Polina Timofeeva

Object detection in thermal imagery for crowd density estimation

Metropolia University of Applied Sciences

Technology

Information Technology

Thesis

06.04.2020

Author(s) Title Number of Pages Date	Polina Timofeeva Object detection in thermal imagery for crowd density estimation 56 pages + 0 appendices 6 April 2020
Degree	Technology
Degree Programme	Information Technology
Specialisation option	Software Engineering
Instructor(s)	Janne Salonen, Head of Department (Metropolia UAS) Mikael Lindblad, Site Project Lead (Nokia)

This work represents a research on deep-learning based approaches for object detection for the purpose of crowd density estimation. Specifically, thermal imagery which allows to preserve personal identity was used to train a deep learning object detection system. The ultimate objective of this work is to provide the client company with the tool which would facilitate better understanding of how various office, meeting, common and work spaces are being used across the campus, optimize crowd flows and drive down maintenance costs.

Theoretical background related to both, traditional computer vision and novel deep-learning methods for object detection was studied and outlined in this work. Based on the outcomes of such comparison, the most promising method was implemented into production.

Extensive empirical evidence obtained through extensive testing of the proposed solution demonstrated that the model exhibits high accuracy, great generalization capabilities and robustness against various perturbations in input images. It was concluded that provided solution satisfies both, accuracy and inference time requirements and therefore qualified to be deployed into production.

Finally, possible directions for further development were outlined. Improved performance can be achieved by alternating backbone network architecture and expanding the training dataset.

Keywords	computer vision, object detection, deep learning, RetinaNet

Contents

List of abbreviations

SWIR	Short Wave Infrared Radiation
MWIR	Mid Wave Infrared Radiation
LWIR	Long Wave Infrared Radiation
IR	Infrared
CV	Computer Vision
SIFT	Scale-Invariant Feature Transform
SURF	Speeded Up Robust Features
AdaBoost	Adaptive Boosting
HOG	Histogram of Oriented Gradients
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
DL	Deep Learning
ML	Machine Learning
CNN	Convolutional Neural Network
SSD	Single-Shot Detector
YOLO	You Only Look Once
FCN	Fully Convolutional Network
RoI	Regions of Interest
RPN	Region Proposal Network
GPU	Graphics Processing Unit
mAP	Mean Average Precision
CPU	Central Processing Unit

1 Introduction

Automated video surveillance has seen an exploding rise of adoption over the past decade. Steady growth in computational power, volume of available data storages and quality of various video cameras allows today to record and store vast amounts of data. Wide range of industries such as security, smart homes and cities, retail and quality control has been leveraging ability to automatically process huge amounts of video data and extract useful insights from it.

Recently, video analysis has been finding its application for problems of crowd behavior monitoring and understanding. Main focus of this work is to find a solution to accurately estimate amount of people present in a given facility. Having means to acquire such information would allow to better understand how various office, meeting, common and workspaces are being used across the campus. Consequently, it can help to optimize crowd flow while making premises more attractive for both, campus employees and visitors, and drive down maintenance costs.

This project has been carried out for Nokia Solutions and Networks Oy in collaboration with LEVITEZER Oy. A set of thermal cameras provided by LEVITEZER company is installed in different key locations across Nokia campus, Espoo. Using a stream of thermal images captured by the cameras, an object detection model allows to extract information about area occupancy whilst completely securing privacy. However, after overviewing existing object detection methods and related work, it becomes evident that the nature of thermal images poses its own constraints and brings challenges that make it hard to adjust any of traditional classic computer vision algorithms for the problem in question.

Therefore, this work turns towards recently emerged and rapidly developing field of deep learning (DL) which has been persistently showing highest performance for various problems in domain of computer vision [1]. The main objective of this work is to investigate possible deep learning based solutions for an accurate indoor crowd estimation using thermal cameras. A range of potentially suitable approaches is tested in practice and documented. Solution which showed the best performance is implemented in production.

2 Theoretical Background

2.1 Thermal Images

All objects emit infrared radiation. In the far infrared range this radiation is perceived as heat. Normally people and animals emit more infrared radiation than their surrounding environment. Special infrared vision systems are used to capture the heat emitted by objects to create a thermal image. Hence thermal imagery can be described as a result of capturing electromagnetic radiation emitted or reflected from an object in the infrared range of the electromagnetic spectrum.

