UNIVERSIDADE D
COIMBRA

Disa Alexandra Queiroz Palma

Enhancing Indoor Human Detection: A Comprehensive Study of YOLOv5 algorithm with Thermal Imagery

Thesis submitted to the University of Coimbra in fulfillment of the requirements for the Master's Degree in Biomedical Engineering specialization in Biomedical Instrumentation, under the scientific supervision of Professor Cristiano Premebida and presented to the Department of Physics.

September of 2023

FCTUC FACULDADE DE CIÊNCIAS
E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Enhancing Indoor Human Detection: A Comprehensive Study of YOLOv5 Algorithm with Thermal Imagery

Disa Alexandra Queiroz Palma

September, 2023

Enhancing Indoor Human Detection: A Comprehensive Study of YOLOv5 Algorithm with Thermal Imagery

Supervisor:

Professor Cristiano Premebida

Co-Supervisor:

Pedro Conde, PhD Student

Jury:

Professor Paulo José Monteiro Peixoto

Professor Cristiano Premebida

Professor Pedro Mariano Simões Neto

João Luís Ruivo Carvalho Paulo, PhD

Dissertation submitted in partial fulfillment for the degree of Master of Science in Biomedical Engineering.

September, 2023

Acknowledgements

Começo por agradecer ao Professor Cristiano Premebida pelas oportunidades, orientação e grande conhecimento. Ao Pedro Conde que foi essencial para o decorrer deste trabalho. Agradeço as horas de reunião, paciência e todos os conselhos. A todos no laboratório de Mecatrónica do ISR pela companhia e apoio nesta fase final.

À minha mãe, Ana Paula, que lutou para garantir que as filhas teriam um futuro melhor. À minha irmã, Celina, que foi a primeira da família a ir para a universidade. O meu percurso foi mais facilitado porque alguém desbravou terreno primeiro. Ao nosso gato, Edu. É a vocês e unicamente a vocês a quem dedico esta tese.

A Biologia. À Ana Albuquerque, à Rebecca Pilzecker, à Inês Santana e ao José Ferreira. Coimbra tornou-se uma Casa, porque vocês fizeram dela uma Casa. Estes cinco anos de Engenharia Biomédica nunca teriam sido um sucesso sem ter amigos. À Beatriz Dias, que desde o início que digo que viria parar aos meus agradecimentos de tese. Aqui está. Obrigada pela tua grande amizade e incondicional apoio em todos os momentos. Um bolinho não chega. À Matilde Palmeira, à Lídia Faria, à Laura Ferreira, à Francisca Afonso, à Raquel Gonçalves, à Joana Antunes e ao Nuno Rodrigues pelas explicações, resumos, sushi e, sobretudo, pela grande amizade. Obrigada por tornarem tudo isto bem mais fácil. Ao Miguel Gomes e à Joana Leiria (e, novamente, à Beatriz Dias), por todas as conversas aleatórias, filosóficas, políticas, por todos os conselhos de vida e académicos. Não podia ter tido mais sorte. Agradeço a todos - professores, colegas e amigos - que se cruzaram comigo nas salas e nos corredores do Departamento de Física ao longo destes últimos anos. Em especial à Beatriz Negromonte, Afonso Ávila, Ema Macedo, Mariana Letra e Clarisse Henriques.

À ANEEB. Obrigada por me arrancarem de Coimbra.

À Academia de Coimbra. À Associação Académica de Coimbra. A minha verdadeira Casa e Universidade. O meu grande amor e também o meu grande desgosto. O curso podia ser feito em qualquer lado, mas a vida que vivi cá nunca poderia tê-la noutro sítio. Foi graças ao CIAAC, ao NEDF, à Secção de Jornalismo e a tantas outras coisas nas quais me enfiei, que tive a oportunidade de desenvolver projetos, colocar em prática as minhas ideias e conhecer grandes amigos. Um especial agradecimento ao Paulo Nogueira Ramos e à Emília Oliveira.

A todos vós, o meu maior **F-R-A!**

This work was partially funded within the scope of the project ULTRABOT – CENTRO-01-0247-FEDER-072644 with funds from SII&DT - I&D Empresarial - Projetos de I&D em Copromoção para Territórios do Interior, by Autoridade de Gestão do CENTRO2020.

Resumo

A deteção de objectos tem uma variedade de aplicações: pode ser usada para auxiliar na agricultura, na deteção de cancro, condução autónoma, robótica ou a identificar intrusos numa propriedade privada. A deteção de objetos tem tido um sucesso significativo com imagens RGB. Contudo, a deteção nestes dados é prejudicada em condições de fraca luminosidade e exige um uso substancial de memória. Uma solução consiste no uso de imagens térmicas, uma vez que requerem menos espaço e são mais adaptáveis a ambientes de luminosidade variável. Deste modo, esta tese explora a aplicação de YOLOv5 num dataset térmico de ambiente interior - o dataset-alvo - com transferência de conhecimento e fine-tuning que correspondem a duas estratégias de aprendizagem profunda que se debruçam sobre a falta de dados e redução do tempo de treino, usando conhecimento aprendido anteriormente em datasets similares e que serão posteriormente aplicados num dataset-alvo. Neste estudo, um dataset-alvo é sujeito a fine-tune com quatro modalidades pré-treinadas: dois datasets RGB (COCO e a versão RGB do dataset-alvo), um dataset térmico (FLIR) e, por fim, a versão cinzenta do dataset-alvo. Os resultados foram comparados a um treino Controlo que se refere aos resultados de treino do dataset-alvo. Os fine-tunes do dataset-alvo são submetidos a várias condições: diferentes taxas de aprendizagem, técnicas de augmentação de dados, congelamento de camadas e uso de diferentes optimizadores (ADAM e SGD). Esta investigação concluiu que pré-treino do COCO atingiu o maior valor de mAP@0.5 independentemente das condições de treino, ultrapassando o formato RGB do dataset-alvo e o FLIR. Este estudo sugere que isto pode acontecer porque a sua dimensão e informação diversificada levam a uma maior generalização. Apesar de outros parâmetros terem impacto e ajudarem a melhorar os resultados, o tamanho do dataset e a diversidade que contém foram as variáveis com a maior influência.

Palavras-chave: detecção de objectos, transferência de conhecimento, aprendizagem profunda, YOLO, imagens térmicas.

Abstract

Object detection has a wide range of applications: it can be used to assist in agriculture, to help detect a mass for cancer diagnosis, enable autonomous driving, robotic perception, or help against home intruders. Object detection has shown significant success with RGB data. However, this type of data does not operate well in poor lighting conditions and demands a substantial amount of storage. One solution could be the use of thermal images, which require less space and are more adaptable to varying luminosity conditions. Therefore, this thesis explores the application of YOLOv5 with transfer learning and fine-tuning in a thermal indoor dataset (the target dataset), which correspond to deep learning strategies that help tackle the lack of data information and reduce training costs by using knowledge learned previously from similar datasets and applying it to a target dataset. In this study, the target dataset is fine-tuned with four pre-training modalities: two RGB datasets (COCO and the RGB format of the target dataset), one thermal dataset (FLIR), as well as the Grayscale format of the target dataset. The results are compared to the training results of Control, which refers to training the target dataset from scratch. The target dataset is fine-tuned in a variety of conditions, including varying learning rates, data augmentation techniques, freezing layers, and SGD and ADAM optimizers. This investigation concluded that using COCO for the pre-training of the model achieves the highest mAP@0.5 independently of its training conditions, surpassing the RGB format of the target dataset and FLIR. This study suggests that it may be because its dimension and diverse information lead to a greater generalization. Although the other parameters have an impact and can help enhance results, the dataset size and the amount of diversity it contains were the variables with the most influence.

Keywords: Object detection, transfer learning, deep learning, YOLO, thermal images.

"*You Only Live Once.*"

— Everyone in 2015, except for Joseph Redmon.

Contents

List of Acronyms

ADAM Adaptive Moment Estimation.

AP Average Precision.

BCE Binary Cross-Entropy.

CNN Convolutional Neural Network.

CSP Cross Stage Partial.

DBN Deep Belief Network.

DL Deep Learning.

DNN Deep Neural Network.

DTL Deep Transfer Learning.

FPN Feature Pyramid Network.

FPS Frames Per Second.

GCD Greatest Common Divisor.

HOG Histogram of Oriented Gradientes.

IoU Intersection over Union.

ISR Institute of Systems and Robotics.

LWIR Long-Wavelength Infrared.

mAP Mean Average Precision.

NMS Non-Maximum Suppression.

PAN Path Aggregation Network.

SGD Stochastic Gradient Descent.

SOTA State of the Art.

SPP Spatial Pyramid Pooling.

SPPF Spatial Pyramid Pooling Fast.

SSD Single Shot Multibox Detector.

SVM Support Vector Machine.

TIR Thermal Infrared.

UAV Unmanned Aerial Vehicle.

YOLO You Only Look Once.

List of Figures

List of Tables

1 Introduction

In recent years, the rapid advancement of artificial intelligence, computer vision, pattern recognition, and deep learning (DL) technologies have led to a new era for artificial perception systems, revolutionizing the way such systems have an impact on real-world applications. Object detection - the field responsible for the localization and classification of objects from images or videos -, the theme explored in this thesis, is one of the most notorious areas belonging to computer/machine vision and has gained significant attention from the AI/ML community as well as from the robotics and intelligent vehicles community. 2D and 3D sensory-based object detection has a wide range of applications, including security surveillance [48], robotic vision applied to agriculture [46], but also in medicine allowing for more rapid diagnosis [56], enabling autonomous driving [1], or even the task of license plate recognition [5]. Object detection has through the years, undergone massive development.

With the development of DL models, these complex and large architectures have been applied to object detection. Right now, the most developed state-of-the-art (SOTA) object detectors have DL models as their backbone to extract, localize, and classify objects.

Object detection can be applied to various types of data. It has a wide application in RGB images and video frames, but more recently it has been studied the application of object detection in thermal images that are used in a wide range of scenarios such as the detection of animals for population control purposes [52] and the detection of fire [47], as well as human detection for space managing [12] or detection of intruders in private properties [68].

1.1 Motivation

Object detection is very important since we are going towards a society that seeks automation for repetitive and dangerous tasks, and with the continuous exponential evolution of our algorithms and capacity, we have been able to open new opportunities for robots and other robotic systems to engage in more complex and demanding action. Object detection allows that. The growing use cases related to object detection can be explained because not only the algorithms (and respective tools) are more accessible but, on the other hand, the technology that permits the capture of images and videos (sensors or cameras) is becoming more affordable [51]. Computers power *i.e.*, hardware

capability, has also become more powerful [53].

Whilst object detectors have been improving their performance in RGB, achieving high accuracies [77][79], some problems have yet to be explored when these algorithms are applied solely to thermal images. So, why consider working with thermal images when RGB images have already delivered promising results? In situations with low visibility and inconsistent conditions, thermal cameras are shown to be the best option. Besides, in day-to-day use, the utilization of RGB imagery consumes substantial storage as opposed to thermal images. There have been works on image fusion that combine color and thermal [36], which have great performances in tackling the problem of poor visibility; however, they do not solve the substantial need for storage. Also, pairs of RGB and thermal images might not always be available since they require equipment for two different modalities, and there might exist a temporal misalignment between them that compromises the performance. Having said that, it is important to continue research that focuses on thermal imagery, paired with transfer learning techniques to improve the generalization of these approaches.

Object detection has diverse domains such as edge detection [57], face detection [74], or pedestrian detection [54] [78]. This work will tackle object detection applied to human detection in an indoor environment through thermal imagery. With that, in this thesis we intend to explore a field that has yet a lot to unfold.

1.2 Objectives

This work aims to investigate the performance of YOLOv5 in human detection applied to thermal images in indoor scenarios. In a nutshell, the main objectives can be outlined as:

- Contribute with a new annotated thermal dataset;
- Study the performance of YOLOv5 with thermal imagery;
- Study the performance of YOLOv5 in fine-tuning a thermal dataset with different pre-trained datasets, such as RGB, FLIR, Grayscale, and COCO, and compare with its baseline (i.e. no transfer learning);
- Study the impact of frozen layers, data augmentation, and the difference between the application of SGD and ADAM optimizers;
- Determine the optimal training conditions for implementing YOLOv5 on thermal images.

1.3 Thesis outline

This thesis is organized to start with a broad view of object detection before narrowing towards the main objectives of this work. It starts in Chapter 2 with the state-of-the-art (SOTA) exploring

the object detection applications and problems, the various algorithms, and their performances. It continues with a summary of the problem of object detection applied to thermal images and the various techniques, as well as the application of transfer learning in DL. Finally, it finishes by discussing relevant related work that explores the various works in the domain of human detection and thermal imaging. In Chapter 3, this study deepens into the methodology by explaining the datasets used and studied, evaluation metrics, as well as YOLOv5's architecture, hyperparameters, and experiment design. Chapter 4 corresponds to the presentation of the results through tables and figures depicting the values of precision, recall, mAP@0.5, mAP@0.5:0.95, and F1-score through the various scenarios studied and respective discussion. Finally, Chapter 5.1 summarizes the work and conclusions taken from it, while also discussing some possible future research directions.

2 State of The Art

This Chapter offers a comprehensive overview of the current SOTA in the field of object detection. It goes into various architectural approaches, presenting a chronological progression of developments that have culminated in the current state of the field, mainly in thermal imaging. It is intended to start with a broad perspective, develop into a more detailed exploration and provide insights into the evolution of object detection techniques and some of the related works.

2.1 General view of object detection

Object detection is a field from computer vision and robotic perception with many applications, from autonomous driving [1] to healthcare diagnosis [56], where it facilitates autonomous early cancer detection. Moreover, with the emergence of Deep Neural Networks (DNNs) and their high accuracy in multiple tasks paired with the development of GPU power, the interest in this field has grown in the past few years since its capabilities have been demonstrated [27].

Object detection encompasses two core tasks: localization and classification. First, the algorithm has to find an object and then classify it. Both of these tasks are difficult because they highly depend on the quality of the camera or sensor, the light conditions, blur, pose, the number of objects, and how overlaid they are. These are some of the challenges addressed by recent research in the field of object detection with DL, where the pipeline traditionally is organized in three stages: informative region selection, feature extraction, and classification [78].

SOTA object detectors use predominantly DNNs, and their backbone is mainly Convolution Neural Networks (CNNs) whose main task is related to the extraction of features. Their complex architectures are able to learn a higher quantity of features with high complexity [27]. There are two types of object detectors: one-stage and two-stage. Two-stage detectors have two steps: propose candidate object bounding boxes and then extract the features from the candidates; while the one-stage detectors do it directly from the input images [27]. Object detection faces some challenges, such as rotated images, scale, and occlusion of objects, among others, that diminish the models' performance.

The history of object detection starts with hand-crafted techniques like Viola Jones Detectors [71] and HOG (Histogram of Oriented Gradients) Detector [9]. The first one used the sliding window

method and differing scales of an image to find a human face. The second was extremely important to work on feature invariance and non-linearity, mainly in pedestrian detection. However, in 2010, the object detection field introduced CNNs that rapidly demonstrated the capability to learn more complex features. As a result, in 2014, the researchers in [17] proposed Regions with CNN (RCNN).

CNNs (represented in Figure 2.1) are a type of DL algorithm specifically used for the tasks of classification [8] and regression [78] that leverage the extraction of features of different levels from an input image [53]. CNNs were created to solve some of the challenges presented in classification problems with images related to the amount of weight each neuron in hidden layers would receive. According to [19], CNNs are based on three main ideas:

1. **Sparse interactions:** reduction of parameters' storage and computation power required by only connecting the neurons to a small and meaningful window of the input image through the use of kernels and filters. This boosts efficiency.

2. **Parameter sharing:** closely related to sparse interaction, parameter sharing allows for the detection of patterns in different parts of the input. By using the same weights and biases for the calculations across the input, it is possible to detect invariance, and, thus, features. The network does not need to learn from scratch in every position, which saves memory space.

3. **Equivariant representations:** is related to the idea that a variation of the input will necessarily create the same variation in the output. This permits the detection of the same features in different parts of the output.

There are also three stages in a typical CNN:

1. **Convolution Stage:** stage where multiple convolutions are applied to the input data. It consists of a set of operations called convolution that will slide over the input to capture different features and patterns. The output is a set of linear activations.

2. **Detector Stage:** the linear activations from the previous stage will go through an activation function. This stage allows the network to detect more complex features and find relationships between data.

3. **Pooling Stage:** in this stage, only the essential information is kept by downsampling. It performs a series of calculations that allows the retention of the maximum value in each small window considered, lowering the necessity of high memory requirements and computation costs.

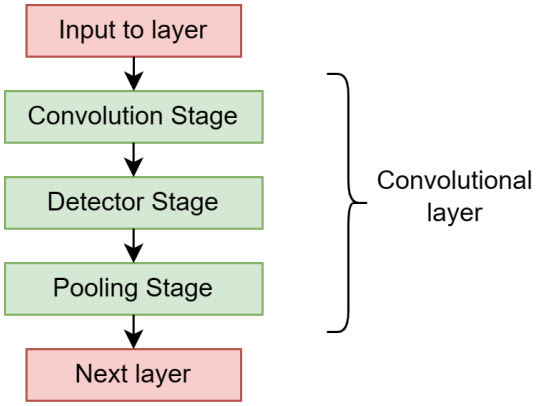


Figure 2.1: CNN's basic architecture. Based on [19].

In the next sections, in this thesis it will be explored SOTA on object detectors.

2.1.1 Object detectors

Generic object detectors can be categorized into two groups [78]: region proposal-based frameworks and regression/classification approaches. On the first one, it is possible to find two-stage detector-type algorithms, like RCNN or Faster RCNN [78]. Two-staged detectors consider the classical object detection pipeline: first, they generate region proposals, followed by the classification of each proposal into classes. On the second one, there are algorithms like YOLO [58] and SSD [42], which are considered one-stage detectors. The types of detectors are illustrated in Figure 2.2.

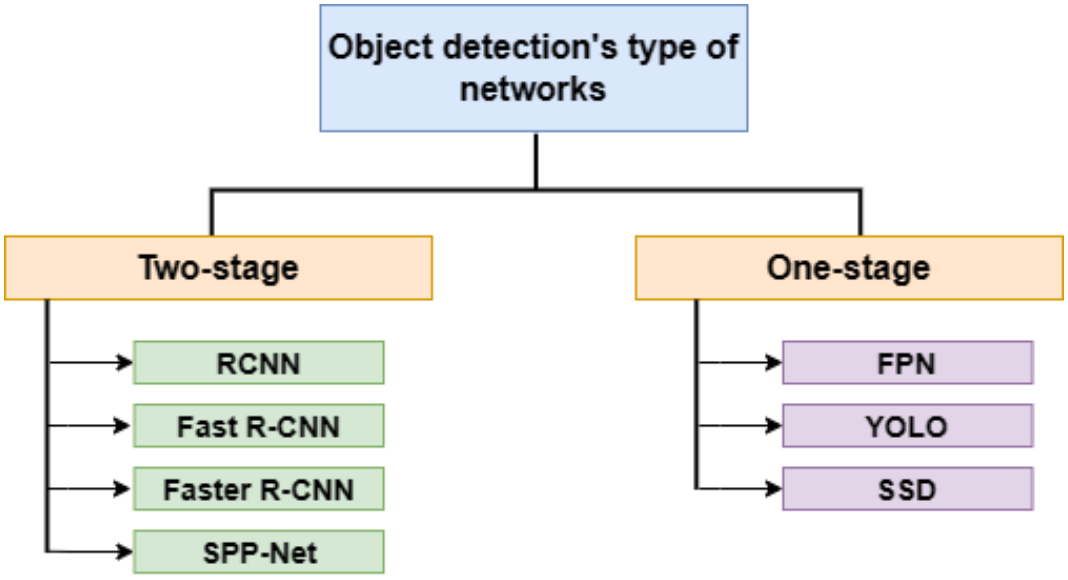


Figure 2.2: Types of object detection detectors referred to in this chapter. Adapted from [1].

