispositivos móviles, que han pasado de tener una sola cámara como en la Figura 1.6 que solo se capturaba el espectro visible (RGB) a tener más cámaras tal como se muestra en la Figura 1.7 en donde varias de las fotos que

tomamos combinan ciertas técnicas de procesamiento para mejorar la imagen tomada aprovechando la información adicional que es capturada como lo es la profundidad dado por la tecnología de una de sus lentes *Lidar*.



Figura 1.6 iPhone 6 [21]



Figura 1.7 iPhone 12 [22]

Convencionalmente la mayoría de las técnicas existentes para realizar tareas de detección y clasificación de objetos trabajan sobre imágenes adquiridas en el espectro visible (RGB), aunque en los últimos años la aparición de las nuevas tecnologías de adquisición de imágenes ha permitido el desarrollo de nuevas técnicas que trabajan sobre imágenes de otros espectros electromagnéticos, tal como el sensor RedEdge (Figura 1.8) que genera imágenes multiespectrales. Esta ventaja permite que más características puedan ser extraídas sobre las imágenes que se procesan.



Figura 1.8 RedEdge-MX MicaSense [23]

En ámbitos de la ciencia encontramos diversos usos de las imágenes multiespectrales como es el caso del espectro NIR (*Near-Infrared*) que posee la capacidad de segmentar imágenes de acuerdo con el material del objeto, gracias a que la reflexión de la superficie de los objetos es dependiente del material. Con ello estas imágenes son ampliamente usadas en aplicaciones como monitorio de infestaciones de cultivos y cámaras de videovigilancia, lo que resulta ser un problema para los usuarios el orientarse con estas imágenes debido a la escasa discriminación entre los colores surgiendo una red neuronal DCGAN (*Deep Convolutional Generative Adversarial Network*) para la colorización de imágenes NIR. [12]

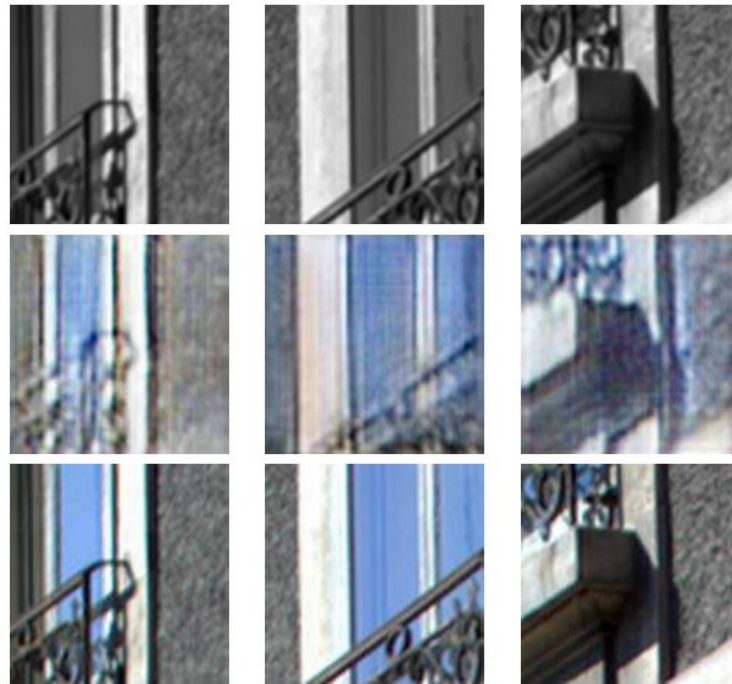


Figura 1.9 Pares de imágenes: (arriba) NIR, (medio) imagen colorizada por la red DCGAN, (abajo) imágenes RGB reales [12]

Adicionalmente, existen dispositivos que capturan información de más de 3 canales como es la Figura 1.8, que es un sensor de 5 canales; en donde varios sectores se ven beneficiados como lo es el sector agrícola. Por ejemplo, en la generación de modelos CGAN (*Conditional Generative Adversarial Network*) que usan de entrada imágenes de una sola banda espectral la NIR (*Near-Infrared*) produciendo una imagen sintética NDVI (Índice Normalizado de Diferencia de Vegetación) [15] que representa el estado de salud de la vegetación sin el uso de la banda R_{RED} necesaria para calcularlo de forma convencional, cabe mencionar que estos modelos fueron entrenados con información de ambas bandas (R_{RED} y NIR) tal como se muestra en la Figura 1.10.



Figura 1.10 Pares de imágenes: (arriba) NIR, (medio) RGB, (abajo) NDVI creado a partir de imágenes NIR y RGB [15]

Otro ejemplo son las configuraciones de drones con múltiples cámaras, donde antes se veían limitados a una sola donde su rango de adquisición era muy corto, por esto surgió la idea de adicionar más cámaras para captar un mayor campo (Figura 1.11) con mejores características surgiendo la tecnología de imágenes *Nadir* y oblicuas permitiendo el registro de huellas y fachadas de edificios lo cual simplifica la identificación e interpretación de algunos objetivos que son difíciles de reconocer a simple vista manejando gran cantidad de información [13].

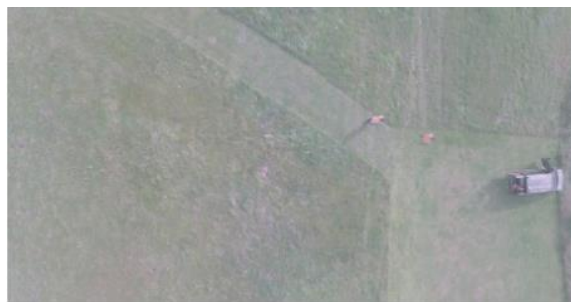


Figura 1.11 Imagen tomada del dron [13]

Continuando con el uso de las cámaras en los drones tenemos también que la existencia de programas de código abierto ayuda a combinar dichas imágenes de diferentes espectros en información útil como lo son los mapas completos (Figura 1.12) en formato RGB, CIR (*Color Infrared*) y NDVI brindando más oportunidades a los pequeños y medianos agricultores en

Ecuador disminuyendo costos por licencias de estos softwares [16]. Con esto podemos observar que el uso o combinación de diferentes bandas electromagnéticas son beneficiosas para conocer el estado del cultivo en este caso.

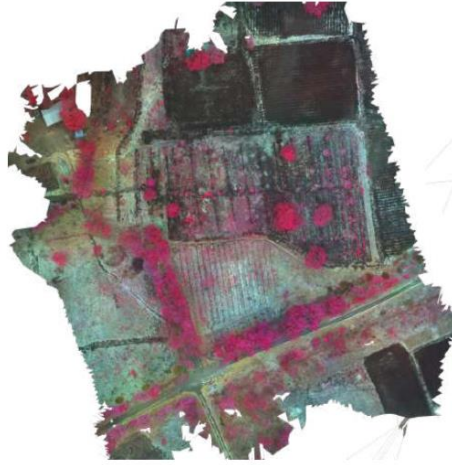


Figura 1.12 Mosaico completo CIR [16]

Tomando como base los ejemplos anteriores se propone analizar el comportamiento de la arquitectura de YOLOv4 sobre imágenes de más de 3 canales, en este caso de 4 canales para así observar si mejora la precisión en la detección y clasificación de objetos haciendo uso de una base de datos pública como lo es el *dataset* de KAIST [9]. Además de analizar si las técnicas de *data augmentation* convencionales más las nuevas como Mosaic y CutOut que fueron usadas para entrenar y realizar el *benchmark* de la red YOLOv4 influyen substancialmente a la robustez y mejora en la precisión de detectar y clasificar objetos en imágenes de 3 canales y 4 canales bajo el mismo *dataset*.

En el capítulo 2 se describen los módulos que tiene nuestro proyecto, y además se menciona la metodología que se usará para el desarrollo de cada uno de estos componentes o módulos. Mencionaremos los *datasets* a usar, las diferentes técnicas de *data augmentation*, las modificaciones que se le realizaron al código fuente de la arquitectura YOLOv4 para que permita entrenar y testear bajo imágenes de 4 canales. Y finalmente se describen las 4 métricas que se usaran para evaluar la arquitectura de YOLOv4 sobre imágenes de 3 canales y 4 canales ambas sin *data augmentation* y con *data augmentation*.

CAPÍTULO 2

En el presente capítulo mostraremos los componentes en los cuales nos basaremos para el desarrollo del proyecto, así como la metodología que se va a usar como solución para cada uno de los componentes propuestos. De esta forma, el capítulo inicia presentando los componentes considerados para el proyecto como son: bases de datos, pre-procesamiento, detección y clasificación de objetos sobre imágenes de cuatro canales, al final del capítulo se presenta la validación y pruebas de la arquitectura YOLOv4 y YOLOv4_4C.

2. Metodología

De forma general se usó una metodología ágil basada en el uso de SCRUM [18], en donde se realizaron reuniones semana a semana para revisar y validar los avances obteniendo una retroalimentación por parte de los involucrados, lo cual generó posibles mejoras y pruebas en el proyecto.

Con el objetivo de subdividir todo el problema global contenido en nuestra propuesta de proyecto, proponemos que este problema sea subdividido en pequeños paquetes de subproblemas a los cuales los llamaremos componentes. De esta forma, los componentes principales para el desarrollo de este proyecto son:

- Bases de datos
- Pre-Procesamiento
- Detección y clasificación de objetos sobre imágenes de cuatro canales
- Validación y pruebas

2.1 Bases de datos

En primer lugar, tenemos el componente que lo llamaremos base de datos. Este componente tiene como objetivo generar dos bases de datos que servirán para validar los resultados de detección y clasificación sobre imágenes de tres canales y sobre imágenes de cuatro canales.

Para cumplir con este propósito se tomará como base de datos inicial la fuente de datos pública conocida como KAIST [9], la cual posee un conjunto de datos (*dataset*) con pares de imágenes que incluyen imágenes RGB (imágenes de tres canales) y su correspondiente imagen térmica (LWIR) de personas, ciclistas y conjuntos de personas denominado como *people* en tres

diferentes lugares, durante el día y la noche con sus respectivas anotaciones, tal como se muestra en la Figura 2.1.