The infrared spectrum consists of light with wavelengths between 0.8 to 14 µm [2, p. 9]. The spectrum can be divided into three different ranges: short wave infrared radiation (SWIR), mid wave infrared radiation (MWIR) and long wave infrared radiation (LWIR). These ranges are visible in the spectrum in Figure 1.

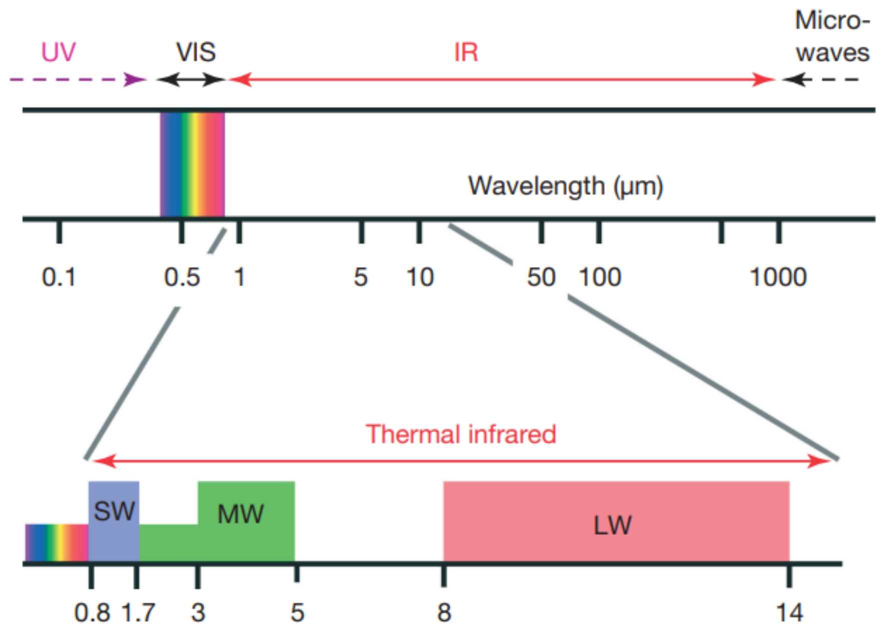


Figure 1. Expanded view of Infrared (IR) and adjacent spectral regions. Copied from [2, p. 10].

Thermal infrared region depicted in Figure 1 is the region for which different IR imaging systems exist. However, special cameras might have wider ranges.

Clearly specialized thermal cameras differ from conventional cameras that work with visual spectrum. Compared to them, thermal cameras have an ability to capture images in total darkness, they are robust to illumination changes, and are less intrusive since it is nearly impossible to identify a person in a thermal image.

Furthermore, same materials have different properties when captured by cameras for the thermal and visual spectrum and consequently they look different [3]. For example, in thermal imagery, there are no shadows because only emitted radiation is captured by the camera.

In regard to noise, thermal imagery also has different characteristics compared to visual imagery. Typically, thermal cameras have lower resolution and a larger percentage of corrupted pixels. In Figure 2 the same frame is captured by thermal camera on the left and normal camera on the right.

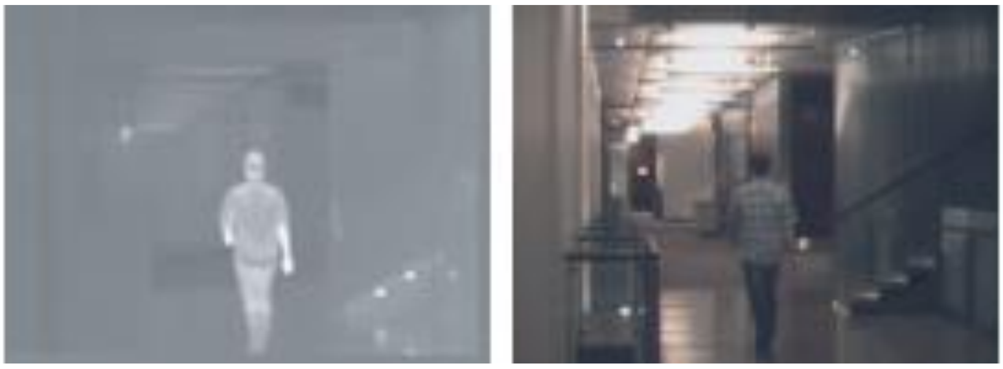


Figure 2. Same scene captured by (a) thermal camera and (b) normal camera. Copied from [4]