In 2014, the Region-based Convolutional Neural Network (RCNN) [17], which is pictured in Figure 2.4a, was proposed as an algorithm that uses selective search (region proposal algorithm that generates proposals of regions based on pixel intensities), chooses and extracts 2000 region proposals from an image, and uses them as inputs. All these regions can have different sizes and

ratios that will be later normalized to the same dimensions.

Shortly after the publication of RCNN, Spatial Pyramid Pooling Networks (SPPNet) [22], represented in Figure 2.3, was presented and allowed for the first time to submit any image to the CNN that did not require a fixed size. Because of this, the algorithm was able to generate a feature map from an entire image, which made the algorithm faster. However, it has some drawbacks; for example, it only fine-tunes the last layers, and the training is multi-staged. All of these were solved by Fast-RCNN [16]. Fast-RCNN, depicted in Figure 2.4b, is an RCNN and SPPNet improvement as it does not use the regions as inputs but uses the image they derive from instead. The feature map that results will create the region proposal [1]. Fast-RCNN increased the VOC07 average precision (of around 20 classes) from 58.5% to 70%, even though the detection speed remained low [79].

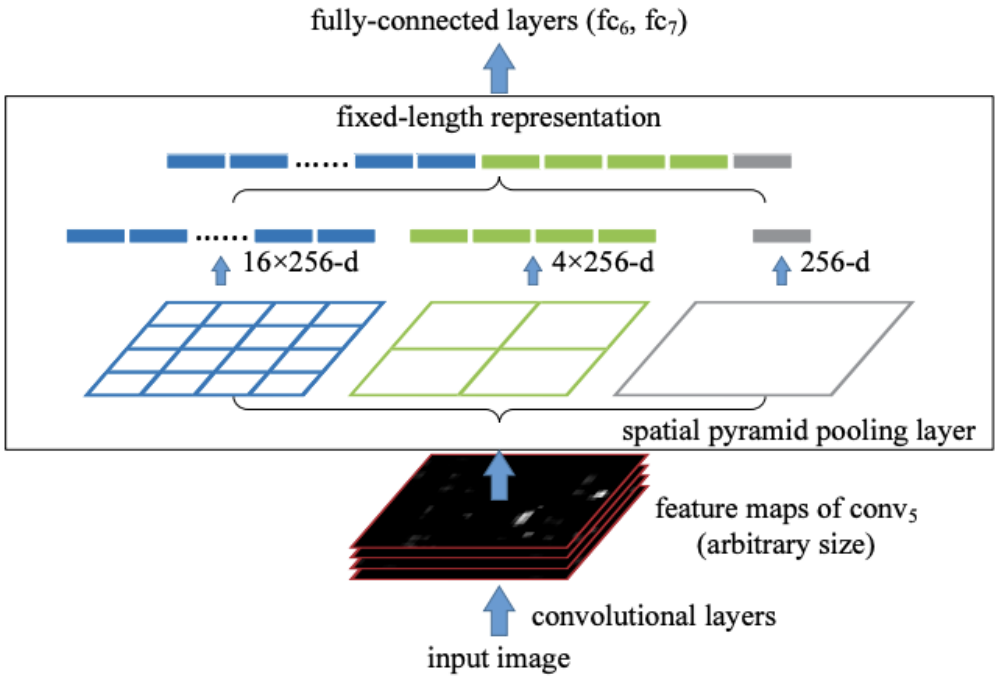


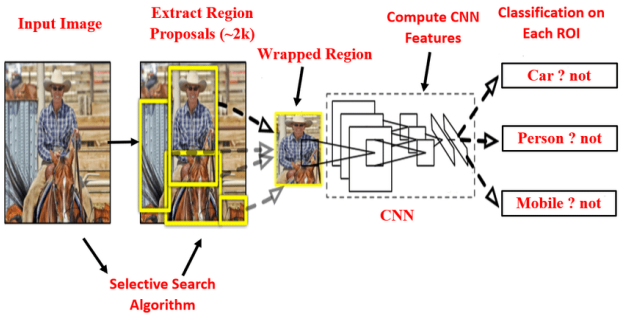
Figure 2.3: SPPNet's architecture. Taken from [22].

Feature Pyramid Networks (FPN) were introduced in 2017 in [38] and have been the basis for a lot of algorithms even today, such as YOLO, Mask-RCNN [21] and RetinaNet [39]. FPN allowed for building high-level semantics at all scales by introducing communication not only vertically but also horizontally with a top-down architecture that uses lateral connections that are illustrated in Figure 2.5b. This communication increased generalization as the new feature maps generated considered information from both low-level and high-level features.

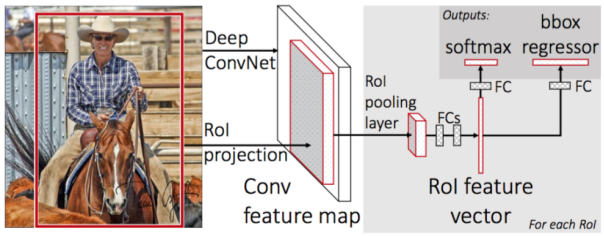
However, due to the continuous slow speed and complexity, a new group of object detectors was created: one-stage detectors or Regression/Classification-based detectors. They are a group of architectures created to diminish the high computational costs associated with scanning an entire region. The main difference is that the algorithms locate and classify directly from the pixels

without the need to create region proposals. They scan the entire image only once. Some of the most famous object detectors in this category are YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector).

YOLO is an object detection algorithm presented in 2016 with the concept of "You Only Look Once" that looks at object detection as a single regression problem from the cells to create a bounding box and predict class probabilities. Since then, there have appeared around eight more refined versions of YOLO that compete with the best detectors. After the third version, YOLO stopped being developed by its original creator, Joseph Redmon, for ethical reasons [29]. Consequently, YOLOv4 [6] was already developed by Alexey Bochkovsky and later versions were developed by Ultralytics [28]. In this thesis the focus will be on YOLOv5; this architectural design incorporates numerous elements from preceding algorithms, which will be further discussed in section 3.3.



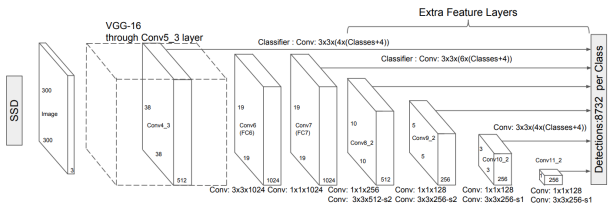
(a) RCNN's architecture taken from [17].



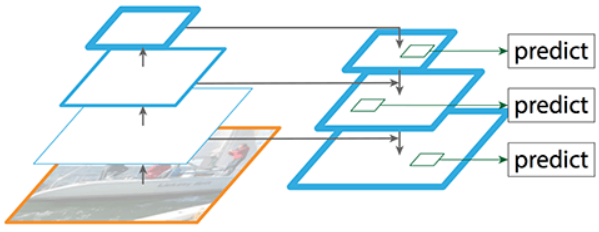
(b) Fast-RCNN's architecture taken from [16].

Figure 2.4: RCNN and Fast-RCNN's architectures.

SSD (illustrated in Figure 2.5a), emerges as an alternative to YOLO [78] and uses the VGG16 architecture as its backbone. To be able to predict the offsets for the bounding boxes at different scales and ratios, SDD adds feature layers at the end of the network (similar to YOLO). This algorithm receives already extracted feature maps, then calculates the confidence for every image and picks the top 200 predictions per input. It seeks to enhance detection speed, by refraining from re-sampling pixels or features for bounding box hypotheses. [43].



(a) SSD's architecture taken from [42].



(b) FPN's architecture taken from [38].

Figure 2.5: Architecture of two of the SOTA detectors described.

In [67] all versions of YOLO until 2023 are reviewed since 2018. Observing the table, YOLOv5

has one of the best performances with 55.8%, only outdone by Scaled-YOLOv4 by 0.2 percentage points and YOLOv7 by two percentage points. The results can be seen in Table 2.1.

Table 2.1: YOLO versions and their performance. Adapted from [67].

Version	**Date**	**mAP@0.5**
YOLOv3	2018	36.2
YOLOv4	2020	43.5
YOLOv5	2020	55.8
Scaled-YOLOv4	2021	56.0
YOLO-R	2021	55.4
YOLO-X	2021	51.2
YOLOv6	2022	52.5
YOLOv7	2022	56.8
YOLOv8	2023	53.9
YOLO-NAS	2023	52.2

In [15] the authors also compare YOLOv5 with more recent versions such as YOLO-X, YOLO-R, and YOLOv7 during a live tracking of players during a football match. YOLOv7 has major changes, such as the introduction of multiple heads in its architecture: a lead head that releases the result and an auxiliary head that helps with middle-layer training. Observing the results among the various versions, YOLOv5 has close SOTA results, only outstanding in terms of FPS (frame per second) rate, and is close in accuracy to YOLOv7, even though this one outperforms the versions already mentioned.

2.2 Thermal imaging

Thermal imaging has a wide range of uses, from military [31] to medicine [56], from disease detection in animals [52] to safety by detecting people's positions in underground mines [37]. Several literature reviews have been done to explore thermal imaging applications [11] [65] [10].

In [10] it is possible to find an extensive list of infrared datasets available publicly. Among them are two famous large thermal datasets: *Teledyne FLIR ADAS Thermal Dataset* (FLIR dataset) and the KAIST Multispectral Pedestrian dataset [7]. However, they are only outdoor datasets [61]. Exclusive indoor datasets, like TIMo [59] are scarce; the majority are mixed with outdoor images, such as the ASL-TIR dataset [55].

Thermal cameras are known as long-wavelength infrared (LWIR) cameras and are capable of detecting electromagnetic radiation between 8 to 12 μm. This radiation comes from the body

temperature of an object, forming an image in a temperature spectrum. In the case of humans, since the body temperature is constant independently from the environment, they are easily distinguished from the background. Thermal images present some advantages, *e.g.*, they are not dependent on illumination variations or weather conditions [65] and can operate well in the dark. Since thermal cameras are the few that can manage to operate in variations of lighting, they are widely used to detect the presence of people usually in poor background conditions [3]. Another advantage is the computational cost associated with it and the amount of memory needed to store it: thermal images do not have as much information and detail as RGB images, which can be more adequate in certain contexts.

However, this type of image presents some challenges; for instance, when a body with a high temperature is around a reflective surface, its image is also reflected, which might confuse object detector algorithms as they are with the same level of brightness and shape [3]. Hot air due to the sun is also a cause of blur in the middle of the day or when the camera is being used for long lengths of time and its temperature increases as well, affecting the capacity to detect thermal images. These also offer less detail relative to visible-light cameras and have lower resolutions; they can perform poorly in environments where there's low thermal contrast between people and their surroundings [34].

Multiple techniques have been applied in human detection with thermal images, such as background subtraction [64], saliency maps [14], as well as HOG [33]. Later, with the development of DNN models and their success with RGB images, these architectures started to be used for object recognition in thermal images; however, they have not attained the same SOTA levels relative to their RGB counterparts [65].

Specifically for object detection in [2], several object detectors were developed and tested, such as Faster RCNN [49], EfficientDet [32] (illustrated in Figure 2.7) and Domain Adaption Framework [50]. The method that achieved the highest mAP@0.5 was Faster RCNN with 77.1%, applied to the FLIR dataset. Domain Adaptation Framework achieved 67.36%.

Domain Adaptation Framework [69] is a technique based on transfer learning between RGB images and thermal images. It transfers low-level features through a method called generative adversarial network (GAN) and, in [50], it is applied to the KAIST Multi-Spectral and FLIR datasets. Still, in the domain adaptation, it was introduced a multi-stage block-wise architecture in EfficientDet, which increased the accuracy and efficiency of object detection in thermal images and also stands on the approach of transferring information learned with RGB imagery to be applied in the thermal domain.

Additionally, some works incorporate both DL techniques as well as the previously mentioned classical methods, such as utilizing saliency maps to enhance human detection by accentuating distinct regions within thermal images that may be different in color, orientation, or depth. Training

with Fast RCNN in [14] the detection in the KAIST Multispectral Pedestrian dataset has achieved a mAP@0.5 of 0.655, but with the saliency map technique, it went up to 0.676. An example of saliency maps applied to thermal images can be seen in Figure 2.6.

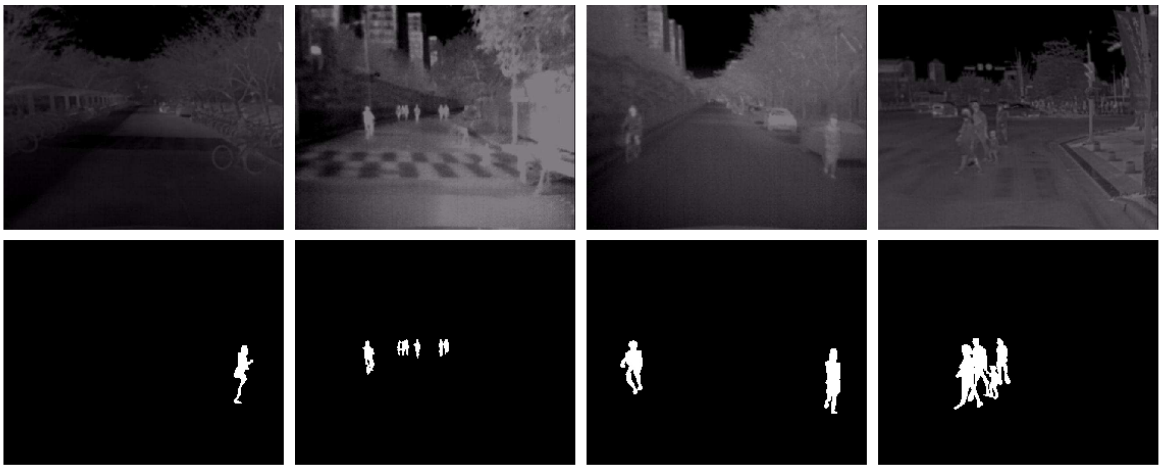


Figure 2.6: Example of saliency maps extracted from thermal images for human detection. Taken from [14].

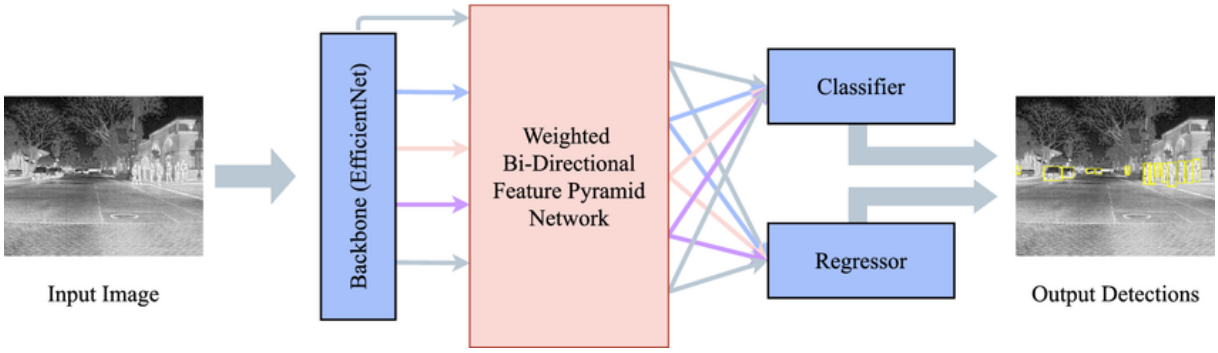


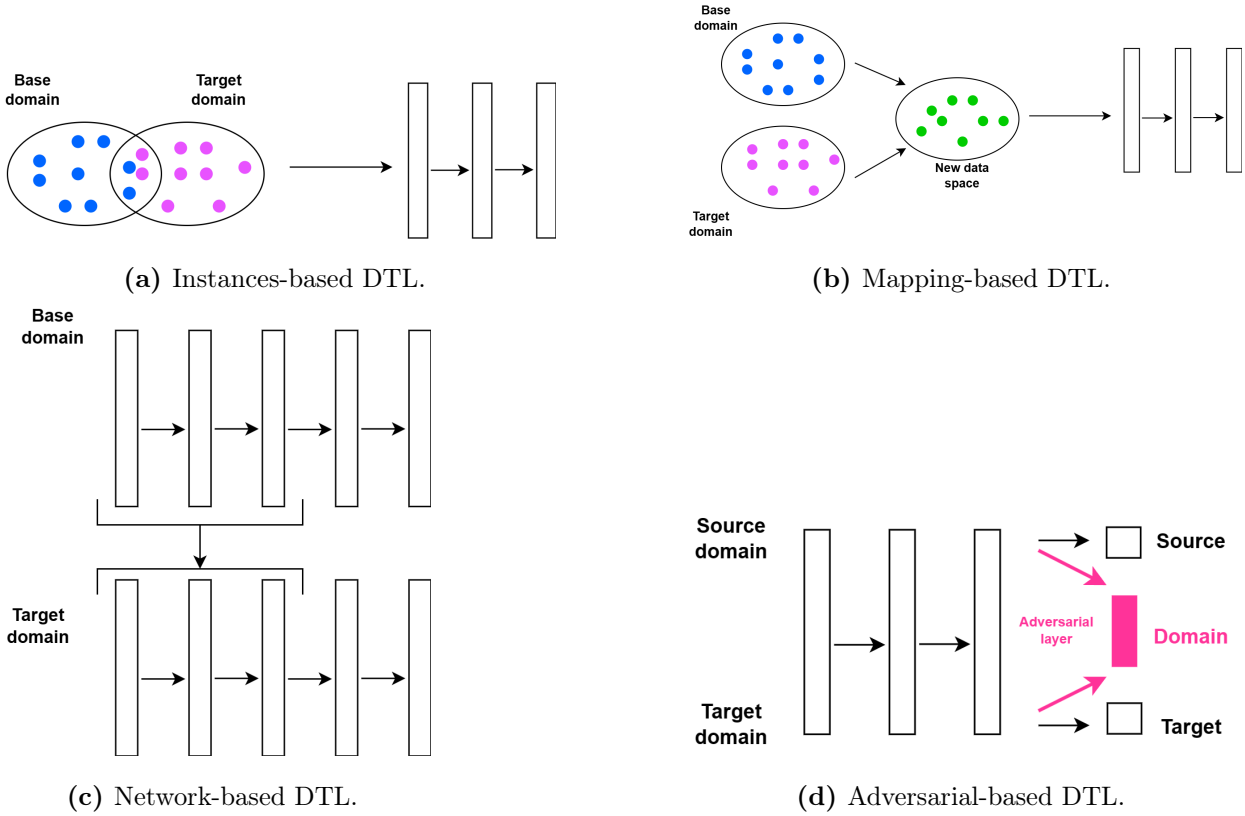
Figure 2.7: Adapted EfficientDets's architecture. Taken from [32].

2.3 Transfer-learning

Usually, DNNs require a large quantity of data, and there may be specific cases when that quantity of information is not available, the dataset is very small, or there simply is not enough data for training. Additionally, it is also time-consuming. Transfer learning is introduced as an approach to address the concept that the knowledge acquired by a model during training of a dataset can be applied to a similar target dataset [18]. In a *lato sensu* perspective, it is quite similar to the logic of how a human thinks: once a person learns about what a car is, they are able to recognize different models, trucks, buses, and pedestrians. In transfer learning, the model learns the features that represent a certain object or class and is later able to use them in similar situations that has never seen before. This process has enabled training time to be saved since the model does not have to start from scratch.

This technique is studied [76] to work well for detecting generic features since moving deeper into the architecture means that the features also become highly specific [62] *i.e.*, that there's a certain degree of generality in the initial layers that can be utilized for various datasets.