Figura 2.1 Ejemplo de imágenes del dataset de KAIST. (Izquierda) RGB, (Derecha) LWIR de la misma escena [9]

A partir de la base de datos original de KAIST, cuatro bases de datos son generadas. Estas bases de datos son: base de datos de imágenes de tres canales (BD 3 canales), base de datos de imágenes de tres canales aumentada (BD 3 aumentada), base de datos de imágenes de cuatro canales (BD 4 canales) y base de datos de imágenes de cuatro canales aumentada (BD 4 canales aumentada), la Figura 2.2 muestra la generación de estas cuatro bases de datos.

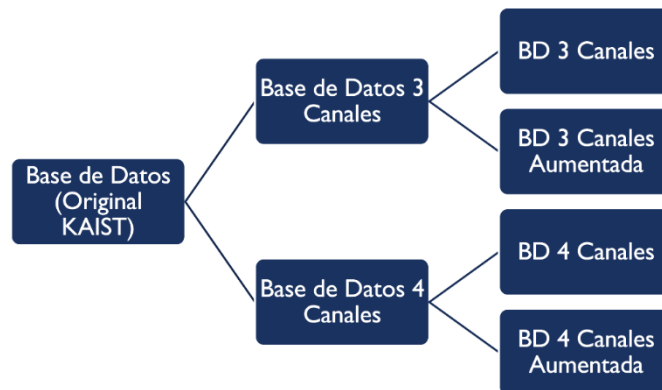


Figura 2.2 Esquema de la generación de las 4 bases de datos a usarse en el proyecto

Se debe tomar en cuenta que la BD 3 canales y la BD 3 canales aumentada serán usadas para medir los resultados obtenidos con la actual arquitectura de YOLOv4, la cual solo funciona con imágenes de tres canales, mientras que la BD 4 canales y la BD 4 canales aumentada servirán para medir los resultados con la nueva versión de YOLOv4_4C que se propone desarrollar en este proyecto sobre imágenes de cuatro canales.

2.1.1 Generación de la BD 3 canales y la BD 3 canales aumentada

La generación de estas bases de datos es necesaria debido a que la base de datos original de KAIST posee imágenes las cuales están divididas en día y noche, así como en diferentes lugares donde se realizó la adquisición como *campus*, *road* y *downtown*, de modo que no se encuentran en una sola carpeta lo cual dificulta el entrenamiento de la arquitectura de YOLOV4.

Para la generación de BD 3 canales se tomó la base de datos original de KAIST (Figura 2.3) que consta de 14100 imágenes en RGB, a la cual se le realizó un proceso de pre-procesamiento que será descrito posteriormente en la sección 2.2, en donde se propone crear un script que luego del pre-procesamiento realice la división de la base de datos en entrenamiento (80%) y pruebas (20%), dado que la arquitectura realiza un proceso de validación, el conjunto de pruebas será usado como evaluación. También se realizará la tarea de copiar cada anotación con su respectiva imagen a un directorio específico.

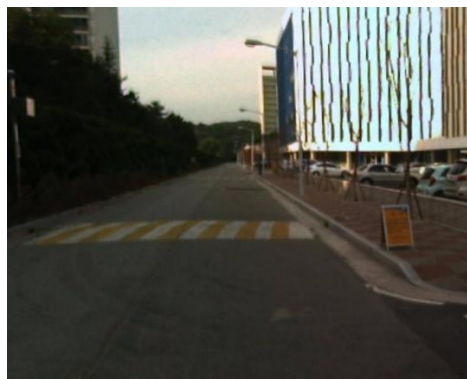


Figura 2.3 Ejemplo de una imagen RGB del *dataset* KAIST [9]

Por otro lado, para generar DB 3 canales aumentada se toma como entrada la base de datos DB 3 canales obtenida previamente, a la cual se le aplicaron técnicas de aumento de datos (*data augmentation*) como: rotación, *crop*, *shear*, *flip*, *scale* y *cutout*, como se muestra en la Figura 2.4 además de emplear la técnica de *mosaic* que fue usada en el *benchmark* de YOLOv4, tal como se observa en la Figura 2.5.



Figura 2.4 Ejemplos de imágenes RGB usando data augmentation



Figura 2.5 Ejemplo de imagen RGB usando data augmentation mosaic

2.1.2 Generación de la BD 4 canales y la BD 4 canales aumentada

La importancia de la generación de estas bases de datos se basa principalmente en convertir la BD 3 canales en imágenes de cuatro canales, las cuales serán necesarias como datos de entrada para la arquitectura YOLOv4_4C, lo cual es el propósito de este proyecto.

Para la generación de BD 4 canales se toma como datos a procesar la BD 3 canales con su respectiva componente térmica, cada par de imágenes (RGB y LWIR) son adicionadas generando como resultado imágenes de cuatro canales, tal como se muestra en la Figura 2.6.

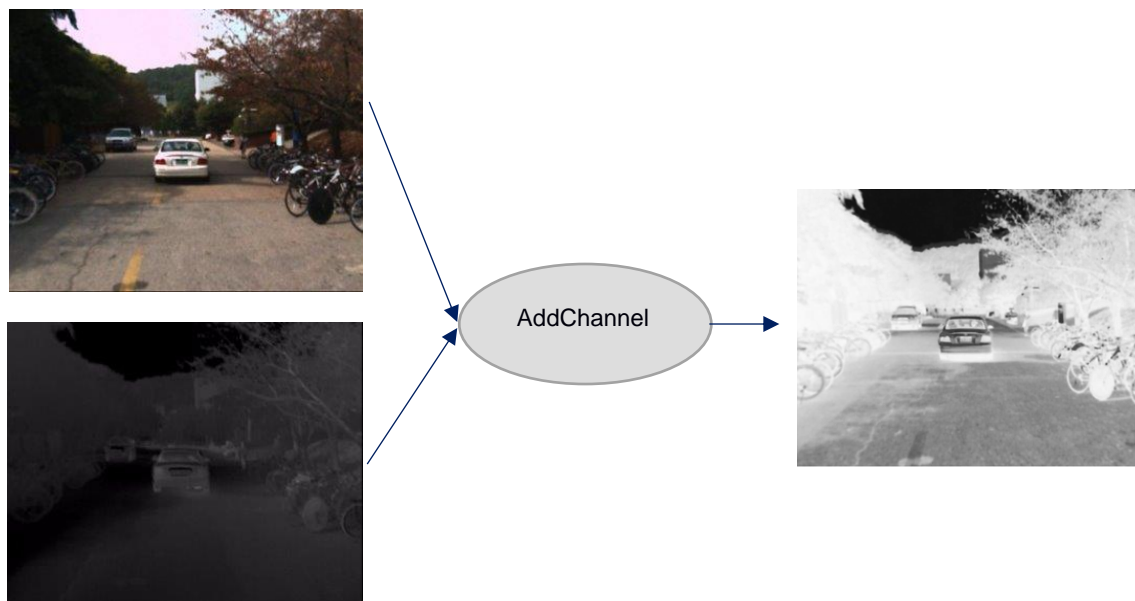


Figura 2.6 Proceso de creación de imágenes de 4 canales – BD 4 canales

Para la generación de BD 4 canales aumentada se toma como referencia el procedimiento aplicado en la generación de la base de datos anterior (DB 4 canales) y se aplica las mismas técnicas de aumento de datos usadas en la sección 2.1.1, pero en este caso no se encuentran implementadas por parte de la arquitectura YOLOv4, ya que esta solo trabaja con imágenes de tres canales (RGB), por lo que se creó un script para la generación de imágenes de cuatro canales con las mismas técnicas que se usaron en el conjunto de datos anterior como se muestran en la Figura 2.7 y Figura 2.8.

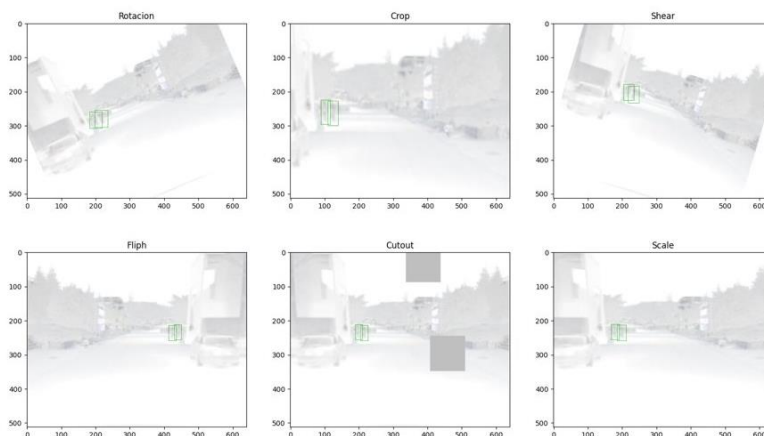


Figura 2.7 Ejemplo de imágenes cuatro canales usando *data augmentation*



Figura 2.8 Ejemplo de imágenes cuatro canales usando data augmentation *mosaic*

2.2 Pre-Procesamiento

En segundo lugar, tenemos el componente llamado Pre-Procesamiento cuya importancia radica en el formato de anotaciones provistas originalmente por el conjunto de datos, en donde dichas anotaciones están en un formato Video Bounding Box (VBB), las cuales serán transformadas en formato Visual Object Class (VOC) para finalmente ser transformada al formato que utiliza la arquitectura actual de YOLOv4, tal como se muestra en la Figura 2.9. Luego de completar la transformación de las anotaciones de las imágenes, se realizará un proceso de generación de imágenes de cuatro canales.

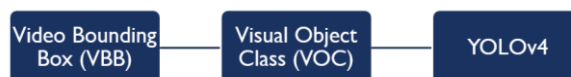


Figura 2.9 Proceso de transformación de anotaciones al formato YOLOV4

Empezaremos describiendo que es una anotación o etiquetado en imágenes con un breve concepto sobre la misma, una anotación podría indicar si un objeto existe en ella, es decir, trata de informar sobre lo que la imagen contiene, como se muestra en la Figura 2.10.



Figura 2.10 Ejemplo de etiquetado de personas en una imagen RGB

Para lograr nuestro propósito de entrenar la arquitectura YOLOv4 modificada debemos de realizar la transformación de las anotaciones de las imágenes al formato YOLOv4, así como generar imágenes de cuatro canales ambos que serán detallados en los siguientes subcomponentes.

2.2.1 Transformación de anotaciones sobre imágenes

El conjunto de datos original de KAIST posee anotaciones que se encuentran en el siguiente formato:

Name, Xmin, Ymin, Xmax, Ymax

En donde *Name* es un identificador del nombre de la clase a la que pertenece, *Xmin* y *Ymin* son las coordenadas de la esquina superior izquierda de la anotación para la imagen RGB, *Xmax* y *Ymax* son el ancho y alto de la anotación para la imagen RGB. Cabe mencionar que estas mismas coordenadas son compartidas con la imagen térmica al estar perfectamente alineadas con la imagen RGB, de modo que no se requiere un proceso de registrado.

Además, las imágenes presentan un identificador de oclusión, que según el contexto de la investigación realizada por Jason Nataprawira, Yanlei Gu, Koki Asami e Igor Goncharenko, es usada para identificar a que nivel se encuentra el objeto, es decir si se encuentra superpuesto por el mismo objeto; el que se encuentra al frente será considerado como 0 (verde), el de en medio será considerado como 1 (amarillo) y el que se encuentra detrás será considerado como 2 (rojo), como se muestra en la Figura 2.11.

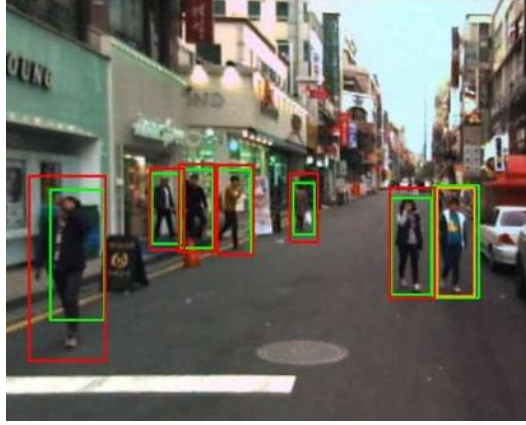


Figura 2.11 Ejemplo de oclusión en imagen RGB de la base de datos KAIST [9]

Considerando que para la versión actual de YOLOv4 utiliza el formato descrito a continuación:

object-class, x_center, y_center, width, height

En donde *object-class* es un identificador de la clase a la cual pertenece la anotación, *x_center* y *y_center* son las coordenadas del centro de la anotación relativos a la imagen, es decir el punto central del rectángulo relativo a la imagen, *width* y *height* son el ancho y alto de anotación, es decir el ancho y alto del rectángulo relativos a la imagen. Al mencionar relativo a la imagen nos referimos a que cada uno es dividido para el tamaño real de la imagen para obtener coordenadas que puedan ser usadas al momento de realizar un redimensionamiento de la imagen.

Para lograr transformar el formato de anotación original provisto por el conjunto de datos de KAIST al que utiliza la versión actual de YOLOv4 se utilizó las siguientes fórmulas, mostrado en Ec 1.

$$\begin{aligned}
 x_{center} &= \frac{\text{coordenada } x \text{ de la anotacion}}{\text{ancho real de la imagen}} \\
 y_{center} &= \frac{\text{coordenada } y \text{ de la anotacion}}{\text{alto real de la imagen}} \\
 width &= \frac{\text{ancho de la anotacion}}{\text{ancho real de la imagen}} \\
 height &= \frac{\text{alto de la anotacion}}{\text{alto real de la imagen}}
 \end{aligned}
 \tag{Ec. 1}$$

Las mismas que son descritas de forma visual en la Figura 2.12 para un mejor entendimiento a continuación:

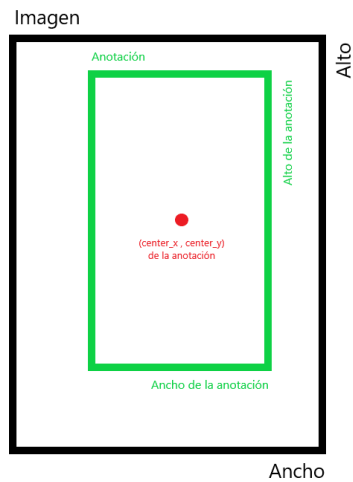


Figura 2.12 Representación visual del formato de anotación de la arquitectura actual de YOLOv4 [9]

2.2.2 Generación de imágenes de cuatro canales

Luego de realizar la transformación de la anotación del conjunto de datos se procedió a generar imágenes de cuatro canales que consiste en la mezcla de la componente a color con la térmica para así generar una imagen de cuatro canales como se muestra en la Figura 2.13. El objetivo de esta generación es debido a que este tipo de imágenes poseen una mayor cantidad de características en comparación con las originales (RGB) y son almacenadas en formato *PNG* que permite guardar un cuarto canal conocido como *alpha* [19].

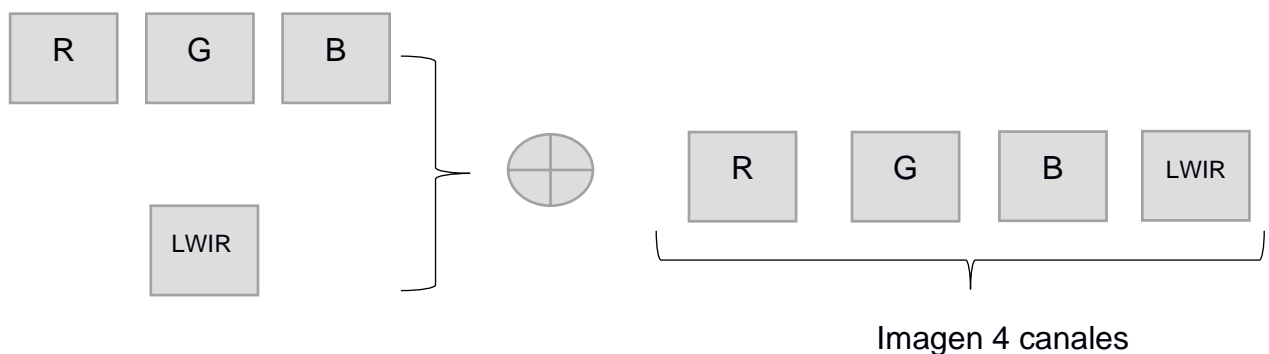


Figura 2.13 Proceso de creación de imagen de 4 canales

2.3 Detección y clasificación de objetos sobre imágenes de cuatro canales

Este componente llamado detección y clasificación de objetos sobre imágenes de cuatro canales propone analizar el código fuente de la implementación original de YOLOv4 que trabaja sobre imágenes de tres canales (RGB), para extender su funcionalidad de trabajo sobre imágenes de cuatro canales. Para esto, es necesario primero encontrar en donde el código fuente original realiza la lectura de imágenes (RGB), de modo que pueda ser modificado para leer imágenes de más de tres canales, lo cual es la propuesta del presente proyecto. Esta modificación del código fuente original permitirá realizar una comparación de cómo se comporta la arquitectura YOLOV4_4C con imágenes que poseen una mayor cantidad de características (4 canales).

2.3.1 Lectura del código fuente de YOLOv4

La lectura del código fuente de la implementación actual de YOLOv4, permitió entender cómo se encuentra estructurada la implementación de esta, en donde se encontraron archivos principales que realizan ciertos procesos, así como archivos secundarios donde solo se realizan tareas de configuración. Por lo que en la estructura del proyecto describimos los más importantes a continuación:

- **Makefile:** Archivo usado para la compilación del proyecto en un sistema operativo Linux, con banderas para compilación usando herramientas como *cuda*, *OpenCV*, entre otras.
- **Darknet.c:** Archivo que ejecuta la arquitectura de red o el proyecto en sí.
- **Detector.c:** Archivo para ejecutar la arquitectura de forma específica, en donde se reciben los parámetros necesarios para que la misma se ejecute.
- **Network.c:** Archivo donde se observa la configuración de la red o arquitectura.
- **Data.c:** Se encarga de la lectura de parámetros necesarios para la ejecución de la arquitectura.
- **Image.c:** Aquí encontramos como la arquitectura realiza la lectura de las imágenes en donde se encuentra definido que, para imágenes de más de tres canales, realice una conversión a imágenes de cuatro canales dependiendo de la bandera usada.
- **Image_opencv.cpp:** Archivo que posee funciones de OpenCV para leer imágenes en donde se utilizan banderas para controlar el tipo de imágenes que se están usando.

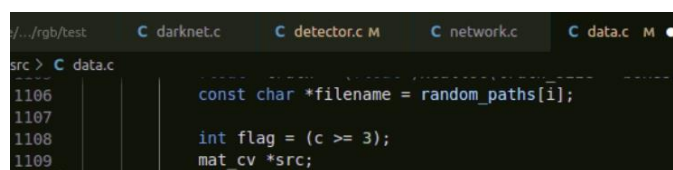
2.3.2 Modificación del código fuente para leer imágenes de cuatro canales

Continuando en conjunto con el paso anterior en donde se encontró que en el archivo `data.c` se realiza la carga o lectura de imágenes en memoria, de tal forma que en este archivo es donde se debió realizar la modificación para lograr leer imágenes de cuatro canales como se describe a continuación:

Al revisar en detalle el archivo `data.c` notamos que posee una bandera de la siguiente forma $c \geq 3$, donde c es número de canales de forma que para imágenes de más de tres canales c se vuelve verdadero que a nivel del lenguaje es interpretado como uno y en el caso de ser menor se vuelve falso que es interpretado como cero, de forma que siguiendo los datos provistos por OpenCV:

- `IMREAD_UNCHANGED = -1`
- `IMREAD_GRAYSCALE = 0`
- `IMREAD_COLOR = 1`

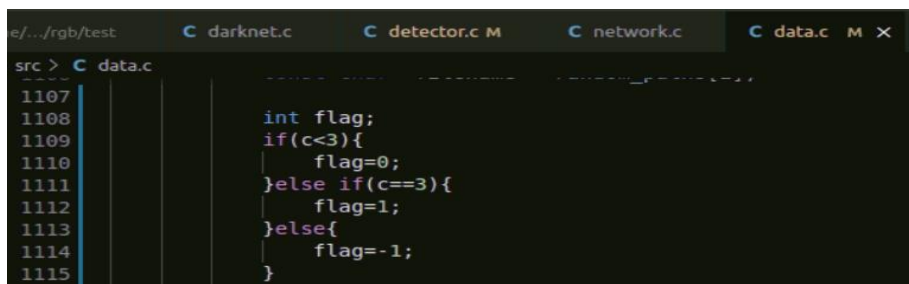
Solo se consideran los casos en los que son 0 o 1 obteniendo imágenes de color o imágenes a escala de grises, sin considerar otros canales como se muestra en la Figura 2.14.



```
src > C data.c
1106 const char *filename = random_paths[i];
1107
1108 int flag = (c >= 3);
1109 mat_cv *src;
```

Figura 2.14 Archivo antes de la modificación para lectura de imágenes tres canales

Se realizó el cambio para que pueda realizar la lectura de imágenes de más de tres canales, como se muestra en la Figura 2.15.



```
src > C data.c
1107
1108     int flag;
1109     if(c<3){
1110         flag=0;
1111     }else if(c==3){
1112         flag=1;
1113     }else{
1114         flag=-1;
1115     }
```

Figura 2.15 Archivo después de la modificación para lectura de imágenes de más de tres canales

De esta forma la arquitectura podrá leer imágenes de más de tres canales dado que utiliza OpenCV para la lectura de esta y requiere esta bandera para saber si debe realizar una transformación sobre la imagen.