According to [66] there are four types of DTL (deep transfer learning): instances-based DTL [73], mapping-based DTL [45], network-based DTL [70] and adversarial-based DTL [44]. These methods are illustrated in Figure 2.8. Instances-based DTL is related to the use of a specific weight adjustment strategy. This means that it selects a few instances from the base dataset as enhancements for the training set in the target one by attributing appropriate weight values to the selected instances. Mapping-based DTL corresponds to the mapping of instances from the base dataset in addition to the target domain into a new data space. It stands on the idea that these instances together, even though they originate from two different datasets, may exhibit greater similarity in the new data space. Adversarial-based DTL refers to the introduction of GAN to find transferable representations for both base and target datasets. Finally, Network-based DTL involves the reuse of part of the pre-trained network in the base dataset, including its structure and parameters, and incorporating it into the DNN to be used in the target dataset.



(a) Instances-based DTL.

(b) Mapping-based DTL.

(c) Network-based DTL.

(d) Adversarial-based DTL.

Figure 2.8: Illustrations of the various DTL methods. Based on [66].

This last one, also known as model-based DTL, is one of the methods studied and applied in this thesis and embraces strategies like fine-tuning, frozen CNN layers, and progressive learning. Network-based DTL, particularly fine-tuning that consists of training in a similar dataset to the

target dataset and fine-tuning on the target dataset, is one of the most commonly used methods due to how easy it is to apply, reduces training costs and does not require the researchers to have a large target dataset [25]. The most popular models to transfer learning with are VGG-Net, Alex-Net, and ResNet since they have already been trained in the dataset ImageNet [63]. In fine-tuning, it is possible to find works like [72] where it is pre-trained a model with images of detected pulses from multiple areas detected by UAV swarms and use YOLO-MobileNet, Fast RCNN, and Cascade RCNN to fine-tune it, achieving improvements in the results.

The freezing of CNN layers is also a very popular method and consists of the process of fixing layers in a pre-trained model and subsequently fine-tuning solely lateral fully connected layers. The CNN layers extract the feature, while the fully connected layers have the task of classification. Then the model will proceed to be fine-tuned on the target data. It is used in [75], where the model is first pre-trained with ImageNet, and then they proceed to transfer the learned information to the target model, freezing all the layers except for the last three that will be trained. This method is also studied in Chapter 4.

2.4 Related work

This section explores related works in the areas of object detection in thermal images, focusing on transfer learning and fine-tuning. For the detection of objects in thermal images, multiple works explored how knowledge learned from previous datasets, particularly RGB datasets, could be used.

In [35] the researchers study the transfer learning of all layers and the impact of retraining a variable number of layers on datasets of different sizes. They used the Caffe framework, 1000 ImageNet, Keras and CIFAR-10. They concluded that it is preferable to freeze only the first few layers for large datasets; on the other hand, models had better performance when more layers were transferred and frozen in small target datasets. In this sense, authors in [48] use YOLOv5 to monitor activities caught in UAV (Unmanned Aerial Vehicle) images through drones. The problem considered is related to the detection of the small size of the objects in a dataset that includes RGB images with the respective pairs of TIR (thermal infrared). The main goal is to infer which model (if RGB, TIR, or a mix of both, with or without transfer learning) is the best to solve the question considered. Analyzing the results of the seven scenarios applied to the TIR test set, it was observed that both YOLO-TIR and YOLO-RGBT (the complete dataset with RGB plus TIR images) with transfer learning performed the best.

Since thermal images are better suited for poor light conditions, the authors in [26], consider the detection of people in thermal videos and images recorded at night in different weather conditions (rain, fog, and clear) in a forest scene. They use the original version of YOLO trained with COCO; however, the authors note that due to the disparities between thermal and RGB images, YOLO

was not able to achieve a reasonable mAP@0.5 with a result of only 23%. On RGB images, this value achieves 90%. On the other hand, when YOLO is pre-trained with an RGB dataset with the addition of thermal images, the AP value achieves 97% even at various distances.

The same authors of the previously cited paper studied in [34] various detection algorithms used for RGB images in a thermal dataset created to simulate different illegal movements in different weather conditions (fog and rain). They proceeded to compare the performance of Faster RCNN, SSD, Cascade RCNN, and YOLOv3. The results showed that YOLOv3 pre-trained with the COCO dataset could still recognize and perform well in the thermal dataset of around 3000 images due to the resemblance of RGB images compared to thermal images. YOLOv3 has also performed well on tests with external image sets, and the best results were related to the combination of all sets used in the mentioned paper. It also showed good generalization.

Also, in [20] the authors focus on a work similar to this thesis: real-time detection of humans in infrared images. They fine-tuned YOLOv3 with a model pre-trained with COCO, achieving a mAP@0.5 of 89.28% on the validation set. They conclude that fine-tuning increases the accuracy in the same validation set relative to YOLOv3 without fine-tuning and that it is important to detect small objects.

In [61] the researchers, to tackle the lack of indoor thermal datasets, created one composed of only indoor thermal images called THS-DATA and adapted YOLOv5. The images were submitted to a variety of pre-training methods to fine-tune them with the THS-DATA, to diminish the reliance on the amount of training data available they preformed the following setps: i) Pretraining on each RGB channel; 2) Pretraining RGB images converted to Grayscale; 3) Pretraining on RGB images converted to Grayscale, but increased the contrast between people and the background. The results showed that training with each channel separately (i) and converting to Grayscale (ii) had the best performance in comparison to simulated images where the contrast was increased (iii).

In summary, these related works demonstrate the importance of transfer learning and fine-tuning object detection models for thermal imaging, not only to address the problem of a lack of data but also to improve performance results. Additionally, various studies use RGB datasets, mainly COCO, to pre-train the models that will be used to train thermal datasets.

3 Methodology

In this Chapter, the goal is to explain the research design, the datasets used for experiments, and the conditions and parameters associated with them, as well as explore in detail YOLOv5's architecture and the metrics that are going to be used to evaluate the results.

3.1 Datasets

The dataset used to support the experimental part and the respective results of this thesis was created at the Department of Electrical and Computer Engineering of the University of Coimbra by a team of researchers from the Institute of Systems and Robotics (ISR) with a mobile robot that also operates in indoor environments (Figure 3.2). It is composed of 757 RGB and thermal pair images for training and 525 pair images for testing, making a total of 2564 images. The pairs are calibrated; however, they present a slight temporal mismatch due to hardware limitations. The separation between testing and training is essential to make sure the model does not overfit during training.

The thermal images that compose the target dataset were taken using an LWIR camera, Flir Boson 640-512 pixels, professional grade with shutter, Lens 50^{o} 8.7mm, 60fps. The RGB images were taken with an Ximea MQ013CG-E2. All of the images have a shape of 640×512x3 pixels and have no augmentations applied to them. The cameras can be seen in Figure 3.3.

(a) Test set image from the RGB dataset.

(b) Test set image from the thermal dataset.

Figure 3.1: Datasets created at DEEC and studied.

The environment includes both images with high and low luminosity (where thermal images are more useful), which creates a contrast between them. It can also be observed that there is a considerable amount of movement in some of the corridors, as represented in Figure 3.1.

Figure 3.2: Mobile robot created and used to capture and collect the images inside the department DEEC-UC.

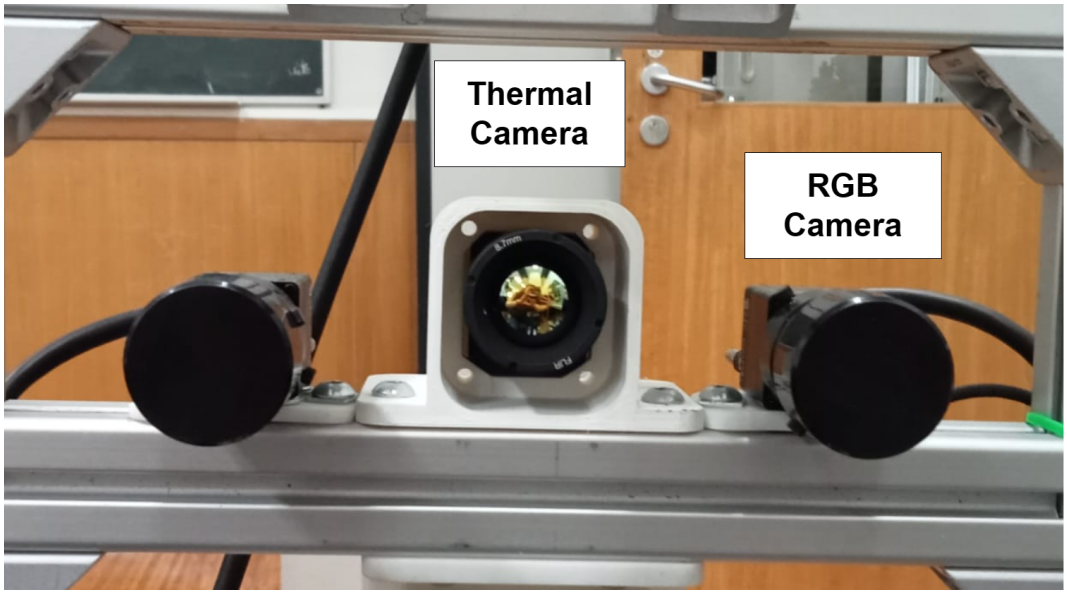


Figure 3.3: Close look at the cameras attached to the robot that captured the images.

After the collection, it was necessary to annotate and label the images from both the RGB and the thermal datasets. The annotation process was carried out using a computer vision annotation tool called CVAT which can be seen in Figure 3.4.

(a) RGB. **(b)** Thermal.

Figure 3.4: Example of two images from the RGB and thermal datasets during the annotation process.

For transfer-learning purposes, the model studied, and which will be described in section 3.3, was pre-trained with the RGB version created by the ISR's members, a Grayscale format (Figure 3.6a) converted from the RGB dataset, and also with two external datasets: *Teledyne FLIR ADAS Thermal Dataset* (that will be addressed as FLIR in this thesis for facilitation purposes) [60] and *MS COCO* [40], both very well known and very common. The MS COCO 2017 dataset (Figure 3.6b) has 80 classes and more than 200k images in different and complex situations, with varying sizes. The pre-trained weights derived from the MS COCO dataset were provided by Ultralytics

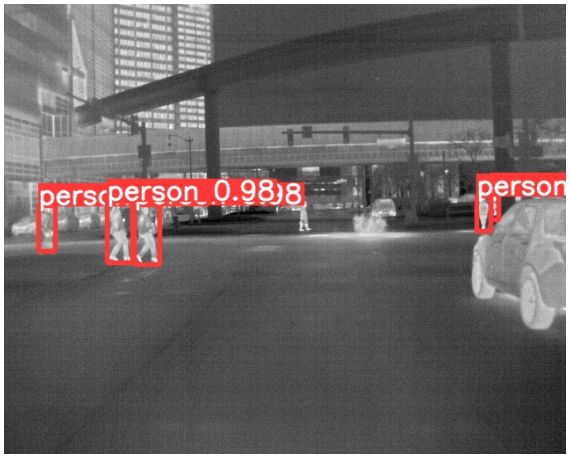
[28].

On the other hand, it was created two versions of pre-train with the FLIR dataset: a multi-class FLIR that will be referred to as FLIR, and a one-class FLIR, that will be referred to as FLIR*. This means that FLIR was trained with all the original classes (Table 3.1) considered by the authors, while FLIR* was trained considering only the "person" class. Examples of these two versions can be observed in Figure 3.5.

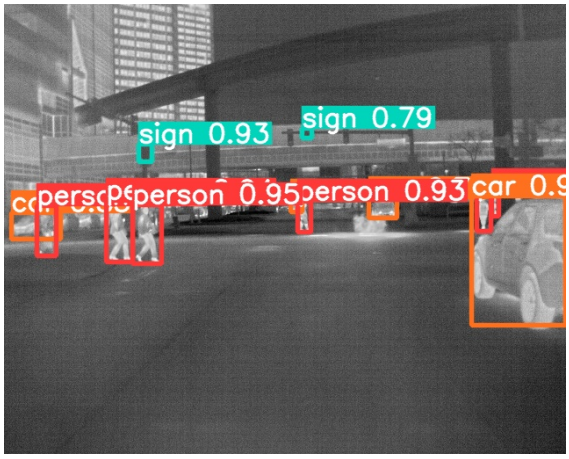
FLIR comprises a total of 26,442 video frames, but it also includes RGB images. Since only the thermal images are required, a selection of 13,000 images was used. FLIR considers 15 label categories (the labels can be seen in Table 3.1). The thermal images were taken with a Teledyne FLIR Tau 2 640x512, 13mm f/1.0 (HFOV 45°, VFOV 37°) camera. There is an RGB version of this dataset, but we chose to take only the thermal images so we could compare the performance.

Table 3.1: Classes considered in FLIR.

FLIR's original classes			
Person	Traffic light	Bus	Scooter
Bike	Fire Hydrant	Train	Stroller
Car	Street Sign	Truck	Dog
Motorcycle	Skateboard	Other Vehicle	-



(a) Image from the test set from FLIR*.



(b) Image from the test set from FLIR.

Figure 3.5: Examples of images after undergoing detection using the model during both the training and testing phases with FLIR and FLIR*.

(a) Grayscale image.

(b) MS COCO image.

Figure 3.6: Example images from two out of the six scenarios tested.

3.2 Research Design and experimental setup

The studies considered in this thesis were conducted using the device's specifications detailed in Table 3.2, as well as Jupyter Notebook, using the PyTorch framework.

Table 3.2: Details about the device used to run the model.

Processor	AMD Ryzen 9 5900X 12-Core Processor
GPU	NVIDIA GeForce RTX 3090, 24576MiB
Installed RAM	32.0 GB
System type	64-bit operating system, x64-based processor

To be able to assess how YOLOv5 performs in the detection of people in thermal images of indoor environments, it was evaluated different scenarios: one Control scenario corresponding to the training from scratch with the in-house thermal dataset (the target dataset) and five pre-trained scenarios where the models are then used to fine-tune the model of our target dataset. The datasets used for pre-training are described in the previous section and correspond to RGB, COCO, FLIR, FLIR*, and Grayscale.

Preparations steps:

1. Organization of the thermal dataset in training, validation, and testing. Train and test were already defined; validation corresponds to about 10% of the training set.

2. Download and organization of the FLIR dataset and corresponding labels, since it was necessary to convert the *.json* format to YOLO Darknet label format to be compatible with the YOLOv5 algorithm.

3. Train from scratch FLIR, FLIR*, RGB, and Grayscale datasets to take the weights to use as pre-training with training parameters defined in Table 3.3, with the difference that they were only trained with alearning rate of 0.01.

After these steps of preparation were finished, the model was fine-tuned with the pre-training weights of the datasets mentioned above, considering the training parameters in Table 3.3. The research steps are described further.

Table 3.3: Training parameters considered through all the training scenarios.

Batch	32	**Weight decays**	0.0005
Epoch	200	**Box Loss Gain**	0.05
Initial learning rate	0.1; 0.01; 0.001	**Class BCE positive weight**	1.0
Final learning rate	0.1	**Object BCE positive weight**	1.0
Class Loss Gain	0.3	**Object Loss Gain**	0.7
IoU training threshold	0.20	**-**	-

Implementation and model-learning steps:

1. Training the target dataset by fine-tuning the model in the same pre-training conditions described in Table 3.3, including the three learning rates described (0.1, 0.01, and 0.1) without freeze;

2. Repetition of the previous step, with the conditions considered in Table 3.3 but additionally the backbone of the model is frozen (the first ten layers);

3. Assess the two best results between steps 1 and 2 and test them, as well as, on the RGB scenario, the constant decrease of frozen layers with a step of 2. The conditions in Table 3.3 remain, except that the learning rate considered is only 0.01 since it was the one associated with the best results;

4. Repetition of step 1 with data augmentation. The values considered for the augmentation parameters are described in Tables 3.4 and 3.5. The conditions considered are the same as described in Table 3.3;

5. Test of optimizers SGD and ADAM with the same conditions considered in Table 3.3, considering only the learning rates 0.01 and 0.001, in scenarios Control, RGB, COCO and FLIR*.

It was chosen to introduce high probability values on the augmentation parameters to evaluate how the algorithm performs under considerable changes in the dataset conditions. However, it is important to note that, during augmentation, the dataset does not increase in size (number of examples/instances).

Table 3.4: Hyperparameters used for augmentation.

Hyperparameter	**Values**	**Hyperparameter**	**Values**
Hue	0.015	Fliplr	0.5
Saturation	0.7	Mosaic Augmentation	1.0
Brightness	0.4	Mixup	0.1
Translation	0.1	Scale	0.9

Table 3.5: Augmentation parameters table.

Hue	Parameter that controls the value of hue.	**Scale**	This parameter controls the scaling of the image. Values below 1 decrease its size and above 1 increase it.
Saturation	Parameter that controls the saturation of the image.	**Mixup**	This parameter controls the probability of applying mixup augmentation using two images to create a new image.
Brightness	Parameter that controls the brightness or value of the image. It affects how light or dark the image appears.	**Fliplr**	This parameter controls the probability of flipping the image horizontally.
Mosaic Augmentation	Corresponds to the probability of applying mosaic augmentation.	**Translation**	This parameter controls image translation.

3.3 Model Architecture

YOLOv5 arrived a month after YOLOv4, hence the two versions have many similarities. However, YOLOv5 came to prove that it has higher accuracy and efficiency [24].

After the first version of YOLO, several versions appeared: YOLOv4, YOLOv5, YOLOv6, YOLOv7, and so on. YOLOv5 is one of the most used due to its accuracy and user-friendliness; it is also faster and requires less computational power. It is an extension of the famous YOLOv3 and is designed to be used with PyTorch.

YOLOv5 has five versions: nano, small, medium, large, and extra-large, whose use depends on the type of dataset, device, and task to which is going to be applied. What distinguishes them is

the width and depth of the layers, as well as their use. For instance, although YOLOv5x has the best performance, it comes with a cost in speed [67]. This thesis it will be used the medium version, YOLOv5m. The parameters corresponding to YOLOv5m can be seen in Table 3.6.

Table 3.6: YOLOv5m's version 7.0 parameters.

	Number of layers	**Number of parameters**	**GFLOPS**
YOLOv5m	212	20856975	47.9

3.3.1 Overview of YOLOv5

YOLOv5 has also introduced new features such as adaptive anchor boxes and mosaic augmentation. In previous versions of YOLO, there was a pre-defined set of anchor boxes that were usually empirically defined from larger datasets, but adaptive anchor boxes enable these anchors to be defined according to the target dataset, allowing the model to have a more accurate starting point [13]. Mosaic augmentation consists of randomly combining four images of the dataset to create one single image. This way it increases the variety and challenges the model, as seen in Figure 3.7.

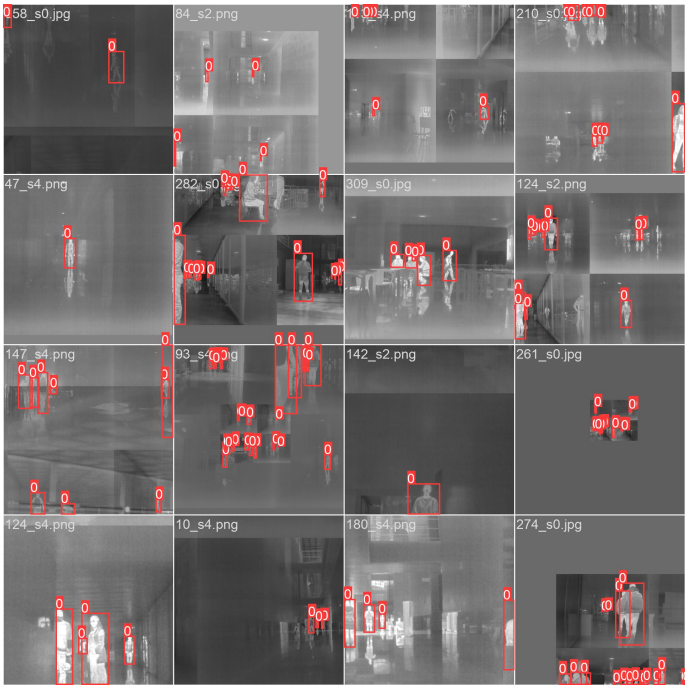


Figure 3.7: Example of the application of mosaic augmentation during the training process with FLIR* dataset at a learning rate of 0.01 without freeze.