2.4 Validación y pruebas

El presente componente que llamaremos Validación y pruebas se basa en presentar los resultados obtenidos luego de las modificaciones realizadas en el código original de la arquitectura YOLOv4 y los entrenamientos realizados haciendo uso de los conjuntos de datos anteriormente descritos. De forma que se compruebe que tipo de resultados se obtuvieron y el porqué de estos.

Para esto, realizaremos los entrenamientos sobre los diferentes escenarios de ambas arquitecturas teniendo: imágenes RGB (Día), imágenes RGBT de cuatro canales (Día), imágenes RGB con aumento de datos (Día), imágenes RGBT de cuatro canales con aumento de datos (Día), imágenes RGB (Noche), imágenes RGBT (Noche), imágenes RGB con aumento de datos (Noche) e imágenes RGBT con aumento de datos (Noche), tal como se muestra en la Figura 3.1. Estos entrenamientos serán realizados en el capítulo 3.

2.4.1 Validación

El objetivo de la validación es detallar los diferentes mecanismos a tomar en cuenta para poder evaluar el correcto funcionamiento de la arquitectura modificada con el fin de observar el alcance que poseerá.

Entre los mecanismos que analizaremos tenemos: Gráficas y Precisiones para cada uno de los diferentes entrenamientos, los cuales permitirán obtener información necesaria para generar comparativas y conclusiones.

2.4.1.1 Gráficas

Las gráficas generadas por las arquitecturas YOLOv4 y YOLOv4_4C al momento de ser evaluadas en los escenarios descritos al inicio del componente 2.4, estará compuesta por el porcentaje de pérdida y precisión del grupo de validación por el número de iteraciones realizadas, de esta forma se observará que cada grafica será diferente dependiendo del escenario evaluado, caso contrario no estaríamos entrenando correctamente el modelo bajo 4 canales que es propósito del actual proyecto.

2.4.1.2 Precisiones

La precisión para cada uno de los escenarios descritos al inicio del componente 2.4, los cuales serán comparados y analizados para comprender y concluir sobre el comportamiento de la arquitectura YOLOv4 y YOLOv4_4C bajo las condiciones planteadas tales como aumento de datos y adicionar un cuarto canal a una imagen RGB proponiendo trabajos a futuro.

2.4.2 Pruebas

A continuación, describiremos como se realizaron las pruebas que usamos para evidenciar las distintas precisiones obtenidas bajo los distintos escenarios en los cuales se evaluaron ambas arquitecturas.

2.4.2.1 Comandos de darknet

En las presentes pruebas haremos uso de comandos preestablecidos de *darknet* quien es el que define la arquitectura de YOLOv4 y YOLOv4_4C. Con el comando “*darknet map detector*” generaremos diversas salidas como la detección y clasificación de una imagen del conjunto de prueba de cada uno de los 8 *datasets* dándonos a su vez el porcentaje de confianza de cada clase.

En el capítulo 3 se presentan y se analizan los resultados generados para el componente de validación y pruebas.

CAPÍTULO 3

Este capítulo muestra los resultados obtenidos en la implementación del componente de detección y clasificación de objetos sobre imágenes de cuatro canales. Para esto, usaremos las arquitecturas de YOLOv4 y YOLOv4_4C sobre diferentes escenarios tales como: imágenes RGB (Día), imágenes RGBT de cuatro canales (Día), imágenes RGB con aumento de datos (Día), imágenes RGBT de cuatro canales con aumento de datos (Día), imágenes RGB (Noche), imágenes RGBT (Noche), imágenes RGB con aumento de datos (Noche) e imágenes RGBT con aumento de datos (Noche). Cada dos escenarios realizaremos un análisis de las gráficas y precisiones obtenidas, comparando con los resultados obtenidos en la versión anterior de YOLOv3 [24].

Además, debemos mencionar que en los escenarios que se usaron imágenes RGB hicimos uso de *transfer learning* con los pesos que se obtuvieron del *benchmark* de la arquitectura de YOLOv4 [8] con el *dataset* MS COCO ya que este *dataset* posee imágenes de personas y ciclistas por lo que brindará un mejor punto de inicio a nuestras redes para entrenar. Por último, se hará uso de las 7 técnicas de *data augmentation* mencionadas en la sección 2.1.2.

3. Resultados y análisis del componente de detección y clasificación de objetos sobre imágenes de cuatro canales

Como se mencionó anteriormente nuestro proyecto nuestro proyecto tiene cuatro componentes: bases de datos, pre-procesamiento, detección y clasificación de objetos sobre imágenes de cuatro canales y el último componente de validación y pruebas.

El objetivo del componente de base datos es generar el conjunto de datos necesario para los 8 distintos entrenamientos que se realizarán en la sección siguiente, la implementación de este componente fue descrita en la sección 2.1. Por otro lado, el componente de pre-procesamiento se basa en transformar el formato de anotación *video bounding box* (VBB) con el tipo de anotación que trabaja la arquitectura YOLOv4; esto es implementado en la sección 2.2.

Para el componente 3 su importancia radica en analizar el actual código fuente de la arquitectura YOLOv4 y realizar las modificaciones pertinentes a fin de crear una nueva arquitectura YOLOv4_4C que procesa imágenes de 4 canales, esto ha sido detallado en la sección

2.3. Finalmente, el componente 4 tiene como propósito comparar y analizar el funcionamiento de la nueva arquitectura YOLOv4_4C con YOLOv4 para así generar las conclusiones correspondientes, que serán detallados en las próximas secciones. Los escenarios que se evaluarán son los siguientes:

| | | | |
|----------------------------|--|------------------------------|--|
| RGB (Día) | RGB + Data Augmentation (Día) | RGB (Noche) | RGB + Data Augmentation (Noche) |
| RGB + Thermal (Día) | RGB + Thermal + Data Augmentation (Día) | RGB + Thermal (Noche) | RGB + Thermal + Data Augmentation (Noche) |

Figura 3.1 Escenarios de entrenamiento para ser evaluados con YOLOv4 y YOLOv4_4C

3.1 Escenario 1: Imágenes RGB (Día)

El primer escenario en el cual se evaluó la arquitectura de YOLOv4 usando solo imágenes RGB en el día muestra resultados del 20.7% en promedio para las tres clases presentadas en la sección 2.1, como se muestra en la Figura 3.2, que posteriormente serán analizadas de forma individual en una tabla para luego comparar sus resultados.

En la gráfica podemos analizar como la precisión del conjunto de *test* (línea roja) llega a una región *plateau* en donde el porcentaje para este conjunto se estabiliza entre 20 y 22% significando que la red ya no podía aprender más del conjunto de *train* al tener un valor de *loss* (línea azul) bastante bajo en esta región.

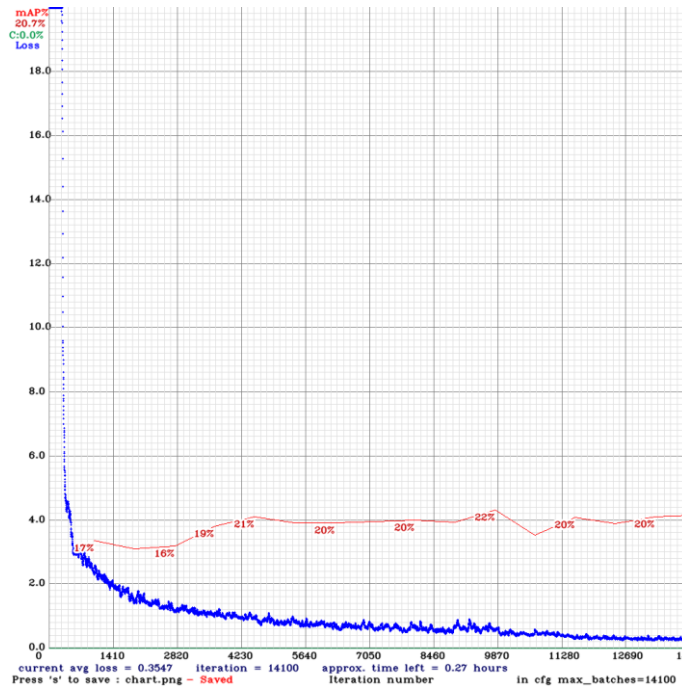


Figura 3.2 Entrenamiento de imágenes RGB (Día) utilizando la arquitectura de YOLOv4

3.2 Escenario 2: Imágenes RGB + *Thermal* (Día)

El segundo escenario en el cual se evaluó la arquitectura propuesta de YOLOv4_4C usando imágenes RGBT de cuatro canales en el día; muestra resultados del 10.8% en promedio para las tres clases, cómo se muestra en la Figura 3.3, la cual será analizada de forma individual, para luego pasar a una comparativa con el escenario anterior.

La gráfica 3.3 nos muestra una variabilidad en la precisión del conjunto de datos de *test* alcanzando un máximo de 16% pero disminuyendo a un 10.8% de precisión gracias a que el *loss* que poseía le permitía mejorar o decrecer la precisión al ajustar los pesos; este efecto es gracias al uso un cuarto canal bajo ambientes adversos a su fortaleza.