YOLO components are all integrated into one single neural network, where the input is a certain number of images given by matrix (3.1).

$$\left[n, w, h, x\right] \tag{3.1}$$

That corresponds to the number of images, width, height, and depth of the image (*e.g.*, RGB is composed of three channels, *i.e.*, the depth of the image is three), respectively.

Each image is divided into an SxS grid. If an object is detected in one of the grid cells, that cell is the one to, first, detect such an object and, second, to predict it. The output of each grid cell is a vector described in matrix 3.2.

$$\begin{bmatrix} P_c \\ B_x \\ B_y \\ B_w \\ B_h \\ C_1 \\ C_2 \end{bmatrix} \tag{3.2}$$

Where,

- P_c − either 0 if no object is detected or 1 if otherwise;
- B_x, B_y − center position x and y of the grid cell where the object was detected;
- B_w, B_h − the width and height of the bounding box surrounding the object;
- C_1, C_2 − classification classes. If one object or more are detected in the grid cell these values are between 0 and 1, whether they are classified as C_1 and/or C_2. Consists of a conditional probability where P($Class_i$|Object).

The prediction of the bounding box position, as well as the center, are given by equations (3.3), (3.4), (3.5), (3.6).

$$B_x = \sigma(t_x) + c_x \tag{3.3}$$

$$B_y = \sigma(t_y) + c_y \tag{3.4}$$

$$B_w = p_w e^{t_w} \tag{3.5}$$

$$B_h = p_h e^{t_h} \tag{3.6}$$

Where c_x and c_y correspond to the coordinates of the grid cell on the image; t_x and t_y represent the predicted offset for the center x-coordinate and y-coordinate; and p_w and p_h represent the factors used to adjust the predicted width and height of the anchor box.

Each vector value consists of a confidence value, reflecting the probability of detection and respective localization of an object. Each grid cell predicts confidence scores regarding their respective

bounding boxes. These confidences are given by equation (3.7). The confidence scores for predicting the classes in the bounding boxes are given by equation (3.8). This equation considers the probability of a certain class existing and the suitability between the bounding box and the object it surrounds. The intersection over union is described in equation (3.9) and it is detailed in section 3.4 therefore:

$$Pr(Object) * IoU \tag{3.7}$$

$$Pr(Object) * IoU^{truth}_{pred} * Pr(Class_i|Object) = Pr(Class_i) * IoU^{truth}_{pred} \tag{3.8}$$

$$Intersection = \frac{Intersected\ Area}{Union\ Area} \tag{3.9}$$

On every cell of the grid mentioned above where the center of the object is detected, in each scale computed by the network, three bounding boxes will be generated associated with a middle point responsible for classifying the object. These bounding boxes are adjusted from the anchors and will be later used to choose the actual bounding box.

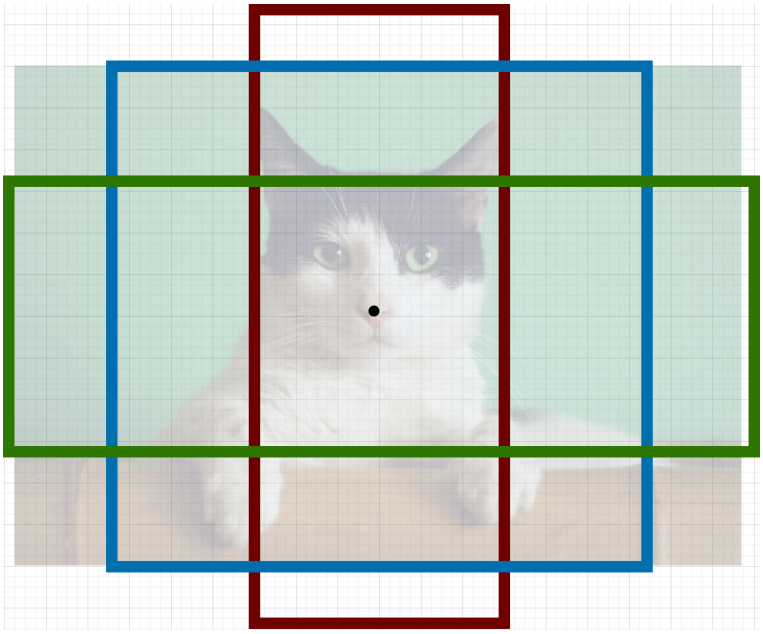


Figure 3.8: Illustrative example of three anchors being generated by a grid cell where the center of the cat was detected. Photo credits: Manja Vitolic.

The bounding box that will detect the object, will be selected through the use of Non-Maximum Surpression (NMS), a post-processing technique that will sort the bounding boxes by their confidence values and use Intersection over Union (IoU) - explained in detail in section 3.4 -, to compare the amount of overlaying between them. If it is above a certain defined threshold, NMS will discard the bounding box with less confidence.

3.3.2 Detailed architecture

Exploring in detail the YOLOv5 network, whose architecture can be seen in detail in Figures 3.9 and 3.10, it consists of three blocks:

- **Backbone:** New CSP-Darknet53 structure, that derives from the Darknet architecture of other YOLO versions.
- **Neck:** the structures used are SPPF (Spatial Pyramid Pooling Fast) and New CSP-PAN.
- **Head:** YOLOv5 used YOLOv3 Head.

Usually, the backbone of an object detector is a CNN that extracts features and creates feature maps of images in different qualities and scales (low-level features in earlier stages and high-level features in deeper layers). It is the main body of the network and is extremely important for accuracy. In YOLOv5, the backbone corresponds to New CSP-Darknet53 (derived from Cross Stage Partial), the same backbone as YOLOv4, with a slight variation in the first layers, which start with a Stem - first layers that process the raw input so it can be more suitable for feature extraction, helping to save memory and processing time [67].

The New CSP-Darknet53 will receive an input that will be divided into two different paths: one that will go directly through a transition and another that will go through a dense block. The outputs of these paths will be concatenated and go through another transition layer. This backbone addresses the repeating gradient information in large models and integrates gradient change into feature maps to improve inference speed, accuracy, and model size by decreasing the parameters. This is represented by the C3 block in Figures 3.9 and 3.10.

From the backbone, the output will be feature maps that will go through the neck that connects the backbone and head. It is used to combine image features and improve semantic and spatial information across different scales. The neck comprises SPPF (Figure 3.10) and New CSP-PAN. The first one is an optimized version done by the authors of YOLOv5 that doubles the time. SPP [22], the base of SPPF, aims to permit an image of any size as input by generating a fixed-length representation through pooling to be fed into fully connected layers (that are the ones to require fixed-size inputs). Another advantage of SPP is the fact it only needs to run once to scan the image, which increases the speed of the process [67]. This algorithm allows for the extraction of important spatial features for context and helps detect objects at different scales. The modification introduced by the authors of [28] consists of calculating three max-poolings with different kernels and paddings. SPPF applies a single max-pooling operation to the input that will be implemented three times, and the output of the first max-pooling operation becomes the input to the second, and so on. The final output is produced by concatenating the original input with the outputs of all three max-pooling operations.

In the neck, it is also possible to find the New CSP-PAN which is an adaptation of PAN (Path Aggregation Network) [41] whose goal is to facilitate the gradient and information flow through a bottom-up pathway since low-level features are essential for large instance identification.

The head of YOLOv5 is the same as its precedent models, such as YOLOv3 and YOLOv4. This means that it will generate three output feature maps corresponding to three different ratios, allowing it to detect small and large objects. The expression used to calculate the feature map is given by equation (3.10),

$$c = (5 + n_{cls}) \times 3 \tag{3.10}$$

where c corresponds to the feature map generated, n_{cls} corresponds to the number of classes, and the number three is related to the number of bounding boxes predicted. The architecture of the algorithm can be seen in figures 3.9 and 3.10.

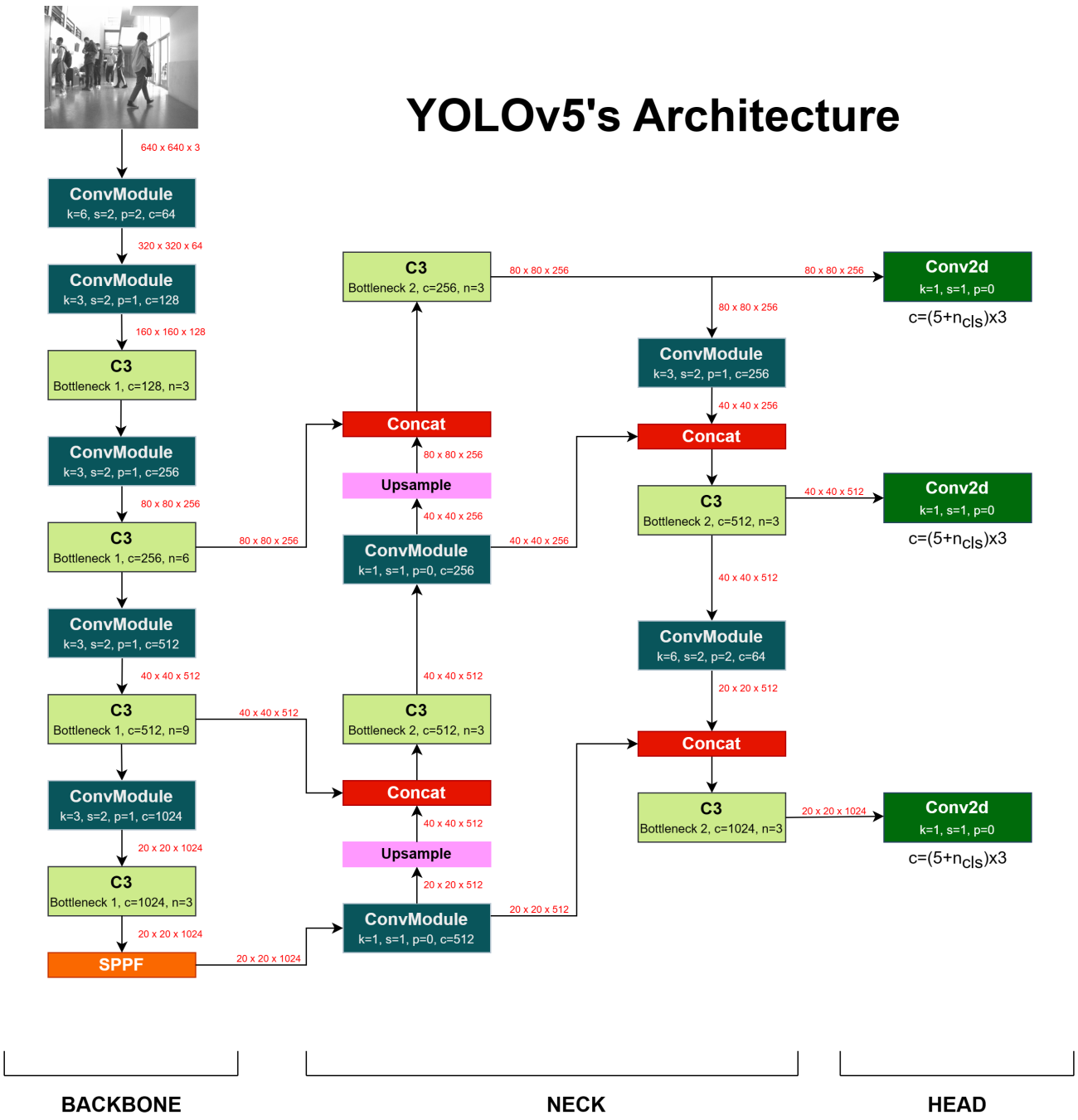


Figure 3.9: YOLOv5's architecture based on [28] and [30].

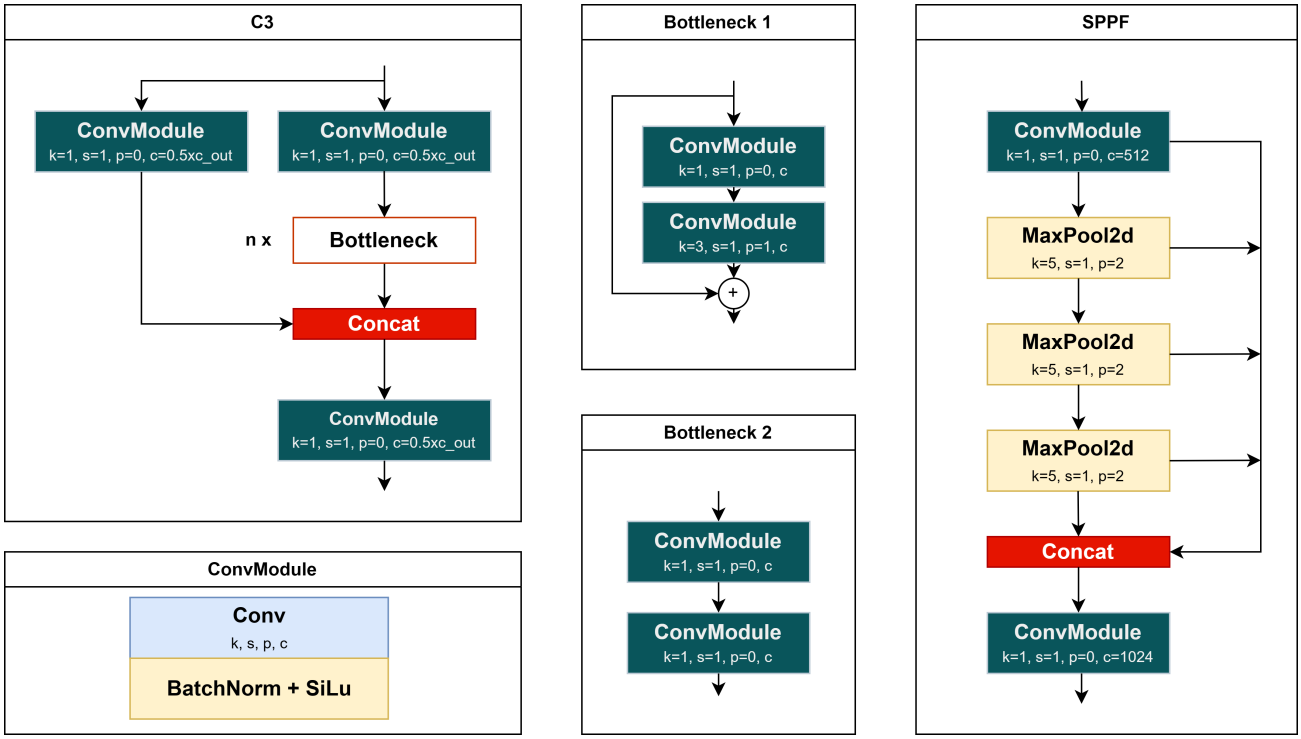


Figure 3.10: YOLOv5's architecture details based on [28] and [30].

3.3.3 Loss, learning rate, activation and optimization functions

The default that corresponds to the Sigmoid Linear activation function (SiLU), was chosen for the activation function, which is mathematically defined in equation (3.11),

$$silu(x) = x \times \sigma(x) \tag{3.11}$$

where $\sigma(x)$ is the logistic sigmoid. For the optimization function, in this thesis it was opted for SGD (Stochastic Gradient Descent) as it is commonly used for SOTA articles and chosen by the algorithm's authors as the default. SGD adjusts the model's parameters using a random subset of the training data during iterations, increasing the randomness.

It was also tested the ADAM (Adaptive Moment Estimation) optimizer as well, represented by equation (3.12) since it is also very used and usually chosen as the default optimizer because it converges faster.

$$w_{t+1} = w_t - \alpha m_t \tag{3.12}$$

Where,

$$m_t = \beta m_{t-1} + (1 - \beta)[\frac{\delta L}{\delta w_t}] \tag{3.13}$$

β corresponds to momentum that is set to $\beta = 0.937$ and α corresponds to the One-Cycle learning rate. In this thesis, when the learning rate is mentioned it is always the initial learning rate (lr_0). The maximum learning rate (lr_f) is set to 0.1, corresponding to the default value. It is used a

learning rate scheduling method called One Cycle where the initial learning rate is low and will increase step-by-step to the maximum learning rate. Then, the learning rate will decrease again, in the second part of training.

For the loss it was used Binary Cross-Entropy (BCE) with Logits Loss predefined in Pytorch and it is defined as found in equation (3.14). This loss function combines a sigmoid layer and the normal BCE Loss function and is considered to be more numerically stable [4].

$$l(x, y) = L = \{l_1, ..., l_N\}^T, l_n = -w_n[y_n \cdot log\sigma(x_n + (1 - y_n) \cdot log(1 - \sigma(x_n))] \tag{3.14}$$

where,

- c corresponds to the number of classes;
- n corresponds to the number of the sample in the batch;
- p_c corresponds to the weight of the positive answer for the class considered.

3.4 Evaluation Metrics

In this thesis the evaluation and comparison of the results were based on precision, recall, mAP@0.5:@0.95 and F1 score. Precision, equation (3.15), consists of the ratio of actual positives relative to all the samples classified as positive,

$$Precision = \frac{TP}{TP + FP} \tag{3.15}$$

where TP corresponds to True Positives and FP corresponds to False Positives. Recall (equation (3.16)) is a sensitivity metric that tells the ratio of the number of correct positives out of all of them,

$$Recall = \frac{TP}{TP + FN}. \tag{3.16}$$

The mean average precision (mAP), expressed as in equation (3.17), is the ratio between precision and recall and is one of the most famous parameters for object detection to evaluate performance.

$$mAP = \frac{1}{N} \sum_{q=1}^{q} Average\ Precision(q) \tag{3.17}$$

where N is the number of objects and q is a given object. This concept is related to IoU which is calculated as the ratio of the areas of intersection and union of the predicted and ground truth bounding boxes, as exemplified in Figure 3.11. IoU determines if a bounding box is a true positive, false positive, or false negative. There are different IoU thresholds to calculate mAP: 0.5:0.05:0.95,

0.5 and 0.95. The first one enables incremental changes in the threshold during evaluation and is used to assess the localization accuracy [79], the others only calculate the percentage described above. In this thesis it is used mAP@0.5 and mAP@0.5:0.95.

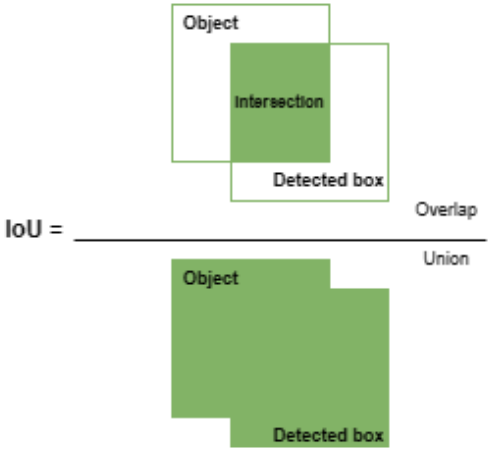


Figure 3.11: IoU visualization [67].

The threshold established by IoU will determine if something is detected:

- IoU $>$ 0.5: True Positive;
- IoU $<$ 0.5: False Positive (wrong detection).

If there is no intersection, no object is detected. The F1-score (equation (3.18)) corresponds to the harmonic mean between precision and recall and ranges between 0 and 1, where 1 represents high precision and recall values. F1 penalizes extreme values of these two metrics [23]. In YOLOv5, the average F1-score for the class is calculated for each threshold.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3.18}$$

It was taken the arithmetic average of the values while analyzing the results. It is defined as in equation (3.19),

$$\bar{x} = \frac{x_1 + x_2 + ... + x_n}{n} = \frac{1}{n} \sum_{1}^{n} x_i \tag{3.19}$$

where n corresponds to the number of elements considered.

4 Experimental results

This Chapter reports the experimental results and the evaluation metrics; they are precision, recall, mAP@0.5, mAP@0.5:0.95, and F1-score (described in section 3.4), calculated on the test set.