Figura 3.3 Entrenamiento de imágenes RGB + *Thermal* (Día) utilizando la arquitectura de YOLOv4_4C

3.3 Comparativa de imágenes RGB (Día) e imágenes RGB + *Thermal* (Día)

En ambos escenarios se obtuvo una curva con ciertas variaciones, pero ambas llegaron a obtener relativamente la misma precisión promedio para las tres clases. Se obtiene (Tabla 1) una mejora del 7.44% con solo usar una nueva versión de YOLO lo que nos muestra que la arquitectura v4 posee mejores extractores de características. También podemos evidenciar que el uso de un cuarto canal puede llegar a ser contraproducente bajo ciertos escenarios como el LWIR en el día disminuyendo la precisión un 2.31% pero cabe recalcar que en YOLOv4_4C no se hizo uso de *transfer learning* y a pesar de ello la diferencia no es cuantiosa.

Tabla 3.1 Comparación de imágenes RGB (Día) y RGB + *Thermal* (Día)

| AP% | Persona | Personas | Ciclista |
|----------------------------|---------|----------|----------|
| RGB YOLOv3 | 45 | <5 | <1 |
| RGB YOLOv4 | 52.44 | 0.91 | 11.29 |
| RGB + Thermal YOLOv4_4C | 42.69 | 2.16 | 4.25 |

3.4 Escenario 3: Imágenes RGB + *data augmentation* (Día)

En este escenario evaluamos la arquitectura de YOLOv4 haciendo uso de imágenes RGBT con *data augmentation* para observar en cuanto se puede mejorar la precisión y robustez. Sin embargo; esta Figura 3.4 se puede ver que no mejoró la precisión con respecto a no usar *data augmentation* en la Figura 3.2. Se obtuvo un 13% de precisión para las tres clases en el conjunto de *test*, mientras que produjo un 21% como máximo en su primera evaluación. Siendo superado por un 1% en precisión con respecto a no usar *data augmentation*.

Lo interesante de la gráfica es que cada vez que se ejecutaba el proceso de validación con los pesos de esa iteración la precisión para las tres clases iba decreciendo poco a poco. Dándonos a entender que las técnicas de *data augmentation* podrían haber estado confundiendo a la red hasta decrecer en un 8% de precisión.

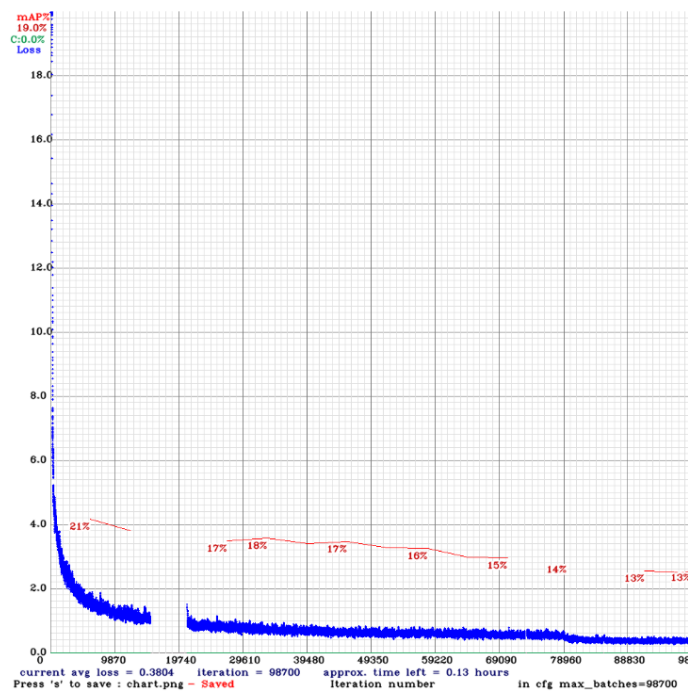


Figura 3.4 Entrenamiento de imágenes RGB + *data augmentation* (Día) utilizando la arquitectura de YOLOv4

3.5 Escenario 4: Imágenes RGB + *Thermal* + *data augmentation* (Día)

En este escenario evaluamos la arquitectura de YOLOv4 haciendo uso de imágenes RGBT con *data augmentation* para observar en cuanto se puede mejorar la precisión y robustez respecto a no usarlo. La Figura 3.5 se puede ver que no mejoró la precisión con respecto a no usar *data augmentation* en la Figura 3.3. se obtuvo un 16% de precisión para las tres clases en el conjunto de *test*, mientras que en este escenario un 15% como máximo de precisión. Siendo superado por un 1% en precisión con respecto a no usar *data augmentation*.

Se repite el decrecimiento de la precisión en cada iteración al igual que RGB + Data Augmentation (Día), aunque en ciertos intervalos la precisión oscilaba entre 12-11% llegando finalmente a caer en un 7%. También se puede ver en la gráfica que el *loss* ya no posee tantos picos desde la iteración 78960.

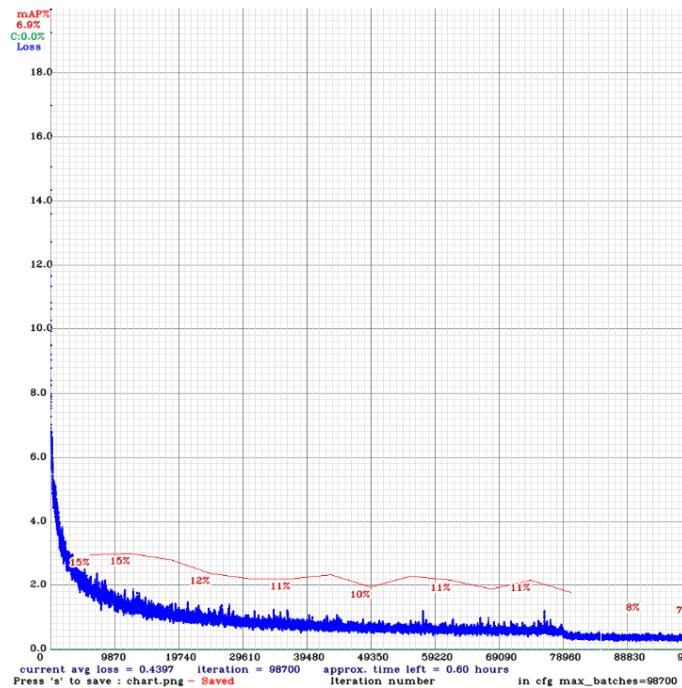


Figura 3.5 Entrenamiento de imágenes RGB + *Thermal* + *data augmentation* (Día) utilizando la arquitectura de YOLOv4

3.6 Comparativa de imágenes RGB + *data augmentation* (Día) e imágenes RGB + *Thermal* + *data augmentation* (Día)

En la Tabla 2 vemos las precisiones para cada una de las clases con los mejores pesos de cada arquitectura; se puede analizar que el uso de las técnicas de *data augmentation* no siempre serán muy beneficiosas ya que comparando con no usarlo (Tabla 1) se obtiene una diferencia de 3.7% que es muy baja, más aún por el costo computacional de entrenar aplicando 6 técnicas de *data augmentation* que es muy alto. Mientras que para ciclista se mejoró un 1.6%. Aun así, la precisión de la arquitectura YOLOv4 superó en un 3.74% en la precisión respecto a la clase persona de YOLOv3.

El uso de un cuarto canal y *data augmentation* continúa advirtiéndonos sobre la utilidad de un cuarto canal bajo condiciones no favorables como es el día al obtener resultados menores (-3.74%) al solo usar 3 canales. Mientras que es importante mencionar que no se usó *transfer learning* sobre los entrenamientos de 4 canales y aun así mantuvo una diferencia del 2.15% usando el *dataset* KAIST que posee menos objetos de la clase persona en comparación a MS COCO.

Tabla 3.2 Comparación RGB + *data augmentation* (Día) y RGB + *Thermal* + *data augmentation* (Día)

| AP% | Persona | Personas | Ciclista |
|----------------------------|---------|----------|----------|
| RGB YOLOv3 | 45 | <1 | <5 |
| RGB DA YOLOv4 | 48.74 | 0.91 | 12.89 |
| RGB + Thermal DA YOLOv4_4C | 42.85 | 0.35 | 1.49 |

3.7 Escenario 5: Imágenes RGB (Noche)

En el quinto escenario se utilizó el conjunto de datos de la noche, para entrenar la arquitectura actual de YOLOv4, en la cual se usó imágenes RGB, obteniendo una precisión promedio de las tres clases del 3.4%, como se observa en la Figura 3.6. En esta gráfica se puede ver que alcanza una precisión máxima de 5% para las tres clases; lo cual es muy bajo a pesar de hacer uso de *transfer learning*.

Y en este escenario es en donde nuestro proyecto se basa, al analizar que la ausencia de *features* importantes o discriminantes en un contexto como las imágenes RGB en la noche

producen precisiones muy bajas por lo que se analizara en la sección 3.8 el resultado de añadir un cuarto canal con esta premisa.

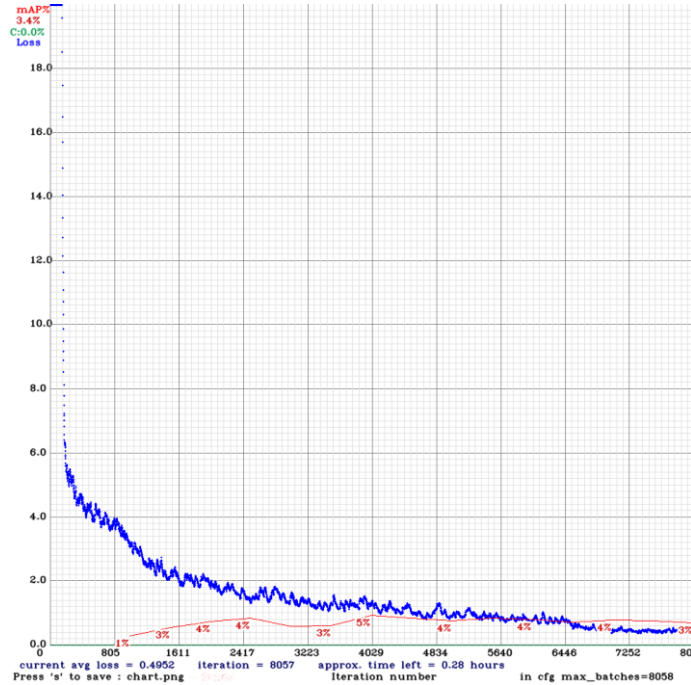


Figura 3.6 Entrenamiento de imágenes RGB (Noche) utilizando la arquitectura de YOLOv4

3.8 Escenario 6: Imágenes RGB + *Thermal* (Noche)

El sexto escenario considerado el más clave para los objetivos del proyecto en el cual se evaluó la arquitectura de YOLOv4_4C, con imágenes RGB más la componente térmica, la cual tiene más efecto en la noche ya que agrega características importantes para el entrenamiento, donde la curva es totalmente diferente a la de la sección 3.7. Logrando una precisión promedio para las tres clases del 14.6%, tal como se muestra en la Figura 3.7.