In section 4.1 it is presented the fine-tuning results for all learning rates considered with and without freeze, with no augmentation, for all the scenarios used for pre-training: Control, RGB, FLIR*, FLIR, and Grayscale (Gray). The conditions used to pre-train and subsequently fine-tune can be seen in Table 3.3 in Chapter 3. This section corresponds to research steps 1 and 2.

In section 4.2 it is analyzed three of the scenarios' performances regarding variation of the number of frozen layers, so it will be evaluated how the model responds in different frozen layer conditions.

In section 4.3, it is demonstrated the evolution with augmentation (the parameters considered can be seen in Table 3.5) in comparison with no augmentation, as well as their respective graphics to help visualization. Finally, this Chapter finishes with section 4.4 where it evaluates how the different optimizers - SGD and ADAM - impact some of the scenarios.

In Appendix .1 it is possible to see some qualitative results of the detections obtained.

4.1 Results on human detection using YOLOv5

Tables 4.1, 4.3 and 4.5 correspond to the results with learning rates of 0.01, 0.001 and 0.1, respectively. Whereas Tables 4.2, 4.4 and 4.6 correspond to the results of their respective learning rates, but with ten layers frozen, corresponding to the backbone. Every table has a Control, that corresponds to the baseline for each learning rate. Control has no frozen layers because it is not pre-trained (i.e., it is trained from scratch with the target dataset).

To be able to compare tables between each other, the average values for each metric were computed; however, to guarantee the outliers did not weigh on the decision, the average of the three best test results on each table were also taken (that corresponds to average-3).

Based on Table 4.1, that shows the results without freeze and with a learning rate of 0.01, it has been found that such configuration outperforms all the other ones in all metrics, including average and average-3. The two best scenarios in this table are COCO (Figure 4.1a) and FLIR* (Figure 4.1b).

Comparing the performance measures in tables 4.1 and 4.3 to 4.2 and 4.4, which give the results with and without freeze, it is possible to see that freezing the layers reduces the performance and decreases all the metric parameter values: mAP@0.5, mAP@0.5:0.95, F1-score, precision, and recall. On the other hand, comparing Tables 4.5 and 4.6, which correspond to the results without and with freeze with a learning rate of 0.1, it was obtained mixed results: COCO and FLIR improved the performance when layers were frozen (relative to the results of the same learning rate but without freeze), while the remaining scenarios follow the tendency of the decrease in performance (observed in the previous comparisons). For this reason, section 4.2 evaluates the impact of frozen layers on three scenarios: COCO, FLIR*, and RGB.

In four out of the six reported results, the model pre-trained with Grayscale data outperforms its RGB counterpart, even though the results exhibit a high degree of similarity. In comparison to the Control scenario, both RGB and Grayscale demonstrate similar performance levels. However, it is noteworthy that in Table 4.4 and Table 4.6, both RGB and Grayscale models fall short on performance achieved in contrast to the Control scenario. Among these tables, RGB performs the worst in Table 4.1 and Table 4.2, while the Control scenario performs the worst in Table 4.3. The sole exception to this trend occurs in Table 4.5, where the FLIR model generates the least favorable results, nonetheless it is very closely aligned with those of the Control scenario.

It is evident that utilizing a large RGB dataset leads to better results compared to using a thermal dataset. COCO's substantial size - which greatly the next-largest dataset used in this study - results in significant improvements in model generalization. A broader generalization means more diverse samples, which helps the model learn a broader range of features and patterns. However, even though FLIR* had good results, its lower performance can also be due to the fact that numerous images did not contain people (even though this is also true for COCO, but the dataset much larger, as stated before), and a portion of them were very similar to each other, having less diversity.

This last observation can give some hints as to why the RGB scenario frequently has one of the weakest performances, althouth typically still retains valuable feature information and is usually an option for pre-training. Linking that to the fact that the Control scenario also frequently underperforms, this suggests that the dataset might be insufficient in terms of size or information, compromising the model's ability to learn. It is possible to see that, even though the FLIR dataset has limitations and is an outdoor thermal dataset, - which takes into account very different context situations - it improved the results significantly compared to RGB and Control therefore, it is most likely large enough to contain sufficient information relative to the scenarios mentioned.

In this section, it is also possible to compare the results between FLIR and FLIR*. It is apparent that the results are similar; however, FLIR* performs better without freezing, on the other hand, FLIR has higher results when the backbone is frozen relative to FLIR*. This might be due to the fact that FLIR has more generalization since it contains information about multiple classes and, by

freezing layers, enables the transfer of low-level feature knowledge.

Looking at the values of average and average-3 across the results, even though there are differences, the learning rate does not have as much impact compared to the size of the dataset for the pre-training stage and the amount of generalization they introduce. However, it does have an impact on the discrepancy between the data and that will be discussed in section 4.3 where augmentation is analyzed.

Table 4.1: Results across the various situations for the learning rate 0.01. Freeze=0, no augmentation.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.824	0.609	0.721	0.372	0.70 (@0.635)
RGB	0.817	0.607	0.704	0.37	0.70 (@0.507)
COCO	**0.903**	**0.781**	**0.866**	**0.544**	**0.84 (@0.810)**
FLIR*	0.833	0.675	0.773	0.477	0.75 (@0.049)
FLIR	0.801	0.679	0.767	0.459	0.73 (@0.065)
Gray	0.815	0.625	0.723	0.384	0.71 (@0.446)
Average	0.832	0.663	0.759	0.434	0.74
Average-3	0.846	0.712	0.802	0.493	0.77

Table 4.2: Performance measures, on the test set, across the various situations for the learning rate 0.01. Freeze = 10, no augmentation.

	Precision	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.824	0.609	0.721	0.372	0.70 (@0.635)
RGB	0.777	0.547	0.636	0.299	0.64 (@0.488)
COCO	**0.888**	**0.731**	**0.828**	**0.491**	**0.80 (@0.711)**
FLIR*	0.751	0.654	0.726	0.406	0.70 (@0.112)
FLIR	0.797	0.658	0.748	0.410	0.72 (@0.173)
Gray	0.756	0.568	0.643	0.303	0.65 (@0.470)
Average	0.799	0.628	0.717	0.380	0.70
Average-3	0.812	0.681	0.767	0.436	0.74

Table 4.3: Reported results for the learning rate 0.001. Freeze=0, no augmentation.

	Precision	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.690	0.501	0.582	0.225	0.58 (@0.520)
RGB	0.697	0.611	0.646	0.311	0.65 (@0.407)
COCO	**0.886**	**0.803**	**0.853**	**0.49**	**0.84 (@0.757)**
FLIR*	0.795	0.677	0.741	0.436	0.73 (@0.104)
FLIR	0.784	0.676	0.715	0.357	0.73 (@0.342)
Gray	0.737	0.604	0.666	0.313	0.66 (@0.565)
Average	0.765	0.645	0.701	0.355	0.70
Average-3	0.822	0.719	0.770	0.428	0.77

Table 4.4: Results using learning rate 0.001, freeze = 10, and no augmentation.

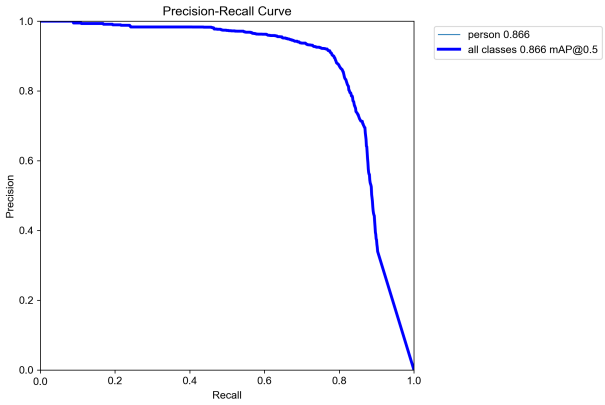
	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.671	0.511	0.563	0.222	0.58 (@0.524)
RGB	0.696	0.504	0.562	0.238	0.58 (@0.591)
COCO	**0.825**	**0.721**	**0.799**	**0.451**	**0.77 (@0.569)**
FLIR*	0.760	0.627	0.692	0.367	0.69 (@0.423)
FLIR	0.802	0.626	0.702	0.334	0.70 (@0.604)
Gray	0.683	0.512	0.559	0.236	0.59 (@0.538)
Average	0.740	0.584	0.646	0.308	0.65
Average-3	0.796	0.658	0.731	0.384	0.72

Table 4.5: Experimental results using learning rate 0.1, freeze=0, no augmentation.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.782	0.606	0.698	0.369	0.68 (@0.454)
RGB	0.802	0.623	0.716	0.388	0.70 (@0.453)
COCO	**0.773**	**0.668**	**0.733**	**0.407**	**0.72 (@0.357)**
FLIR*	0.765	0.663	0.716	0.394	0.71 (@0.239)
FLIR	0.718	0.634	0.692	0.373	0.67 (@0.123)
Gray	0.782	0.642	0.727	0.396	0.71 (@0.275)
Average	0.770	0.639	0.714	0.388	0.70
Average-3	0.773	0.658	0.725	0.399	0.71

Table 4.6: Results for learning rate = 0.1, freeze=10, and again without augmentation.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.782	0.606	0.698	0.369	0.68 (@0.454)
RGB	0.792	0.524	0.628	0.305	0.63 (@0.488)
COCO	**0.826**	**0.699**	**0.785**	**0.436**	**0.76 (@0.522)**
FLIR*	0.774	0.634	0.705	0.38	0.70 (@0.208)
FLIR	0.816	0.631	0.729	0.393	0.71 (@0.306)
Gray	0.757	0.568	0.643	0.301	0.65 (@0.465)
Average	0.791	0.610	0.698	0.364	0.69
Average-3	0.805	0.655	0.740	0.403	0.72



(a) COCO. **(b)** FLIR*.

Figure 4.1: The precision-recall graphics of the two best scenarios, with a learning rate of 0.01.

4.2 A deeper look into layer freezing

In this section, it is taken a deeper look into the effect of freezing on the fine-tuning of our model. Specifically, the pre-training scenarios COCO, FLIR*, and RGB, were chosen to fine-tune our target dataset and evaluate a varying number of frozen layers (ranging from 0 to 12 with a step of 2). The results are presented for a fixed learning rate of 0.01, both in the form of tables (4.7, 4.9 and 4.8) and plots (4.2 and 4.7, for COCO; 4.4 and 4.5, for RGB; and 4.6 and 4.7, for FLIR*), for a more intuitive interpretation. The training parameters can be found in Tables 3.4, in Chapter 3. This section corresponds to the third research step.

It can be observed that the COCO and FLIR* scenarios exhibit similar behavior, in terms of the effect of frozen layers. The values of mAP@0.5 are minimal for more frozen layers, increasing as the number of frozen layers decreases, as shown by Figures 4.2 and 4.6, with Figures 4.3 and 4.7 following the tendency. The initial layers of a backbone network are responsible for detecting

fundamental features like edges and corners. This indicates that the model primarily learns basic, low-level features. As the network progresses deeper, it becomes apparent that the particular features it identifies diverge significantly from those found in COCO and FLIR*. In the case of COCO, the model achieves its highest performance, as shown in graphic 4.2, when two layers are frozen. However, with FLIR*, it is preferable to not freeze any layers. This follows the idea stated in the previous section that suggests that COCO provides more valuable information for learning low-level features compared to FLIR*, likely because of its greater generalization capability and also it might be because the FLIR dataset has images that do not contain any people and repeated scenarios.

In contrast, RGB behaves quite differently, as observed in graphics 4.4 and 4.5, as it shows a peak in performance when the entire backbone is frozen (corresponding to layers 0-9 in the model, *i.e.*, there are ten frozen layers). This suggests that up to the backbone, the model is acquiring crucial information for training, and comparatively to COCO and FLIR*, these features are more specific; therefore, there's still some resemblance between the RGB dataset and the target dataset. That said, fine-tuning with the backbone frozen with RGB achieves better results.

Table 4.7: Results for various numbers of frozen layers, with COCO pre-training. Learning rate of 0.01.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
F12	0.888	0.731	0.828	0.491	0.800 (@0.711)
F10	0.888	0.731	0.828	0.491	0.800 (@0.711)
F08	0.89	0.749	0.84	0.506	0.820 (@0.644)
F06	0.898	0.768	0.855	0.523	0.830 (@0.720)
F04	0.891	0.782	0.86	0.536	0.830 (@0.705)
F02	0.901	**0.795**	**0.872**	0.539	**0.84 (@0.770)**
F00	**0.903**	0.781	0.866	**0.544**	**0.84 (@0.810)**
Average	0.894	0.762	0.850	0.519	0.82

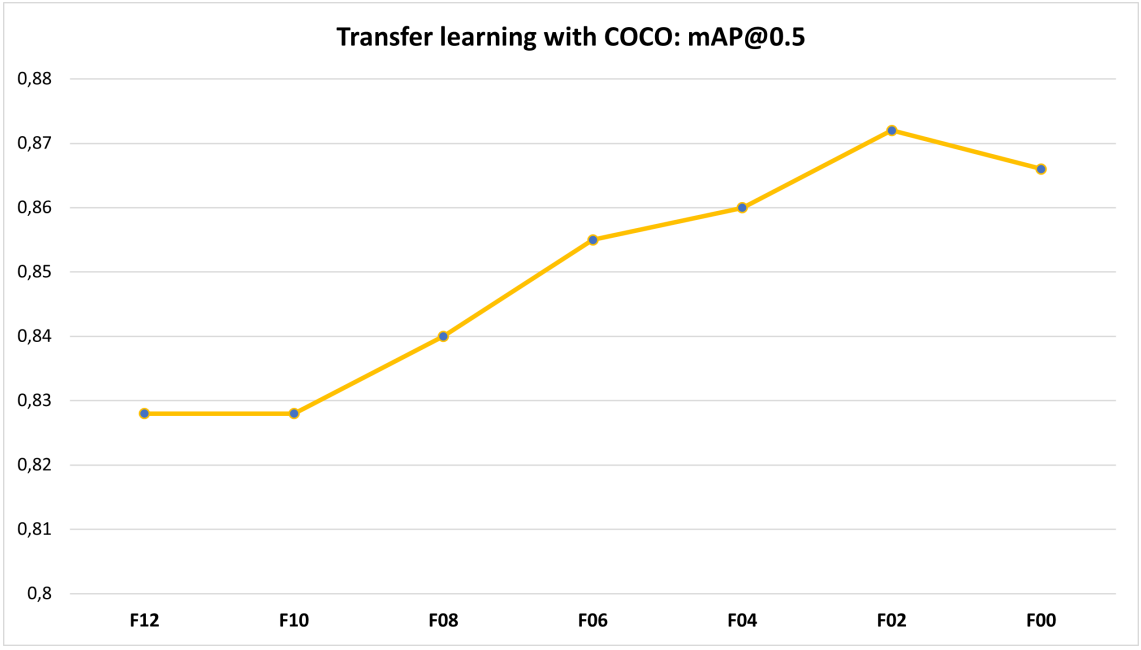


Figure 4.2: Evolution of the performance of the model with the variable freeze at a learning rate of 0.01, pre-trained with COCO.

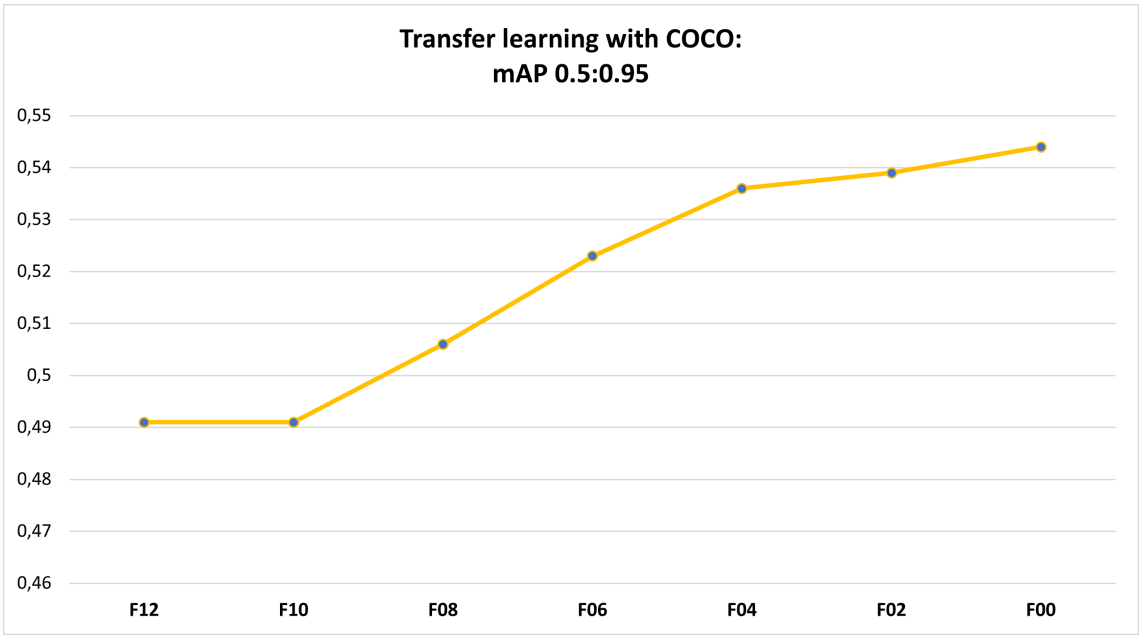


Figure 4.3: Evolution of the performance of the model with the variable freeze at a learning rate of 0.01, pre-trained with COCO.

Table 4.8: Freeze evaluation for RGB. Learning rate of 0.01.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
F12	0.768	0.554	0.635	0.299	0.64 (@0.457)
F10	**0.888**	**0.731**	**0.828**	**0.491**	**0.80 (@0.711)**
F08	0.722	0.566	0.639	0.304	0.64 (@0.388)
F06	0.763	0.581	0.658	0.317	0.66 (@0.312)
F04	0.775	0.598	0.668	0.359	0.68 (@0.390)
F02	0.775	0.626	0.695	0.362	0.68 (@0.390)
F00	0.817	0.607	0.704	0.370	0.70 (@0.507)
Average	0.787	0.609	0.690	0.357	0.67

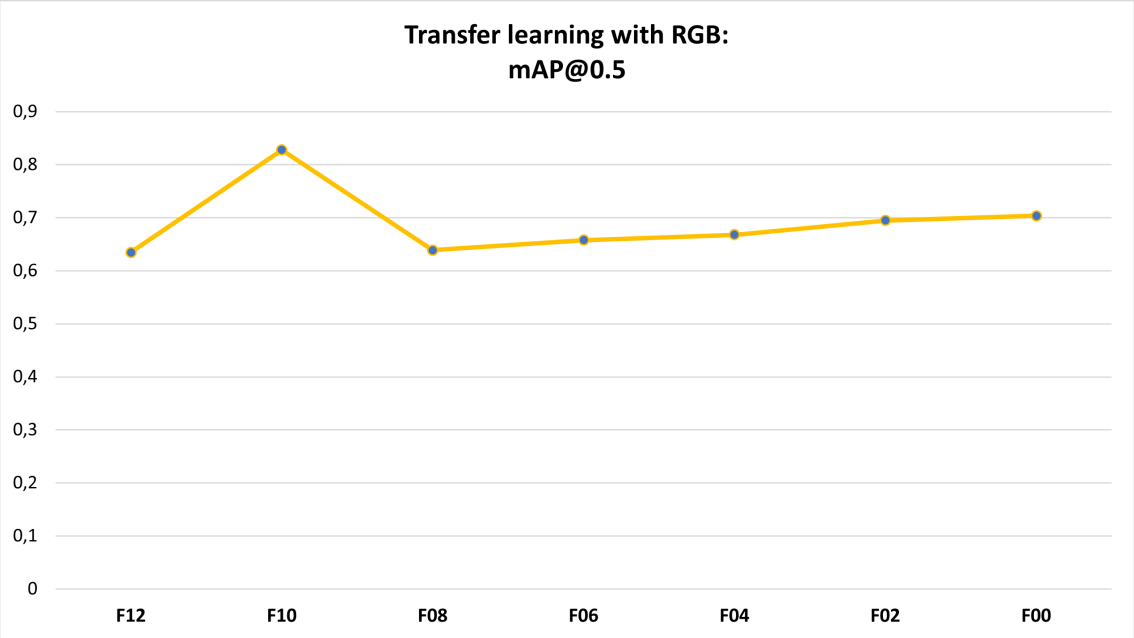


Figure 4.4: Impact of the varying numbers of frozen layers on mAP@0.5, with at a learning rate of 0.01, pre-trained with RGB.

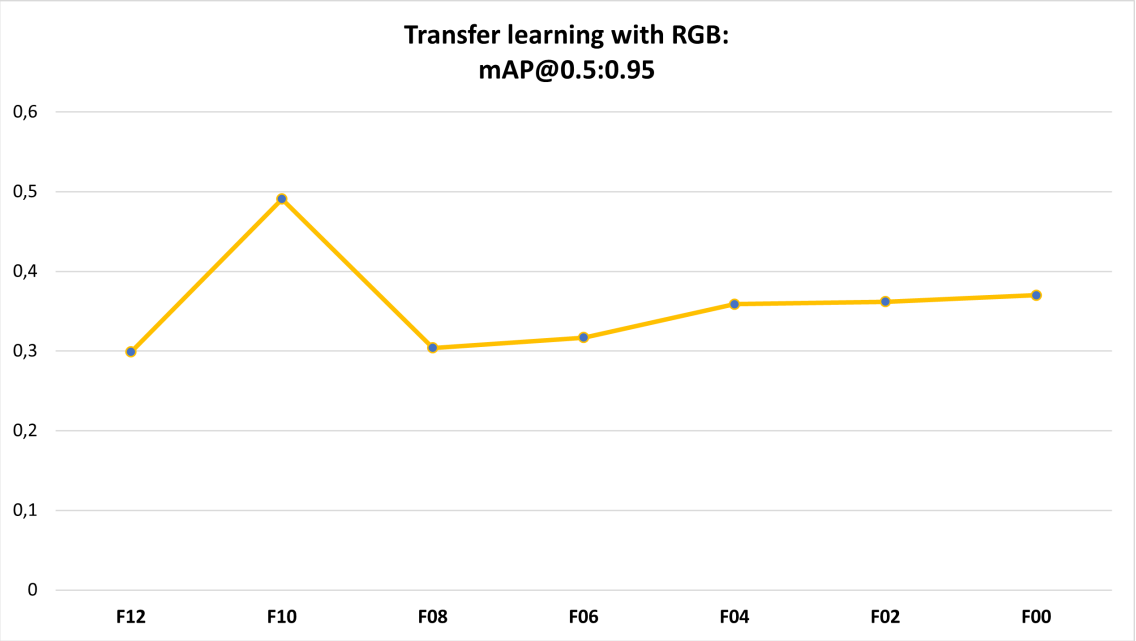


Figure 4.5: Impact of the varying numbers of frozen layers on mAP@0.5:0.95, with at a learning rate of 0.01, pre-trained with RGB..

Table 4.9: Impact of frozen layers with FLIR* pre-training. Learning rate of 0.01.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
F12	0.755	0.649	0.729	0.411	0.70 (@0.180)
F10	0.751	0.654	0.726	0.406	0.70 (@0.112)
F08	0.792	0.646	0.745	0.423	0.71 (@0.160)
F06	0.790	0.659	0.734	0.431	0.72 (@0.080)
F04	0.807	0.677	0.756	0.458	0.74 (@0.054)
F02	0.818	0.691	0.768	0.468	0.75 (@0.071)
F00	**0.833**	**0.675**	**0.773**	**0.477**	**0.75 (@0.049)**
Average	0.786	0.663	0.743	0.433	0.72

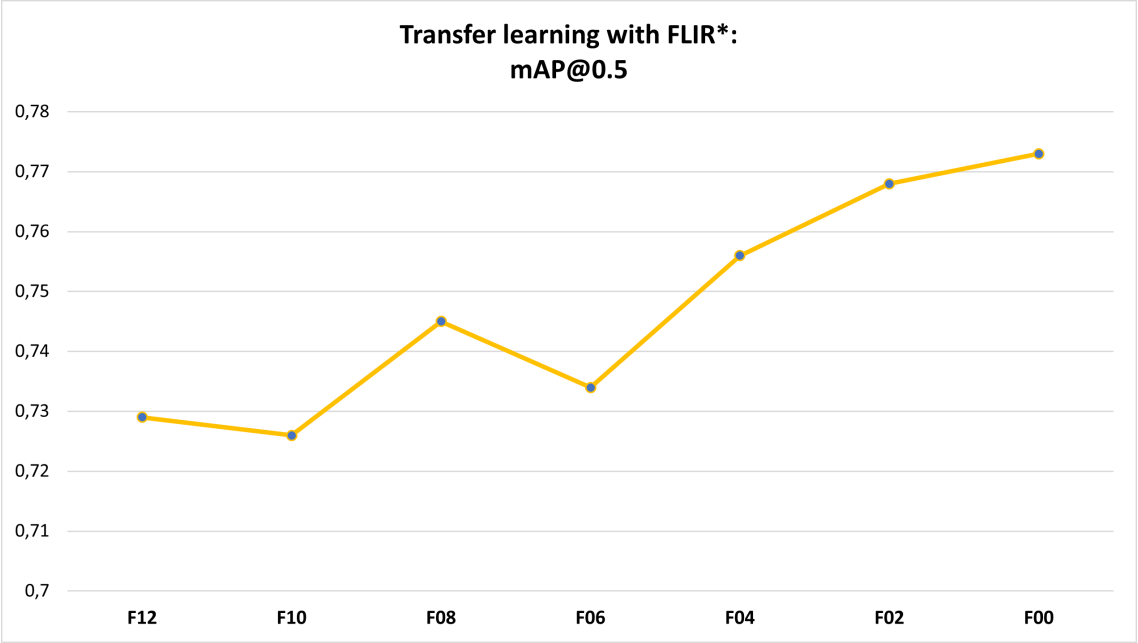


Figure 4.6: Study of the evolution of the performance when the model is fine-tuned with FLIR*. Learning rate of 0.01.

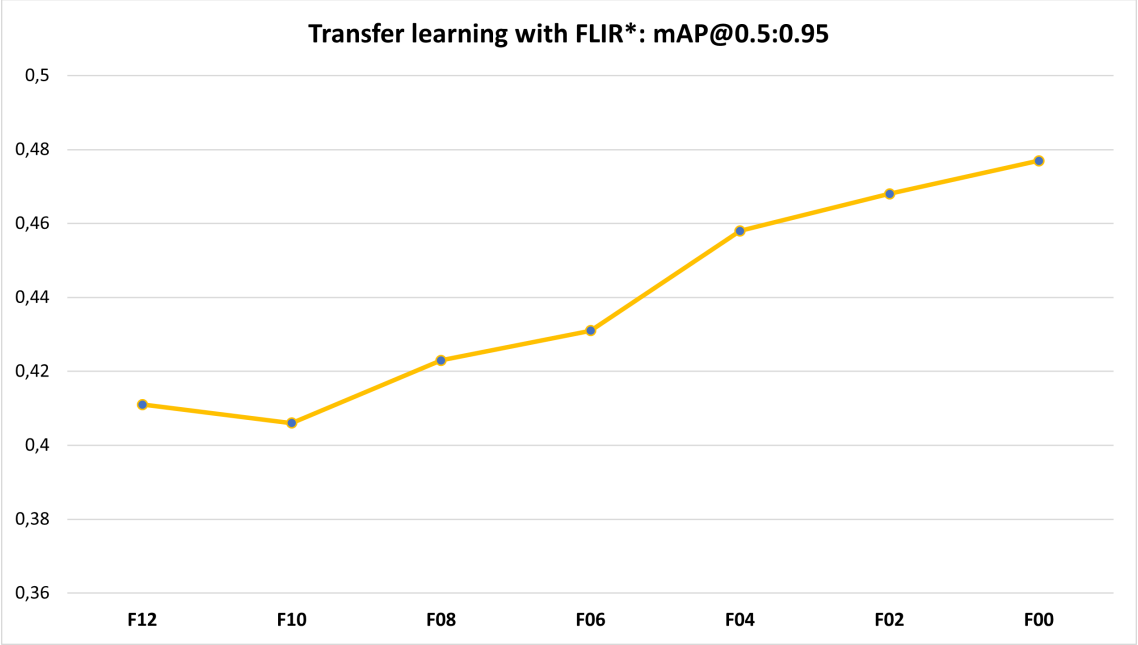


Figure 4.7: Study of the evolution of the performance when the model is fine-tuned with FLIR*. Learning rate of 0.01.

4.3 The effects of data augmentation

In this section, the results of the application of augmentation are analyzed. The parameters and probability values applied can be seen in Table 3.4, in Chapter 3. The training parameters can be seen in the same Chapter in Table 3.3. This section corresponds to the fourth research step.

It was expected to see significant improvements in all scenarios and metrics since augmentation introduces variability into the training datasets as discussed in Chapter 3, and indeed, these improvements were observed. In the previous section, Table 4.3 had the lowest results, but with augmentation, this modality (without freeze with a learning rate of 0.001) was able to surpass not only all the results (for different learning rates) from the previous sections but was also able to reach similar results with augmentation relative to Tables 4.10 (illustrated in Figures 4.8 and 4.9) and 4.12 (represented by the Figures 4.12 and 4.13). In general, this modality still underperforms, but the difference compared to Table 4.10 is not significant.

Another insight that can be taken into account is that Table 4.12 has very steady results considering the mAP@0.5:0.95 compared to other tables. Nonetheless, the learning rate of 0.01 remains the best option. On the other hand, when considering only the best three cases, the learning rate does not have much impact on the performance of these scenarios, since Table 4.11 (represented in Figures 4.10 and 4.11), with a learning rate of 0.001, has a very similar performance to Table 4.10 with a learning rate of 0.01. Table 4.12 is a bit behind, but the difference is not significant.

These results are related to the fact that augmentation improves the amount of variation the model is trained with, which means it increases generalization as well, even though the dataset did not increase. Consequently, it might be easier for the model to detect which features and patterns are more significant since there are features invariant across the various transformations. In the case of the results for a learning rate of 0.1, this might be especially true.

Nevertheless, it is still observed that the learning rate does not have as much impact as other variables, as mentioned in section 4.2. It is possible to see that it affects the discrepancy between scenarios' results. Using a learning rate of 0.1 shows more stabilized results around the same values *i.e.*, values of mAP@0.5 and mAP@0.5:0.95 are similar between modalities. Augmentation does not change this behavior, which suggests that the use of a larger learning rate allows for the model to have more variation in relation to the local minima, and with the update of the learning rate that occurs during training, it is easier for very different datasets that start at different points to converge around the same local minima. Whereas very small learning rates do not allow as much variation, as reflected in the figures, so each category might be concentrated around different values.

For SGD, a learning rate of 0.01 shows to be the best to balance variation without forgetting pre-training information.

Table 4.10: Compares the results across the various scenarios for the learning rate 0.01 with augmentation. Freeze=0.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.898	0.773	0.878	0.530	0.85 (@0.559)
RGB	0.895	0.799	0.881	0.534	0.85 (@0.547)
COCO	**0.937**	**0.835**	**0.918**	**0.601**	**0.88 (@0.626)**
FLIR*	0.922	0.830	0.906	0.567	0.87 (@0.547)
FLIR	0.918	0.811	0.893	0.557	0.86 (@0.511)
Gray	0.918	0.788	0.883	0.543	0.85 (@0.559)
Average	0.915	0.806	0.893	0.555	0.86
Average-3	0.926	0.825	0.906	0.575	0.87

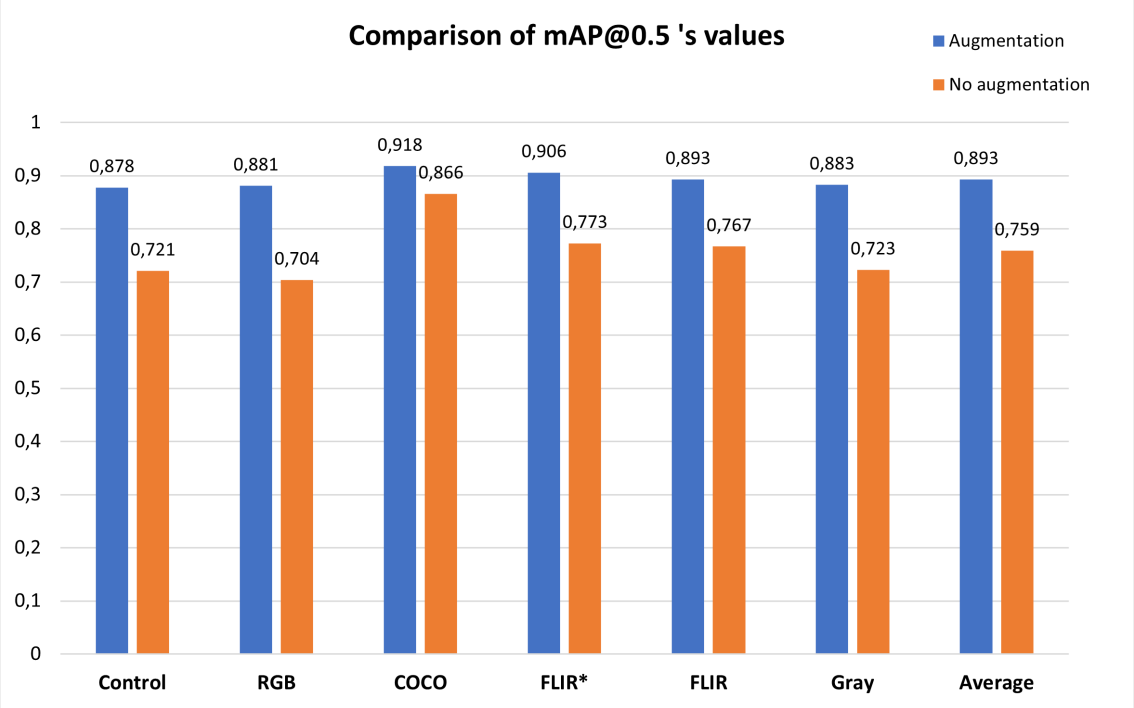


Figure 4.8: Evolution of mAP@0.5 at a learning rate of 0.01, when exists augmentation and comparison without augmentation.

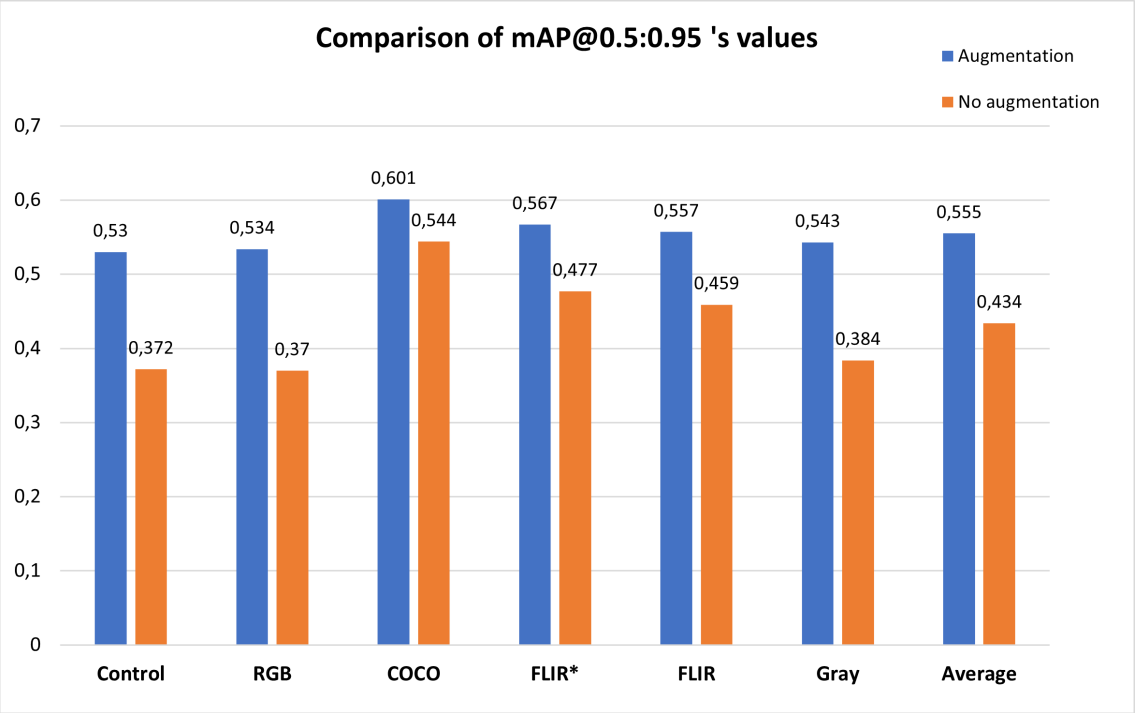


Figure 4.9: Evolution of mAP@0.5:0.95 at a learning rate of 0.01, when exists augmentation and comparison without augmentation.

Table 4.11: Comparison of the performance across the various modalities for the learning rate 0.001 with augmentation. No freeze.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.814	0.714	0.800	0.429	0.76 (@0.476)
RGB	0.848	0.742	0.84	0.488	0.79 (@0.549)
COCO	**0.931**	**0.842**	**0.921**	**0.599**	**0.88 (@0.550)**
FLIR*	0.904	0.817	0.900	0.562	0.86 (@0.551)
FLIR	0.892	0.800	0.885	0.540	0.84 (@0.507)
Gray	0.890	0.738	0.841	0.478	0.81 (@0.536)
Average	0.886	0.784	0.869	0.523	0.82
Average-3	0.909	0.820	0.902	0.567	0.86

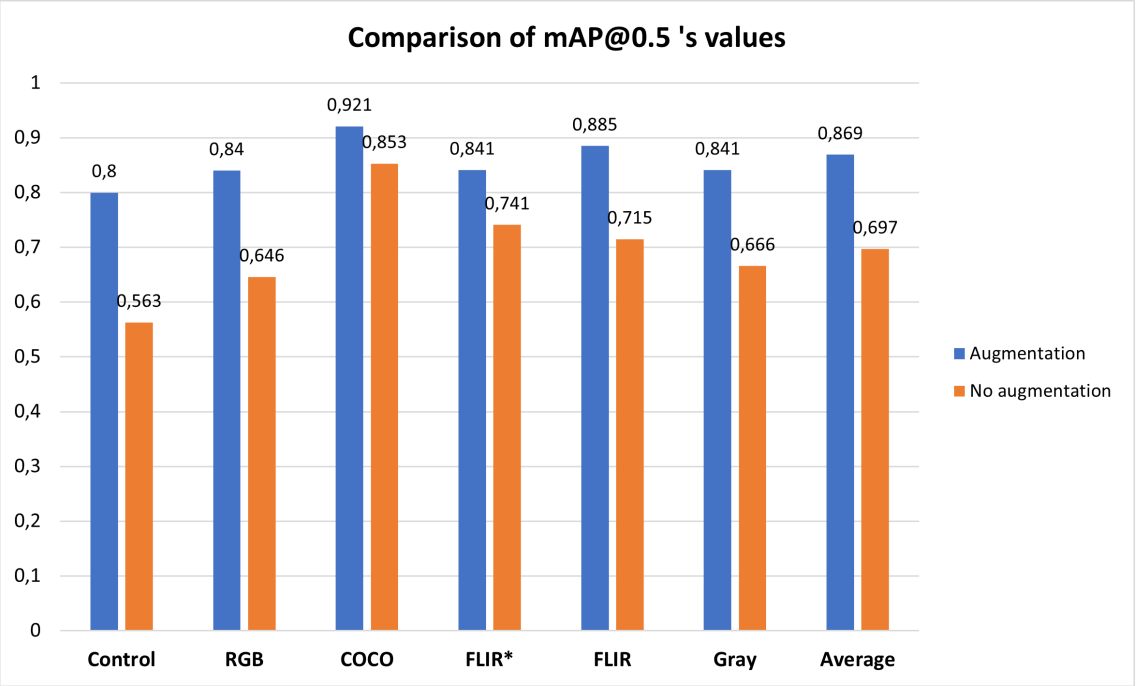


Figure 4.10: Evolution of mAP@0.5 for the learning rate of 0.001, comparing the results with and without augmentation.