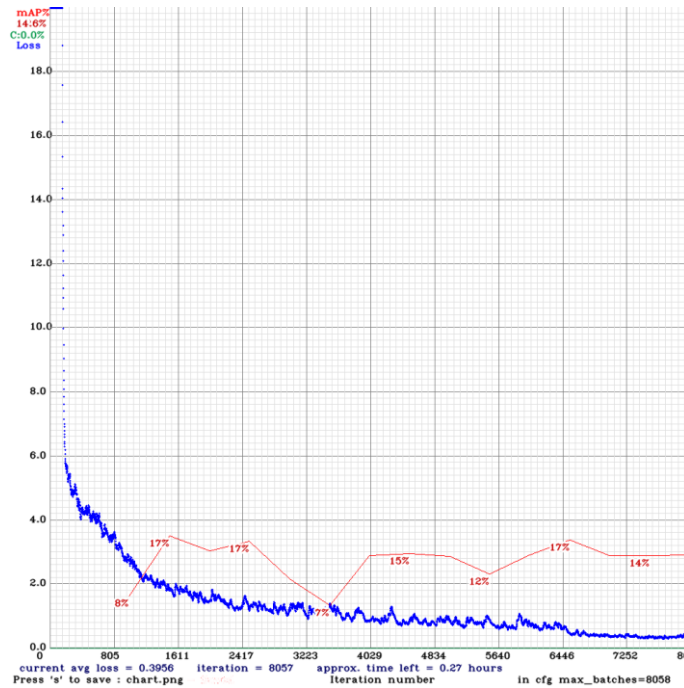


Figura 3.7 Entrenamiento de imágenes RGB + *Thermal* (Noche) utilizando la arquitectura de YOLOv4_4C

3.9 Comparativa de imágenes RGB (Noche) e imágenes RGB + *Thermal* (Noche)

Podemos evidenciar que el uso correcto de un cuarto canal ayuda a incrementar la precisión en la detección y clasificación de objetos. Tal es el caso de la componente LWIR que permite diferenciar las 3 clases del fondo al poder capturar su temperatura.

En la Tabla 3 analizamos que la arquitectura YOLOv4 posee un decremento de precisión respecto a su versión previa YOLOv3. Por lo que los extractores de *features* en YOLOv4 poseen una cierta dificultad bajo escenarios nocturnos usando imágenes RGB disminuyendo en un 6.84%. Mientras que en la nueva YOLOv4_4C podemos superar en un 22.63% a YOLOv3 al hacer uso del canal térmico.

Tabla 3.3 Comparación RGB (Noche) y RGB + *Thermal* (Noche)

| AP% | Persona | Personas | Ciclista |
|------------|---------|----------|----------|
| RGB YOLOv3 | 20 | <1 | 30 |
| RGB YOLOv4 | 13.16 | 0.21 | 0.69 |

| | | | |
|----------------------------|-------|------|------|
| RGB + Thermal YOLOv4_4C | 42.63 | 9.57 | 0.48 |
|----------------------------|-------|------|------|

3.10 Escenario 7: Imágenes RGB + *data augmentation* (Noche)

El séptimo escenario evaluó la arquitectura YOLOv4 sobre imágenes RGB en la noche con la particularidad de analizar el efecto de aplicar *data augmentation* sobre el *dataset*.

En la Figura 3.8 se obtuvo un 5.6% de precisión para las 3 clases, pero a su vez un máximo de 8%. Algo interesante es la curva de *loss* que a diferencia de no usar *data augmentation* los saltos no son muy altos, mientras que aquí son pronunciados haciendo que la red aprenda. Además, la curva de precisión para las tres clases (línea roja) se mantuvo nuevamente oscilando entre 6-7% lo que nos da a entender que ya no se podía mejorar la precisión con el conjunto de datos que con que se entrenaba.

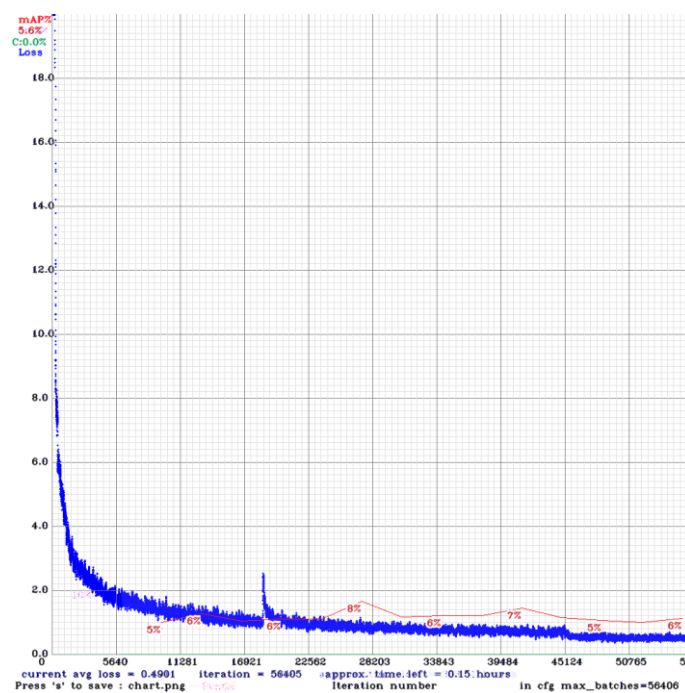


Figura 3.8 Entrenamiento de imágenes RGB + *data augmentation* (Noche) utilizando la arquitectura de YOLOv4

3.11 Escenario 8: Imágenes RGB + *Thermal* + *data augmentation* (Noche)

En el presente escenario entrenamos la arquitectura YOLOv4_4C usando las imágenes RGBT de los 3 sitios en la noche y aplicando las 6 técnicas de *data augmentation*.

En la Figura 3.9 vemos como existe una gran cantidad de picos en el *loss* (línea azul) que representa como las técnicas de *data augmentation* obligan a la red a aprender mientras que alcanza una precisión final de 10% para las 3 clases, y un 23% de precisión máxima en su primera evaluación. La precisión para el conjunto de datos de *test* (línea roja) decrece de un 23% hasta llegar a oscilar entre 13% y 10% sucediendo lo mismo que en la Figura 3.5.

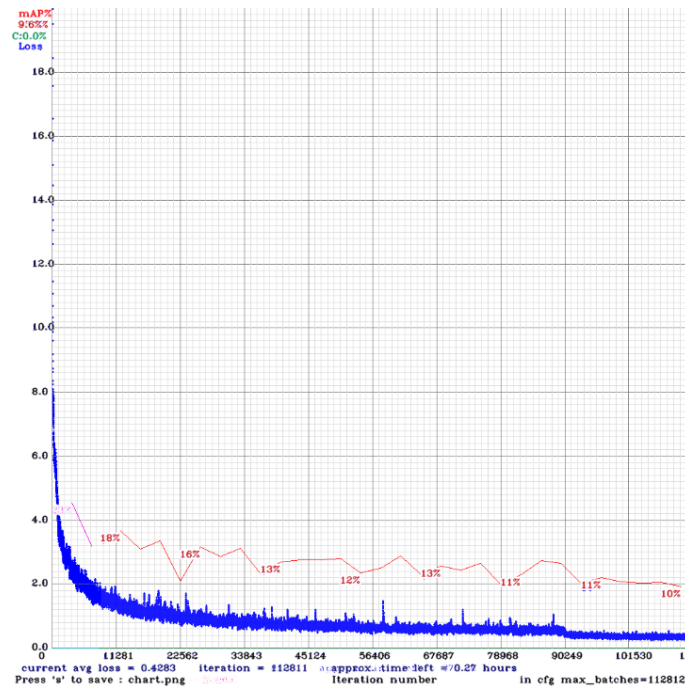


Figura 3.9 Entrenamiento de imágenes RGB + *Thermal* + *data augmentation* (Noche) utilizando la arquitectura de YOLOv4_4c

3.12 Comparativa de imágenes RGB + *data augmentation* (Noche) e imágenes RGB + *Thermal* + *data augmentation* (Noche)

La Tabla 4 se evidencia que el uso de las técnicas de *data augmentation* son beneficiosas para mejorar la precisión mejorando un 4.31% con la arquitectura YOLOv3, pero perdiendo un 29.67% para la clase ciclista. Mientras que el uso de un cuarto canal como la LWIR en escenarios nocturnos sigue siendo importante y sumado a la aplicación de *data augmentation* se mejoró un 26.9% respecto a la arquitectura YOLOv4 con *data augmentation*.

Otro dato importante es el incremento de precisión en un 11.15% y 8.58% respecto a YOLOv4 y YOLOv4_4C con los datos obtenidos en la Tabla 3; afianzando los beneficios de las técnicas de *data augmentation*.

Tabla 3.4 Comparación de imágenes RGB + *data augmentation* (Noche) y RGB + *Thermal* + *data augmentation* (Noche)

| AP% | Persona | Personas | Ciclista |
|-------------------------------|---------|----------|----------|
| RGB YOLOv3 | 20 | <1 | 30 |
| RGB DA YOLOv4 | 24.31 | 0.04 | 0.33 |
| RGB + Thermal DA YOLOv4_4C | 52.57 | 4.64 | 1.27 |

CAPÍTULO 4

Finalmente, presentaremos las conclusiones y recomendaciones que se obtuvieron luego de la evaluación de las arquitecturas YOLOv4 y YOLOv4_4C modificada usando los 8 escenarios propuestos en el capítulo anterior.