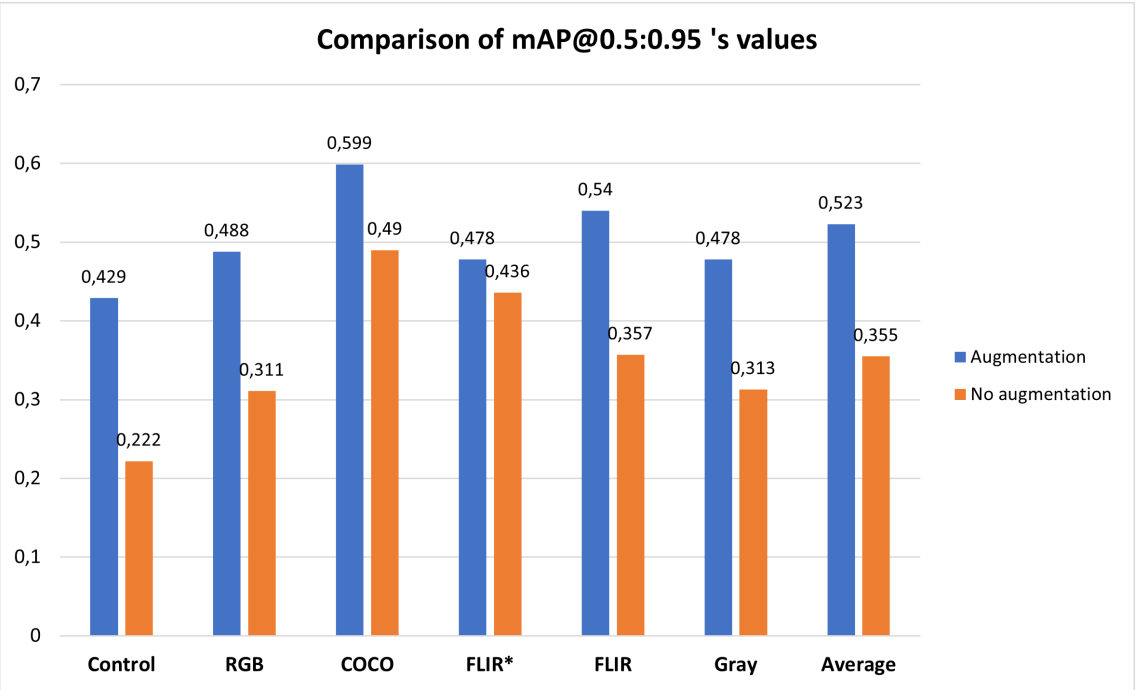


Figure 4.11: Evolution of mAP@0.5:0.95 for the learning rate of 0.001, comparing the results with and without augmentation.

Table 4.12: Compares the augmentation results across the various modalities for the learning rate 0.1, freeze=0.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.902	0.796	0.885	0.54	0.85 (@0.508)
RGB	0.903	0.800	0.884	0.544	0.85 (@0.550)
COCO	**0.927**	**0.808**	**0.901**	**0.562**	**0.86 (@0.563)**
FLIR*	0.914	0.793	0.887	0.549	0.85 (@0.529)
FLIR	0.919	0.802	0.897	0.555	0.86 (@0.518)
Gray	0.892	0.792	0.885	0.544	0.84 (@0.546)
Average	0.910	0.799	0.890	0.549	0.85
Average-3	0.920	0.801	0.895	0.555	0.86

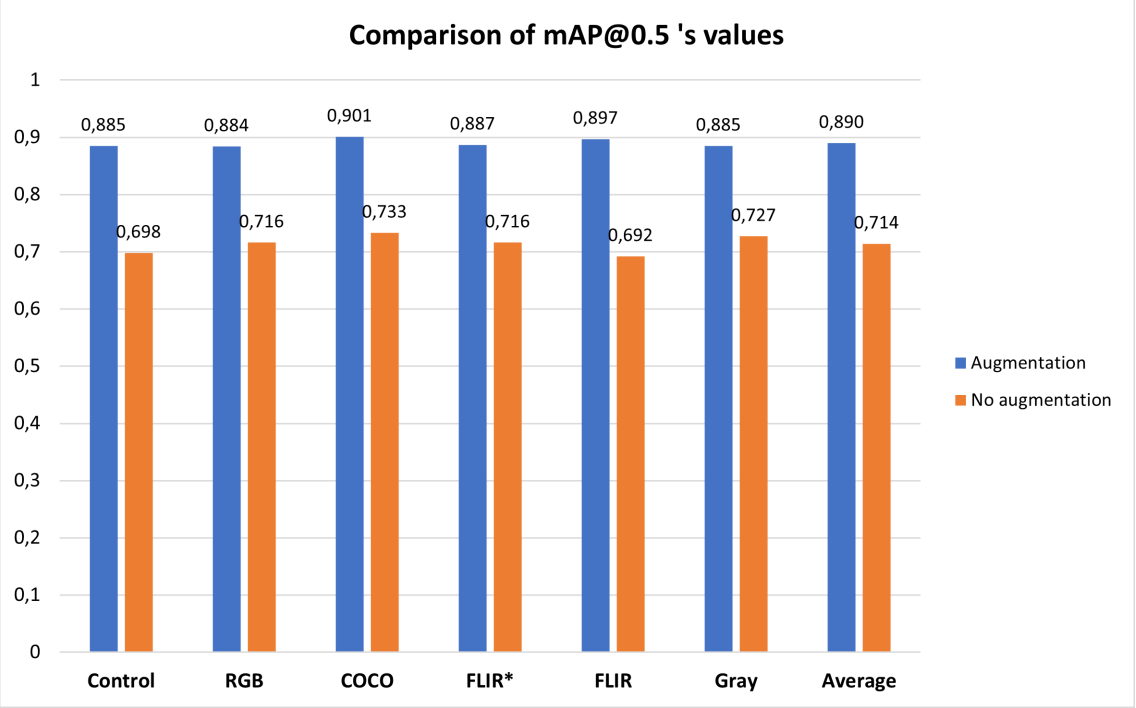


Figure 4.12: Results for mAP@0.5 at a learning rate of 0.1, with and without augmentation.

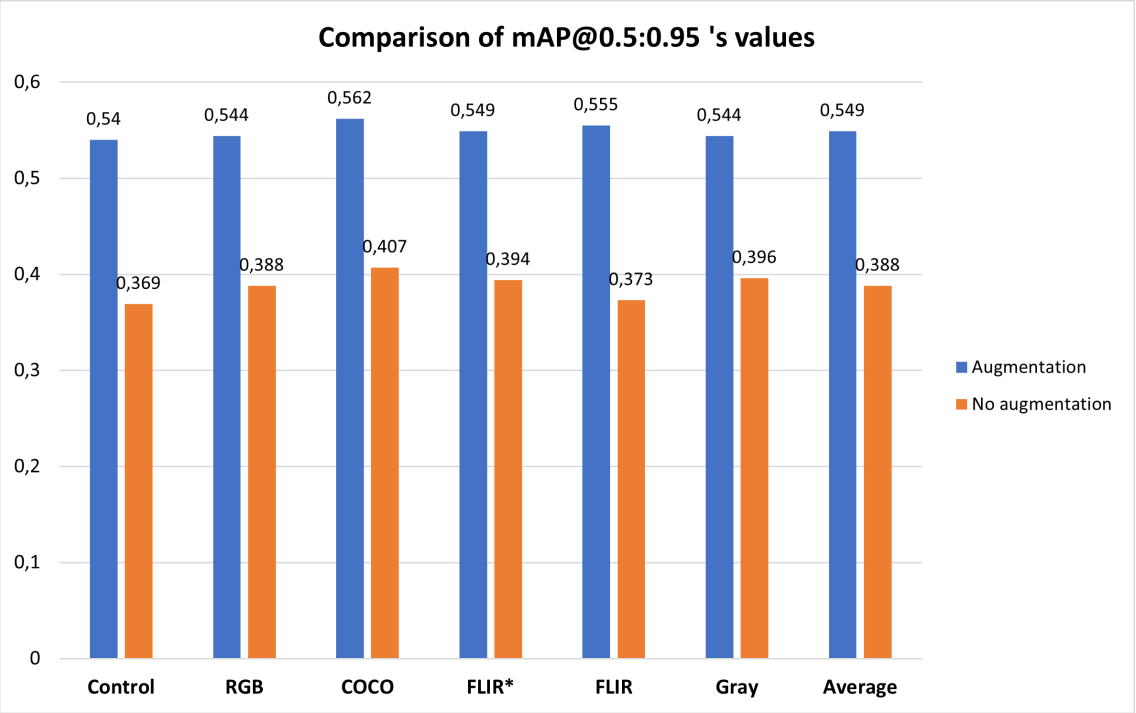


Figure 4.13: Results for mAP@0.5:0.95 at a learning rate of 0.1, with and without augmentation.

4.4 Optimizers

In this section, the ADAM optimizer was tested for two learning rates: 0.01 (results in Table 4.13), which is the best learning rate for SGD, as concluded in section 4.1; and 0.001 (results in Table 4.14), the default learning rate. These results, performed using fine-tuning with the pre-trained scenarios of COCO, RGB, and FLIR*, as well as the Control modality, were compared to the SGD results from section 4.1. The training conditions are stipulated in Table 3.3 of Chapter 3. This section corresponds to the last research step.

It is possible to notice that the model pre-trained with COCO remains the best transfer learning option either for SGD or ADAM. For a learning rate of 0.01, Table 4.16 shows that using the SGD optimizer allows for better results than using ADAM. However, when using an initial learning rate of 0.001, ADAM is the best option for the majority of the scenarios, as evidenced in Table 4.16. In this last case, COCO's value of mAP@0.5 differs and is better for the SGD optimizer.

It was also tested if using a learning rate of 0.001 with ADAM performed better than the best scenario with SGD with the learning rate of 0.01. Table 4.17 illustrates the case for RGB and Control scenarios. COCO and FLIR* still achieve better performance with SGD; however, the difference is not significant.

It was observed that using ADAM increases the time it takes to run the model. In the previous sections, none of the scenarios surpassed the 30-minute mark; however, using the ADAM optimizer, it would take as much as 50 minutes to run. This can be explained by the nature of the two opti-

mizers. Even though ADAM is known to converge faster due to the implementation of momentum and RMSprop concepts that allow rapid convergence, the amount of data that it has to consider might be slowing down the optimizer since it also considers previous iterations' gradients. On the other hand, SGD considers the current point and weight to update the loss function; consequently, it does not require as much computational memory.

Table 4.13: Compares the results with ADAM optimizer across the various scenarios for the learning rate 0.01. Freeze=0, no augmentation.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.763	0.576	0.675	0.353	0.66 (@0.306)
RGB	0.729	0.628	0.692	0.368	0.67 (@0.160)
COCO	0.822	0.626	0.739	0.391	0.70 (@0.208)
FLIR*	0.753	0.612	0.69	0.365	0.71 (@0.409)

Table 4.14: Compares the results across the various situations with ADAM optimizer for the learning rate of 0.001 (recommended initial learning rate). Freeze=0, no augmentation.

	Precisions	Recall	mAP@0.5	mAP@0.5:0.95	F1 Score
Control	0.817	0.615	0.729	0.393	0.70 (@0.572)
RGB	0.800	0.617	0.712	0.388	0.70 (@0.391)
COCO	0.836	0.76	0.835	0.504	0.80 (@0.362)
FLIR*	0.82	0.692	0.787	0.465	0.75 (@0.255)

Table 4.15: 0.01 ADAM vs 0.01 SGD

	0.01 ADAM		0.01 SGD	
	mAP@0.5	mAP@0.95	mAP@0.5	mAP@0.95
Control	0.675	0.353	**0.721**	**0.372**
RGB	0.692	0.368	**0.704**	**0.37**
COCO	0.739	0.391	**0.866**	**0.544**
FLIR*	0.69	0.365	**0.773**	**0.477**

Table 4.16: 0.001 ADAM vs 0.001 SGD

	0.001 ADAM		0.001 SGD	
	mAP@0.5	mAP@0.95	mAP@0.5	mAP@0.95
Control	**0.729**	**0.393**	0.582	0.225
RGB	**0.712**	**0.388**	0.646	0.311
COCO	0.835	**0.504**	**0.853**	0.49
FLIR*	**0.787**	**0.465**	0.741	0.436

Table 4.17: 0.001 ADAM vs 0.01 SGD

	0.01 SGD		0.001 ADAM	
	mAP@0.5	mAP@0.95	mAP@0.5	mAP@0.95
Control	0.721	0.372	**0.729**	**0.393**
RGB	0.704	0.37	**0.712**	**0.388**
COCO	**0.866**	**0.544**	0.835	0.504
FLIR*	0.773	**0.477**	**0.787**	0.465

5 Conclusion

The main goal of this thesis was to investigate the effectiveness of YOLOv5 in the context of human detection, using thermal imagery, and its performance when pre-trained with several datasets (RGB, FLIR, Grayscale, and COCO) - and fine-tuned in a thermal indoor dataset. We then proceeded to compare this performance with a baseline scenario where the model was trained from scratch with the thermal dataset created in ISR.

The thermal dataset was trained in numerous different conditions: different number of frozen layers, with or without data augmentation, and with different optimizers. Significant conclusions can be drawn from the reported results. First, even though in multiple scenarios the learning rate had a certain impact on the performances, the behavior observed through the different tables remained stable *i.e.*, RGB and Control were often performing worse than the others. Although the Grayscale scenario seemed to frequently outperform RGB, the results were very similar. COCO's performances were the ones to suffer less variation, leading to COCO remaining as the best scenario, followed by FLIR*, which achieved its peak with a learning rate of 0.01. This suggests that it is preferable to use a large RGB dataset since it allows for a bigger generalization than using a smaller but thermal one to pre-train.

It was not expected for RGB to perform so badly since it is the RGB version of the target dataset, it has a high degree of similarity. It is interesting to notice that RGB reaches a peak in accuracy when the backbone is frozen, which contrasts with pre-training with datasets like COCO and FLIR*, where the model achieves better results when almost none to no layers are frozen. This suggests that, even though the outcomes of fine-tuning with RGB may not be very high, the dataset is capable of contributing with more specific and high-level information to the model, while COCO and FLIR* contribute with low-level features. The same can be inferred for the other scenarios that achieved lower performances with a frozen backbone.

The results also help to conclude that the use of data augmentation, even when it does not increase the training set, is still extremely useful for fine-tuning since it introduces variation that helps enhance dataset generalization and performance. This permitted different training scenarios to achieve similar results to those observed in the main results section.

Concerning the learning rates, for the optimizer SGD, 0.01 shows to be the best option since it is capable of balancing pre-train information with variation. It is not high enough to overshoot,

but it is also not small enough to either need more epochs to converge or prevent finding the global local minima.

The fact that across all situations, COCO was capable of remaining the most efficient dataset for pre-training shows that the dataset size and the information it contains are the two variables that had the most impact out of all the variables studied. Although the other modalities were capable of improving their performances, they never achieved the same results as COCO. Larger datasets like COCO contribute to increased performance since they allow the model to learn from a broader range of features and patterns. In the same sense, this is confirmed by the fact that pre-training with the RGB dataset frequently underperformed, although this dataset corresponds to the target dataset in its RGB format. Adding to that, it is also an RGB format dataset like COCO, therefore it is lacking sufficient size and, consequently, informational content might be related to its poor results. Despite FLIR and FLIR* being large datasets composed of thermal imagery as well, the fact that they underperformed relative to the COCO dataset can be because the dataset has repeated scenarios, since the interval between the times the images were taken is small, and in numerous of them there are no people. This might have undermined the results.

The choice of the optimizer also significantly impacts the model. For a learning rate of 0.01, SGD generally outperforms ADAM. However, when using a learning rate of 0.001, ADAM performs better for most scenarios. Nevertheless, it has also to be taken into account that ADAM increases computational time compared to SGD due to the amount of information computed during iterations.

In summary, the choice of a pre-training dataset, freezing layers, augmentation, learning rate, and optimizer are all crucial for the success of object detection. However, some variables showed a greater impact, such as the size of data and the amount of diversity it contains. This diversity can also be reached with augmentation.

5.1 Future Work

As future work, it would be interesting to dive into the impact of different optimizers and evaluate the results, as well as include the change of more hyperparameters (beyond those shown in Table 3.3).

It would also be relevant to study the impact of augmentation by increasing the dataset instead of only including variation to see if the performance increases or even surpasses the performance of fine-tuning a model with pre-trained COCO dataset. Alongside this study, exploring transfer learning with freeze in a larger, thermal indoor dataset might result in better performance or, at least, with both indoor and outdoor environments. Not only because they might introduce more generalization and decrease the number of repeated scenarios, but also because they might include more similar scenarios as well. Furthermore, another future direction to pursue could include the

impact of different transfer learning methods, particularly the ones described in Chapter 2, especially with the Domain Adaptation Framework, which transfers low-level features through GAN.

Bibliography

[1] Abhishek Balasubramaniam and Sudeep Pasricha. "Object Detection in Autonomous Vehicles: Status and Open Challenges". In: (2022), pp. 1–6. URL: http://arxiv.org/abs/2201.07706.

[2] Ganbayar Batchuluun et al. "A Study on the Elimination of Thermal Reflections". In: (2019).

[3] Ganbayar Batchuluun et al. "Deep Learning-Based Thermal Image Reconstruction and Object Detection". In: *IEEE Access* 9 (2021), pp. 5951–5971. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3048437. URL: https://ieeexplore.ieee.org/document/9311732/.

[4] *BCE with Logists Loss*. https://pytorch.org/docs/master/generated/torch.nn.BCEWithLogitsLoss.html. Accessed: 27-08-2023.

[5] Bharat Bhushan, Simran Singh, and Ruchika Singla. "License Plate Recognition System using Neural Networks and Multithresholding Technique". In: *International Journal of Computer Applications* 84 (2013), pp. 45–50. URL: https://api.semanticscholar.org/CorpusID:1191747.

[6] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *CoRR* abs/2004.10934 (2020). arXiv: 2004.10934. URL: https://arxiv.org/abs/2004.10934.

[7] Yukyung Choi et al. "KAIST Multi-Spectral Day/Night Data Set for Autonomous and Assisted Driving". In: *IEEE Transactions on Intelligent Transportation Systems* 19.3 (2018), pp. 934–948. DOI: 10.1109/TITS.2018.2791533.

[8] Siu Chung et al. "Current State of the Art in Object Detection for Autonomous Systems". In: (September 2021).

[9] N. Dalal and B. Triggs. "Histograms of oriented gradients for human detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. Vol. 1. 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[10] Kevser Irem Danaci and Erdem Akagunduz. *A Survey on Infrared Image and Video Sets*. 2023. arXiv: 2203.08581 [cs.CV].

[11] Rikke Gade and Thomas Baltzer Moeslund. "Thermal cameras and applications: a survey". In: *Machine Vision and Applications* 25 (2013), pp. 245–262. URL: https://api.semanticscholar.org/CorpusID:8446711.

[12] Rikke Gade et al. "Automatic Occupancy Analysis of Sports Arenas". In: 2012. URL: https://api.semanticscholar.org/CorpusID:62324133.