4. Conclusiones y recomendaciones

Las conclusiones y recomendaciones de este proyecto son descritas a continuación:

4.1 Conclusiones

- Al generar las imágenes con más de tres canales, previamente se debe evaluar el provecho que se obtendrá con el uso de el o los nuevos canales en las imágenes a utilizar, ya que en otro caso podría generar malos resultados y no se aprovecharía de forma correcta las características contenidas en las imágenes de más de tres canales.
- La arquitectura YOLOv4 posee mejores extractores de características que YOLOv3 en ambientes diurnos mientras que para ambientes nocturnos YOLOv3 es mejor bajo imágenes RGB.
- El uso del canal *Thermal* bajo condiciones nocturnas es sumamente beneficioso para la detección y clasificación de la clase persona al poder agregarle más *features* a la red como la temperatura que permite diferenciar a una persona del fondo. Este mismo beneficio se obtendría para objetos que posean temperatura.
- Los entrenamientos de la arquitectura YOLOv4_4C usando imágenes RGBT fueron realizados sin el uso de *transfer learning* ya que MS COCO posee solo imágenes RGB por lo que no conocen de la componente *Thermal* y no están inicializados los pesos para ese canal, y a pesar de esto superó en la precisión de detectar y clasificar en ciertos escenarios como el nocturno.

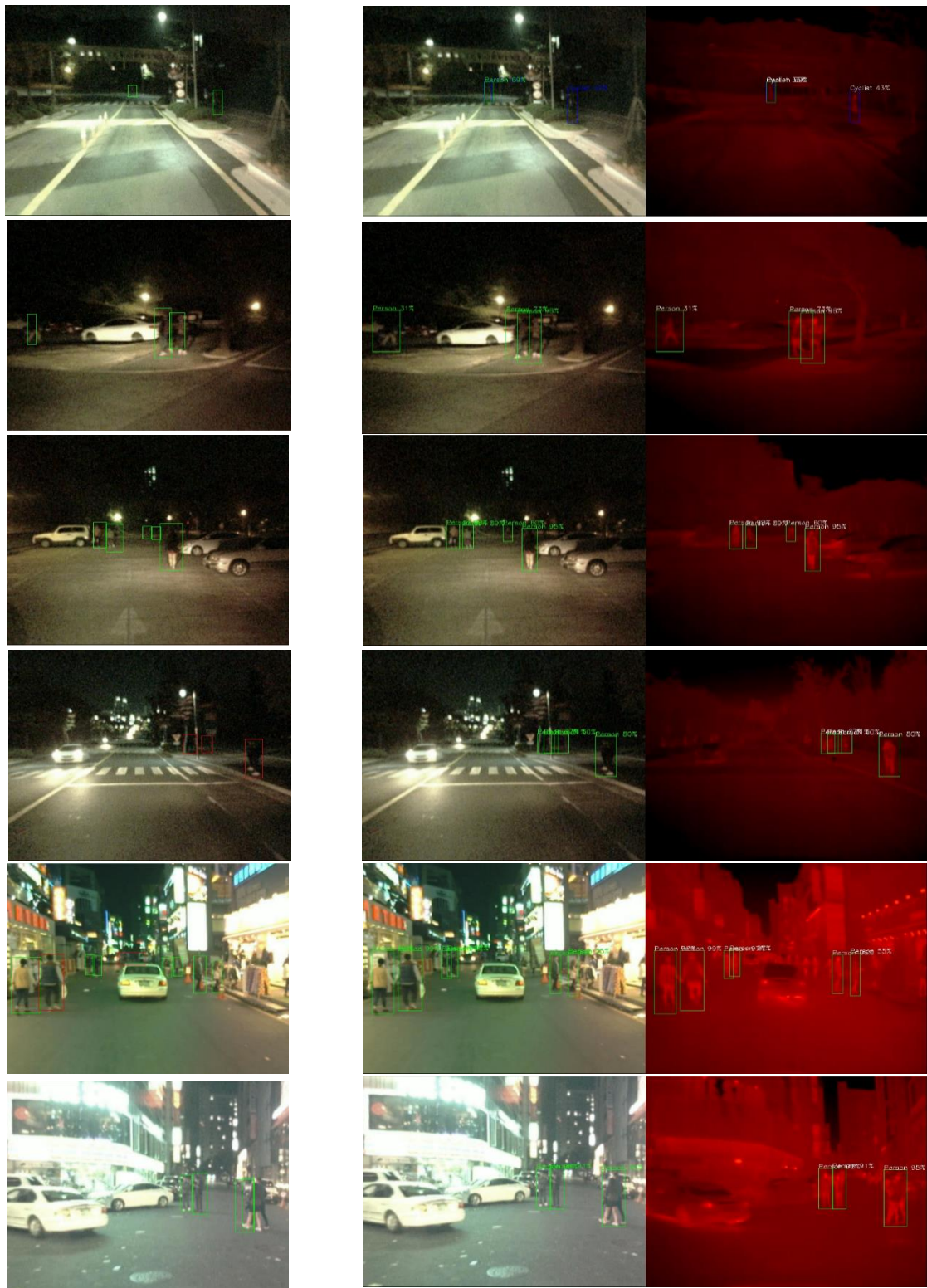
- Las técnicas de *data augmentation* aumentan significativamente la precisión para la clase con mayor número de objetos en el *dataset*, pero su comportamiento es aleatorio con clases con objetos de menor presencia en el *dataset*.
- Las técnicas de *data augmentation* generan un gran coste computacional dependiendo del número de técnicas a aplicar en el *dataset* en el entrenamiento de los modelos. Y con ello el tiempo de espera es más largo con respecto a no usar *data augmentation*.
- Al usar técnicas de *data augmentation* como crop y rotate, hay que considerar que las anotaciones se pueden sobresalir de la imagen generando errores al entrenar; por lo que esas anotaciones se las deben de recortar al tamaño de salida de la imagen final.

4.2 Recomendaciones

- Al usar aumento de datos hay que considerar que el tiempo y el costo computacional se incrementará de forma proporcional a la cantidad de datos que se obtengan al aplicar las diferentes técnicas de aumento de datos.
- Al compilar la arquitectura se debe considerar hacer una revisión previa de las banderas de compilación de la arquitectura ya que esto permitirá identificar si se ejecutará solo en el CPU o solo en la GPU.
- Cuando ocurra un corte de energía o una falla al momento del entrenamiento, se podrá retomar el entrenamiento de la arquitectura desde los últimos pesos y luego unir las gráficas para obtener una gráfica final sin perder el conocimiento adquirido previo al apagón. Se debe de ejecutar el mismo comando de entrenamiento, pero en la parte donde se usan los pesos de *transfer learning* (si es el caso) ubicar los “*last weights*” de la carpeta *backup*; antes de ejecutar el comando crear un respaldo de las gráficas (*charts*) para luego unir las usando un script o a través de una aplicación de diseño.
- Cuando se entrena la arquitectura modificada utilizando transfer learning los pesos que serán usados deben poseer un cuarto canal dado que si estos solo contienen tres canales se estaría realizando un entrenamiento erróneo.

- Al generar las imágenes con más de tres canales, previamente se debe evaluar el provecho que se obtendrá con el uso de el o los nuevos canales en las imágenes a utilizar, ya que en otro caso podría generar malos resultados y no se aprovecharía de forma correcta las características contenidas en las imágenes de más de tres canales.
- Ante las interrupciones durante el entrenamiento de cualquier modelo se recomienda hacer un respaldo de los mejores pesos y las gráficas (*Loss* y *Precisión*) antes de seguir entrenando la red ya que ambas se sobrescriben y se perdería esa información.
- Se recomienda retirar las imágenes con *data augmentation* del conjunto test o validación ya que la precisión disminuye en un porcentaje muy bajo (1-2%) y es lógico ya que estas técnicas se basan más para ayudar la convergencia correcta de los pesos impidiendo a la red memorizar.

Anexos

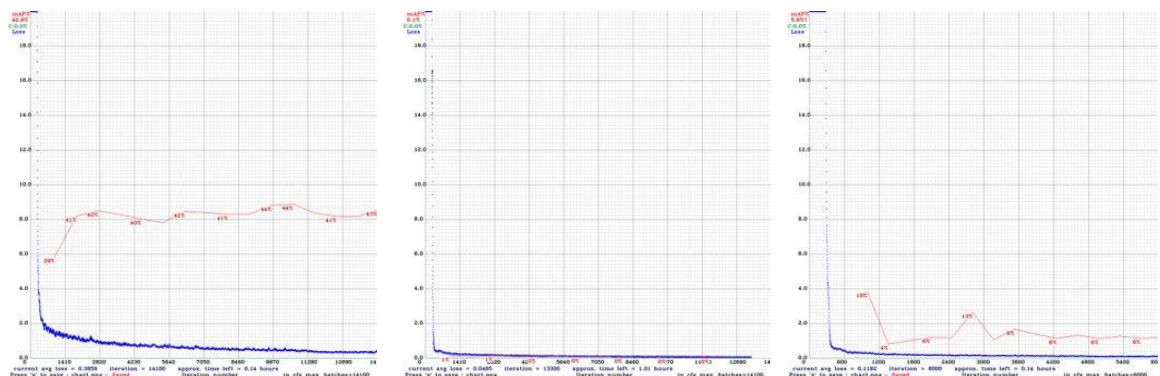


Anexo 1.1 Resultado de entrenamiento de RGB + *Thermal* + *data augmentation* (Noche) utilizando la arquitectura de YOLOv4_4C. (Izquierda) *Ground truth*, (centro) detección en RGB y (derecha) detección en *pseudocolor Thermal*

Apéndice

Se realizaron nuevos entrenamientos para determinar la validez del enunciado que se encontró en la literatura, en donde mencionan que el desbalanceo del número de objetos de las clases afecta a aquellas clases con menor número de objetos obteniendo menor porcentaje en relación con la de mayor número de objetos en la labor de detección y clasificación.

Se obtuvieron 9 nuevas graficas de los entrenamientos en donde usamos el mismos dataset de KAIST. Para las primeras 3 seleccionamos el conjunto de datos de día de forma idéntica al entrenamiento de la Figura 3.2; es decir, hicimos uso de transfer learning con los pesos de MS COCO y en los etiquetados solo se tomó una clase a analizar en específico y las demás no se las tomaba en cuenta para solo clasificar y detectar una clase en específico.

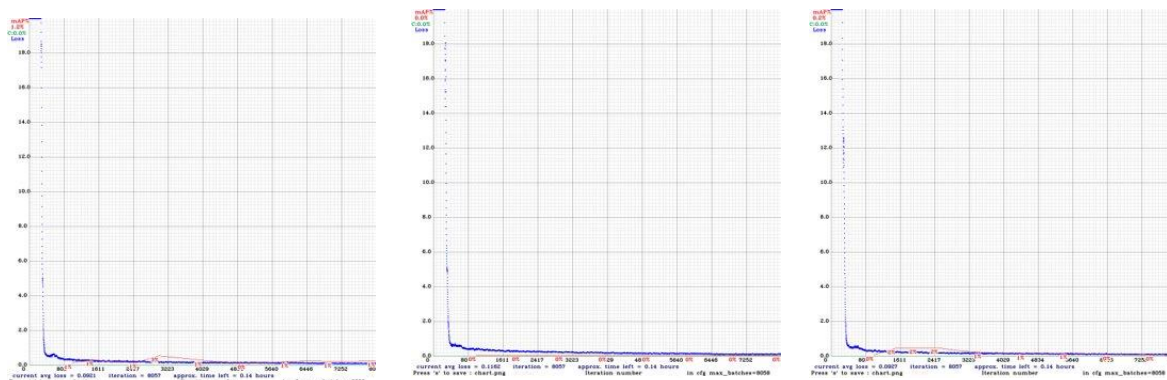


Apéndice 1.1 Resultado de entrenamiento de RGB (Día) utilizando la arquitectura de YOLOv4. (Izquierda) Persona, (centro) personas y (derecha) ciclista

En el figura del apéndice 1.1 podemos ver que para la clase persona se obtuvo una precisión máxima de 44% para personas 1% y ciclista 19%; lo cual no representa un gran cambio con los

resultados obtenidos en las precisiones de la Tabla 3.1; sin embargo, se puede mejorar en un 7.71% la clase ciclista que obtiene esa precisión en su primera evaluación, dándonos a entender que los pesos de MS COCO son los responsables de esta mejoría al poseer conocimiento de ciclistas en su base de datos y se analiza esta clase de forma aislada a las demás. Mientras que se pierde un 8.44% para la clase persona que presumiblemente es gracias al etiquetado de la clase personas que no se encuentra presente cuando ella se entrena generando un falso positivo en su detección y clasificación.

Para las siguientes 3 se toma como referencia el entrenamiento 3.6, seleccionando el conjunto de datos de noche; haciendo uso de transfer learning y la misma forma de analizar los etiquetados de los 3 entrenamientos previos.

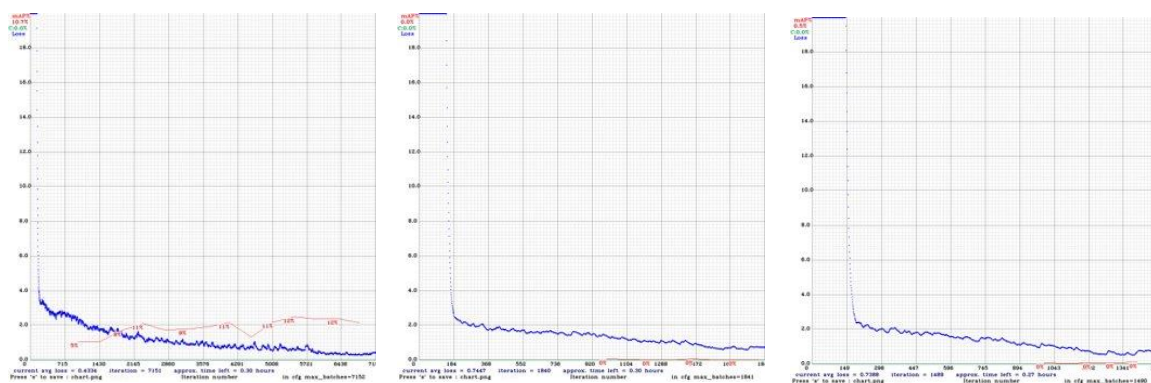


Apéndice 1.2 Resultado de entrenamiento de RGB (Noche) utilizando la arquitectura de YOLOv4.

(Izquierda) Persona, (centro) personas y (derecha) ciclista

En la figura del apéndice 1.2 podemos ver que para la clase persona se obtuvo una precisión máxima de 3% para persona, personas 0% y ciclista 2%; lo cual no representa un gran cambio con los resultados obtenidos en las precisiones de la Tabla 3.3; la clase persona disminuyó su precisión en un 10.16% aquí se repite el problema de los falsos positivos de la clase personas al no estar presentes las etiquetas en la detección y clasificación. En la clase personas se mantuvo el porcentaje a menor de 1% lo cual es muy bajo, y para ciclista se mejoró en un 1.31%.

Para las últimos 3 se toma como referencia el entrenamiento 3.7, seleccionando el conjunto de datos de noche; con la particularidad de no usar *transfer learning* y analizar solo las imágenes que poseen la clase a entrenar desechando del conjunto de train imágenes que solo contienen las demás clases.



Apéndice 1.3 Resultado de entrenamiento de RGB + *Thermal* (Noche) utilizando la arquitectura de YOLOv4_4C. (Izquierda) Persona, (centro) personas y (derecha) ciclista

En la figura del apéndice 1.3 vemos una precisión máxima de 12% para la clase persona, 0% personas y 0% ciclista. Estas dos últimas clases se vieron gravemente afectadas por el número de iteraciones tan bajo que tenían al existir pocas imágenes con las clases personas y ciclista, por lo que se recomienda usar las 6000 iteraciones como mínimo tal cual recomienda Darknet. Además, la clase persona disminuyó un 8% al no existir etiquetas de la clase personas ocasionando falsos positivos dando origen al decremento de la precisión.

Finalmente, podemos analizar que es correcto lo que menciona el estado del arte acerca del desbalanceo del número de objetos en las imágenes de cada clase, además que influye la poca congruencia del etiquetado de la clase personas generando presiones bajas en la detección y clasificación de persona y personas.

Referencias

- [1] J. Jiang, X. Fu, R. Qin, X. Wang and Z. Ma, "High-Speed Lightweight Ship Detection Algorithm Based on YOLO-V4 for Three-Channels RGB SAR Image", *Remote Sensing*, vol. 13, no. 10, p. 1909, 2021. Available: 10.3390/rs13101909 [Accessed 10 June 2021].
- [2] F. Rosenblatt, "Perceptron Simulation Experiments", *Proceedings of the IRE*, vol. 48, no. 3, pp. 301-309, 1960. Available: 10.1109/jrproc.1960.287598 [Accessed 11 June 2021].
- [3] Y. Wei, S. Tran, S. Xu, B. Kang and M. Springer, "Deep Learning for Retail Product Recognition: Challenges and Techniques", *Computational Intelligence and Neuroscience*, vol. 2020, pp. 1-23, 2020. Available: 10.1155/2020/8875910 [Accessed 15 June 2021].
- [4] F. Mazen and A. Nashat, "Ripeness Classification of Bananas Using an Artificial Neural Network", *Arabian Journal for Science and Engineering*, vol. 44, no. 8, pp. 6901-6910, 2019. Available: 10.1007/s13369-018-03695-5 [Accessed 15 June 2021].
- [5] A. Hernández-García and P. König, "Further Advantages of Data Augmentation on Convolutional Neural Networks", *Artificial Neural Networks and Machine Learning – ICANN 2018*, pp. 95-103, 2018. Available: 10.1007/978-3-030-01418-6_10 [Accessed 15 June 2021].
- [6] Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2021). Transformers in Vision: A Survey. *arXiv preprint arXiv:2101.01169*.
- [7] O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., ... & Walsh, J. (2019, April). Deep learning vs. traditional computer vision. In *Science and Information Conference* (pp. 128-144). Springer, Cham.
- [8] Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*.
- [9] S. Hwang, J. Park, N. Kim, Y. Choi and I. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baseline", *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. Available: 10.1109/cvpr.2015.7298706 [Accessed 2 August 2021].

- [10] Z. Song, Q. Chen, Z. Huang, Y. Hua and S. Yan, "Contextualizing object detection and classification", CVPR 2011, 2011. Available: 10.1109/cvpr.2011.5995330 [Accessed 16 June 2021].
- [11] Kiaee, Nadia & Hashemizadeh, Elham & Zarrinpanjeh, Nima. (2017). A SURVEY ON OBJECT DETECTION AND CLASSIFICATION METHODS.
- [12] P. L. Suárez, A. D. Sappa and B. X. Vintimilla, "Infrared Image Colorization Based on a Triplet DCGAN Architecture," 2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2017, pp. 212-217, doi: 10.1109/CVPRW.2017.32.
- [13] D. Wierzbicki, "Multi-Camera Imaging System for UAV Photogrammetry", Sensors, vol. 18, no. 8, p. 2433, 2018. Available: 10.3390/s18082433 [Accessed 17 June 2021].
- [14] Perez, L., & Wang, J. (2017). The effectiveness of data augmentation in image classification using deep learning. arXiv preprint arXiv:1712.04621.
- [15] P. L. Suarez, A. D. Sappa and B. X. Vintimilla, "Learning image vegetation index through a conditional generative adversarial network," 2017 IEEE Second Ecuador Technical Chapters Meeting (ETCM), 2017, pp. 1-6, doi: 10.1109/ETCM.2017.8247538.
- [16] Realpe, Miguel & Godoy, Axel & Reyes, José. (2019). Uso de software de código abierto para fusión de imágenes agrícolas multiespectrales adquiridas con drones. 10.18687/LACCEI2019.1.1.254.
- [17] Yun, S., Han, D., Oh, S. J., Chun, S., Choe, J., and Yoo, Y., "CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features", <https://arxiv.org/abs/1905.04899> , 2019.
- [18] Trigás Gallego, M. (2012). Metodologia scrum.
- [19] J. Burg, The science of digital media. Upper Saddle River, NJ: Prentice Hall/Pearson Education, 2009.
- [20] Lowe, D. (2021). Distinctive Image Features from Scale-Invariant Keypoints. Retrieved 8 July 2021, from

- [21] Apple. Official Apple Support. (n.d.). https://support.apple.com/kb/sp705?locale=en_US.
- [22] Apple. Official Apple Support. (n.d.). https://support.apple.com/kb/SP832?locale=en_US.
- [23] An AgEagle company. MicaSense. (n.d.). <https://micasense.com/es/sistema-camara-dual/>.
- [24] Nataprawira, J., Gu, Y., Asami, K., & Goncharenko, I. (2020). Pedestrian Detection in Different Lighting Conditions Using Deep Neural Networks. *IICST*.

*Repositorio del código usado en el proyecto

https://github.com/axelauza97/yolov4_4c