[13] Mingyu Gao et al. "Adaptive anchor box mechanism to improve the accuracy in the object detection system". In: *Multimedia Tools and Applications* 78 (19 Oct. 2019), pp. 27383–27402. ISSN: 1380-7501. DOI: 10.1007/s11042-019-07858-w. URL: http://link.springer.com/10.1007/s11042-019-07858-w.

[14] Debasmita Ghose et al. *Pedestrian Detection in Thermal Images using Saliency Maps*. 2019. arXiv: 1904.06859 [cs.CV].

[15] Ismat Saira Gillani et al. "Yolov5, Yolo-x, Yolo-r, Yolov7 Performance Comparison: A Survey". In: Academy and Industry Research Collaboration Center (AIRCC), Sept. 2022, pp. 17–28. ISBN: 9781925953763. DOI: 10.5121/csit.2022.121602. URL: https://aircconline.com/csit/papers/vol12/csit121602.pdf.

[16] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[17] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[18] Yanfeng Gong et al. "A transfer learning object detection model for defects detection in X-ray images of spacecraft composite structures". In: *Composite Structures* 284 (2022), p. 115136. ISSN: 0263-8223. DOI: https://doi.org/10.1016/j.compstruct.2021.115136. URL: https://www.sciencedirect.com/science/article/pii/S0263822321015531.

[19] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. Cambridge, MA, USA: MIT Press, 2016.

[20] Anishi Gupta and Uma Gupta. "Real Time Target Detection for Infrared Images". In: IEEE, Jan. 2020, pp. 570–574. ISBN: 978-1-7281-2813-9. DOI: 10.1109/ICISC47916.2020.9171208. URL: https://ieeexplore.ieee.org/document/9171208/.

[21] Kaiming He et al. "Mask R-CNN". In: (Mar. 2017). URL: http://arxiv.org/abs/1703.06870.

[22] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361. DOI: 10.1007/978-3-319-10578-9_23. URL: https://doi.org/10.1007%2F978-3-319-10578-9_23.

[23] Steven A Hicks et al. "On evaluation metrics for medical applications of artificial intelligence". In: (). DOI: 10.1101/2021.04.07.21254975. URL: https://doi.org/10.1101/2021.04.07.21254975.

[24] Marko Horvat, Ljudevit Jelečević, and Gordan Gledec. *A comparative study of YOLOv5 models performance for image localization and classification Hascheck-Croatian Academic Spelling Checker View project*. 2022. URL: https://www.researchgate.net/publication/363824867.

[25] Mohammadreza Iman, Hamid Reza Arabnia, and Khaled Rasheed. "A Review of Deep Transfer Learning and Recent Advancements". In: *Technologies* 11.2 (Mar. 2023), p. 40. DOI: 10.3390/technologies11020040. URL: https://doi.org/10.3390%2Ftechnologies11020040.

[26] Marina Ivašić-Kos, Mate Krišto, and Miran Pobar. "Human Detection in Thermal Imaging Using YOLO". In: vol. Part F1482. ACM, Apr. 2019, pp. 20–24. ISBN: 9781450371810. DOI: 10.1145/3323933.3324076. URL: https://dl.acm.org/doi/10.1145/3323933.3324076.

[27] Licheng Jiao et al. "A Survey of Deep Learning-Based Object Detection". In: *IEEE Access* 7 (July 2019), pp. 128837–128868. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2939201. URL: https://ieeexplore.ieee.org/document/8825470/.

[28] Glenn Jocher. *Ultralytics YOLOv5*. Version 7.0. 2020. DOI: 10.5281/zenodo.3908559. URL: https://github.com/ultralytics/yolov5.

[29] *Joseph Redmon on why he abandoned investigation*. https://shorturl.at/cpzFV. Accessed: 01-08-2023.

[30] Hyun-Ki Jung and Gi-Sang Choi. "Improved YOLOv5: Efficient Object Detection Using Drone Images under Various Conditions". In: *Applied Sciences* 12 (July 2022), p. 7255. DOI: 10.3390/app12147255.

[31] M. Kastek et al. "Concept of infrared sensor module for sniper detection system". In: *35th International Conference on Infrared, Millimeter, and Terahertz Waves*. 2010, pp. 1–2. DOI: 10.1109/ICIMW.2010.5612447.

[32] Shreyas Bhat Kera, Anand Tadepalli, and J. Jennifer Ranjani. "A paced multi-stage block-wise approach for object detection in thermal images". In: *Visual Computer* 39 (6 June 2023), pp. 2347–2363. ISSN: 01782789. DOI: 10.1007/s00371-022-02445-x.

[33] Yash Khandhediya, Karishma Sav, and Vandit Gajjar. *Human Detection for Night Surveillance using Adaptive Background Subtracted Image*. 2017. arXiv: 1709.09389 [cs.CV].

[34] Mate Kristo, Marina Ivasic-Kos, and Miran Pobar. "Thermal Object Detection in Difficult Weather Conditions Using YOLO". In: *IEEE Access* 8 (2020), pp. 125459–125476. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2020.3007481. URL: https://ieeexplore.ieee.org/document/9133581/.

[35] Maarten C. Kruithof et al. "Object recognition using deep convolutional neural networks with complete transfer and partial frozen layers". In: *Optics and Photonics for Counterterrorism, Crime Fighting, and Defence XII*. Ed. by Douglas Burgess et al. Vol. 9995. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series. Oct. 2016, 99950K, 99950K. DOI: 10.1117/12.2241177.

[36] Chengyang Li et al. *Illumination-aware Faster R-CNN for Robust Multispectral Pedestrian Detection*. 2018. arXiv: 1803.05347 [cs.CV].

[37] Xiaoyu Li et al. "Improved YOLOv4 network using infrared images for personnel detection in coal mines". In: *Journal of Electronic Imaging* 31, 013017 (Jan. 2022), p. 013017. DOI: 10.1117/1.JEI.31.1.013017.

[38] Tsung-Yi Lin et al. *Feature Pyramid Networks for Object Detection*. 2017. arXiv: 1612.03144 [cs.CV].

[39] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[40] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. 2015. arXiv: 1405.0312 [cs.CV].

[41] Shu Liu et al. "Path Aggregation Network for Instance Segmentation". In: *CoRR* abs/1803.01534 (2018). arXiv: 1803.01534. URL: http://arxiv.org/abs/1803.01534.

[42] Wei Liu et al. *SSD: Single Shot MultiBox Detector*. Dec. 2016. DOI: 10.1007/978-3-319-46448-0_2. URL: http://link.springer.com/10.1007/978-3-319-46448-0_2.

[43] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: (Dec. 2016), pp. 21–37. DOI: 10.1007/978-3-319-46448-0_2. URL: http://link.springer.com/10.1007/978-3-319-46448-0_2.

[44] Mingsheng Long et al. *Conditional Adversarial Domain Adaptation*. 2018. arXiv: 1705.10667 [cs.LG].

[45] Mingsheng Long et al. *Deep Transfer Learning with Joint Adaptation Networks*. 2017. arXiv: 1605.06636 [cs.LG].

[46] Chris Lytridis et al. "An Overview of Cooperative Robotics in Agriculture". In: *Agronomy* 11.9 (2021). ISSN: 2073-4395. DOI: 10.3390/agronomy11091818. URL: https://www.mdpi.com/2073-4395/11/9/1818.

[47] Yibing Ma et al. "Smart Fire Alarm System with Person Detection and Thermal Camera". In: *Computational Science – ICCS 2020* 12143 (2020), pp. 353–366. URL: https://api.semanticscholar.org/CorpusID:219967638.

[48] Aprinaldi Jasa Mantau et al. "A Human-Detection Method Based on YOLOv5 and Transfer Learning Using Thermal Image Data from UAV Perspective for Surveillance System". In: *Drones* 6 (10 Oct. 2022), p. 290. ISSN: 2504-446X. DOI: 10.3390/drones6100290. URL: https://www.mdpi.com/2504-446X/6/10/290.

[49] Usha Mittal, Sonal Srivastava, and Priyanka Chawla. "Object detection and classification from thermal images using region based convolutional neural network". In: *Journal of Computer Science* 15 (7 2019), pp. 961–971. ISSN: 15526607. DOI: 10.3844/jcssp.2019.961.971.

[50] Farzeen Munir et al. *Exploring Thermal Images for Object Detection in Underexposure Regions for Autonomous Driving*. 2021. arXiv: 2006.00821 [cs.CV].

[51] Nermin K. Negied, Elsayed E. Hemayed, and Magda B. Fayek. "Pedestrians' detection in thermal bands – Critical survey". In: *Journal of Electrical Systems and Information Technology* 2.2 (2015), pp. 141–148. ISSN: 2314-7172. DOI: https://doi.org/10.1016/j.jesit.2015.06.002. URL: https://www.sciencedirect.com/science/article/pii/S2314717215000343.

[52] Yu Oishi et al. "Animal Detection Using Thermal Images and Its Required Observation Conditions". In: *Remote Sensing* 10.7 (2018). ISSN: 2072-4292. DOI: 10.3390/rs10071050. URL: https://www.mdpi.com/2072-4292/10/7/1050.

[53] Sankar K. Pal et al. "Deep learning in multi-object detection and tracking: state of the art". In: *Applied Intelligence* 51 (9 Sept. 2021), pp. 6400–6429. ISSN: 0924-669X. DOI: 10.1007/s10489-021-02293-7. URL: https://link.springer.com/10.1007/s10489-021-02293-7.

[54] Yanwei Pang et al. "Mask-Guided Attention Network for Occluded Pedestrian Detection". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. Oct. 2019.

[55] Jan Portmann et al. "People detection and tracking from aerial thermal views". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 2014, pp. 1794–1800. DOI: 10.1109/ICRA.2014.6907094.

[56] Rizwan Qureshi et al. "A Comprehensive Systematic Review of YOLO for Medical Object Detection (2018 to 2023)". In: (July 2023). DOI: 10.36227/techrxiv.23681679.v1. URL: https://www.techrxiv.org/articles/preprint/A_Comprehensive_Systematic_Review_of_YOLO_for_Medical_Object_Detection_2018_to_2023_/23681679.

[57] Jakaria Rabbi et al. "Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network". In: *Remote Sensing* 12.9 (2020). ISSN: 2072-4292. DOI: 10.3390/rs12091432. URL: https://www.mdpi.com/2072-4292/12/9/1432.

[58] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* 2016-Decem (June 2015), pp. 779–788. ISSN: 10636919. DOI: 10.1109/CVPR.2016.91. URL: http://arxiv.org/abs/1506.02640.

[59] Pascal Schneider et al. "TIMomdash;A Dataset for Indoor Building Monitoring with a Time-of-Flight Camera". In: *Sensors* 22.11 (2022). ISSN: 1424-8220. DOI: 10.3390/s22113992. URL: https://www.mdpi.com/1424-8220/22/11/3992.

[60] The World's Sixth Sense. *FREE Teledyne FLIR Thermal Dataset for Algorithm Training*. data retrieved from Teledyne FLIR. URL: https://www.flir.com/oem/adas/adas-dataset-form/.

[61] Zhengqiang Shao et al. "A Dataset and A Lightweight Object Detection Network for Thermal Image-Based Home Surveillance". In: IEEE, Nov. 2022, pp. 1332–1336. ISBN: 978-616-590-477-3. DOI: 10.23919/APSIPAASC55919.2022.9980050. URL: https://ieeexplore.ieee.org/document/9980050/.

[62] Tasfia Shermin et al. *Transfer Learning Using Classification Layer Features of CNN*. 2019. arXiv: 1811.07459 [cs.CV].

[63] Marcel Simon, Erik Rodner, and Joachim Denzler. *ImageNet pre-trained models with batch normalization*. 2016. arXiv: 1612.01452 [cs.CV].

[64] Rajkumar Soundrapandiyan and P.V.S.S.R. Chandra Mouli. "Adaptive Pedestrian Detection in Infrared Images Using Background Subtraction and Local Thresholding". In: *Procedia Computer Science* 58 (2015). Second International Symposium on Computer Vision and the Internet (VisionNet'15), pp. 706–713. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.08.091. URL: https://www.sciencedirect.com/science/article/pii/S1877050915022024.

[65] P. Srihari, Jonnadula Harikiran, and Bolem Sai Chandana. "A Comparative Analysis of Object Detection and Human Activity Recognition in Thermal Imaging". In: IEEE, Sept. 2022, pp. 1590–1596. ISBN: 978-1-6654-9707-7. DOI: 10.1109/ICIRCA54612.2022.9985697. URL: https://ieeexplore.ieee.org/document/9985697/.

[66] Chuanqi Tan et al. *A Survey on Deep Transfer Learning*. 2018. arXiv: 1808.01974 [cs.LG].

[67] Juan Terven and Diana Cordova-Esparza. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. arXiv: 2304.00501 [cs.CV].

[68] Nguyen Duc Thuan, Le Hai Anh, and Hoang Si Hong. "PDIWS: Thermal Imaging Dataset for Person Detection in Intrusion Warning Systems". In: *2023 IEEE Statistical Signal Processing Workshop (SSP)*. 2023, pp. 71–75. DOI: 10.1109/SSP53291.2023.10208055.

[69] Eric Tzeng et al. *Deep Domain Confusion: Maximizing for Domain Invariance*. 2014. arXiv: 1412.3474 [cs.CV].

[70] "Unsupervised Transfer Learning via Multi-Scale Convolutional Sparse Coding for Biomedical Applications". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40 (5 May 2018), pp. 1182–1194. ISSN: 01628828. DOI: 10.1109/TPAMI.2017.2656884.

[71] Paul Viola and Michael Jones. "Robust Real-Time Face Detection". In: *International Journal of Computer Vision* 57 (May 2004), pp. 137–154. DOI: 10.1023/B:VISI.0000013087.49260.fb.

[72] Liangtian Wan et al. "UAV swarm based radar signal sorting via multi-source data fusion: A deep transfer learning framework". In: *Information Fusion* 78 (2022), pp. 90–101. ISSN: 1566-2535. DOI: https://doi.org/10.1016/j.inffus.2021.09.007. URL: https://www.sciencedirect.com/science/article/pii/S1566253521001834.

[73] Yonghui Xu et al. "A Unified Framework for Metric Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 29 (2017), pp. 1158–1171. URL: https://api.semanticscholar.org/CorpusID:2745878.

[74] Shuo Yang et al. *Face Detection through Scale-Friendly Deep Convolutional Networks*. 2017. arXiv: 1706.02863 [cs.CV].

[75] Zhi Yang et al. "Deep transfer learning for military object recognition under small training set condition". In: *Neural Computing and Applications* 31 (10 Oct. 2019), pp. 6469–6478. ISSN: 14333058. DOI: 10.1007/s00521-018-3468-3.

[76] Jason Yosinski et al. *How transferable are features in deep neural networks?* 2014. arXiv: 1411.1792 [cs.LG].

[77] Syed Sahil Abbas Zaidi et al. *A Survey of Modern Deep Learning based Object Detection Models*. 2021. arXiv: 2104.11892 [cs.CV].

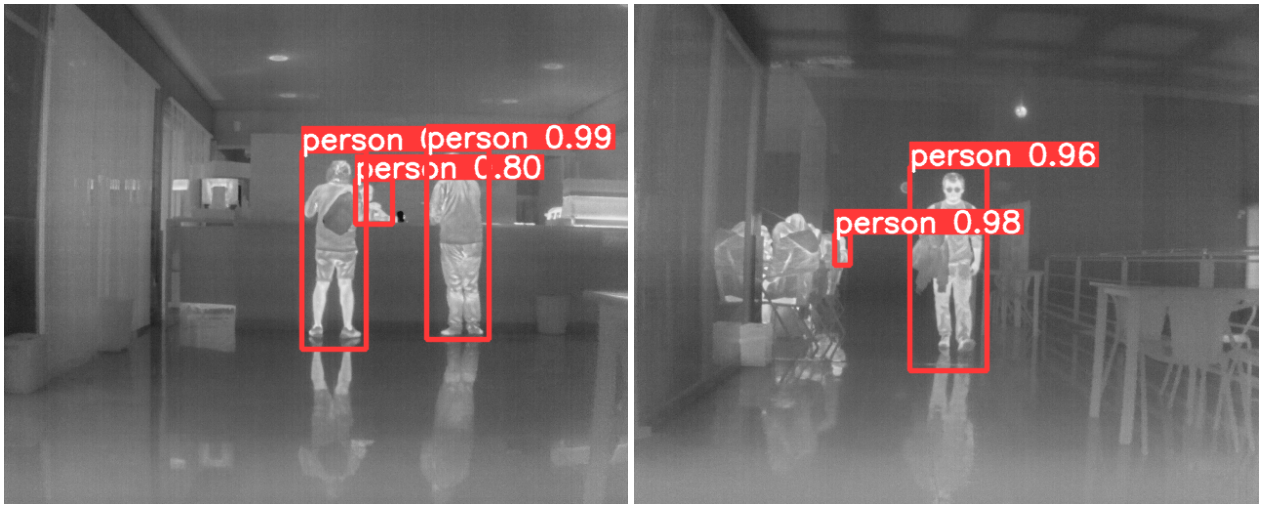
[78] Zhong-Qiu Zhao et al. "Object Detection With Deep Learning: A Review". In: *IEEE Transactions on Neural Networks and Learning Systems* 30 (11 Nov. 2019), pp. 3212–3232. ISSN: 2162-237X. DOI: 10.1109/TNNLS.2018.2876865. URL: https://ieeexplore.ieee.org/document/8627998/.

[79] Zhengxia Zou et al. "Object Detection in 20 Years: A Survey". In: (May 2019). URL: http://arxiv.org/abs/1905.05055.

.1 Detection examples from the thermal indoor dataset

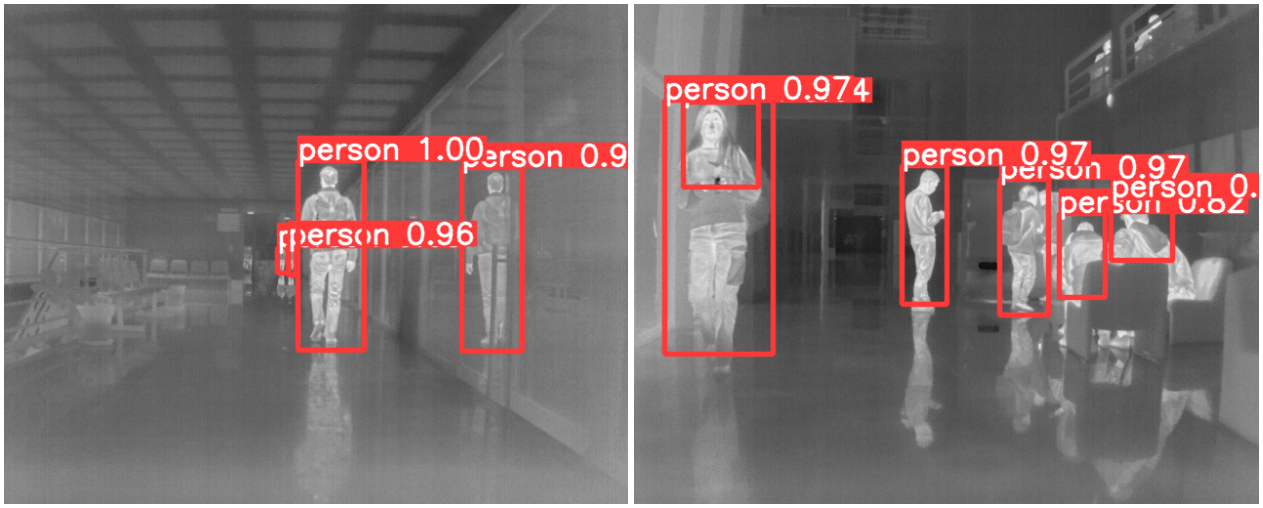
(a) No detection. **(b)** Successful detection.

Figure 1: Examples of unsuccessful and successful detection..



(a) Occlusion that did not interfere with detection. **(b)** Occlusion that interfered with detection.

Figure 2: Occlusion.



(a) Reflection. **(b)** Sobreposition of labels.

Figure 3: Examples of reflection detection.