An example of a thermal image on the left in Figure 2 might occur to look similar to a grayscale image. Therefore, some research papers argue that same computer vision methods that work well for visual imagery can be successfully applied to thermal imagery. At the same time, other papers, for example [3], suggest that there are several core differences between thermal and visual imagery, hence a specially designed approaches should be used for analyzing thermal images. Therefore, one of the objectives of this project is to empirically determine which argument is true.

2.2 Object Detection

The first essential step of any automated surveillance system is an object detection stage. Historically, the task of identifying objects in an image has been solved via different computer vision (CV) methods. However, recently alternative deep learning methods enabled extensive advancements in every computer vision task including object detection [1].

Object detection is one of the most common computer vision tasks and an integral part of any automated surveillance system. The term of object detection is frequently used interchangeably with another computer vision term of object recognition. Hence before proceeding, it is crucial to appropriately define and discern the terms of image classification, object detection and object recognition [5].

Image classification is the process of taking an image as input and outputting a class label or a probability associated with the class label out of a set of given classes. Object detection usually refers to the process of discovering the presence of an object in an input image, alongside with a localization of that object. An output of an object detection task is a bounding box described in the format of *(**x**, **y**, **w**, **h**)* where ***x*** and ***y*** are coordinates of a bottom left point of a bounding box, ***w*** is the width of the bounding box and ***h*** is the height of a bounding box. There could be many bounding boxes representing different objects of interest within the. Finally, object recognition is assumed to be the process of identifying detected objects. Respectively, an output of an object recognition task is a set of bounding boxes and corresponding classes.

Object recognition can also be considered a classification problem, which aims to classify detected objects into a set of predefined classes [6]. Figure 3 illustrates differences between these computer vision tasks.

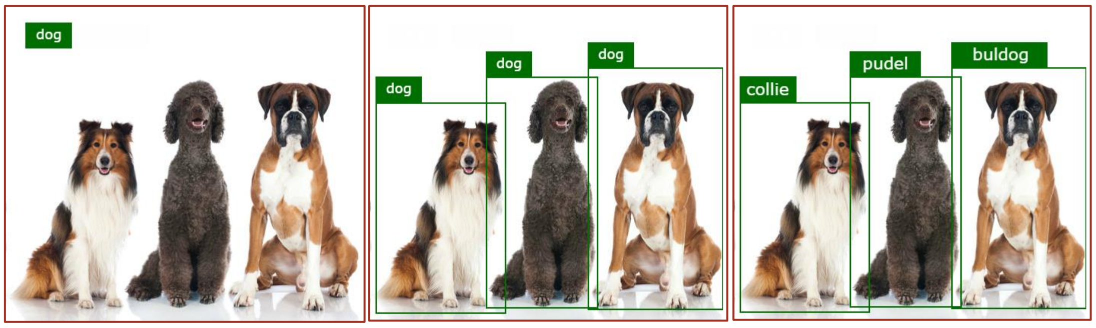


Figure 3. Examples of image classification, object detection and object recognition tasks. (a) - image classification output, (b) - object detection output, (c) - object recognition output.

Depending on the purpose of a specific automated surveillance system, either object detection or object recognition might be needed. Following chapters cover specifically object detection techniques.

3 Object Detection Methods

Detailed understanding of mechanics behind various object detection methods provides an essential theoretical background for further discussion and comparison of traditional and current state-of-the-art object detection algorithms. Moreover, deep understanding of both types of algorithms comes extremely important when it is time to choose the right implementation approach for the project. Therefore, following sections focus on essential mechanisms behind inner workings of both, traditional computer vision and deep learning methods for object detection.

3.1 Traditional Object Detection Methods

Due to the overall academic interest and potential of automated vision systems to increase both the productivity and efficiency of organizations, significant advances have been seen in the field of object-detection systems over last 50 years [7, p. 830]. Traditional approaches to computer vision tasks date back to 1960s when the first applications of pattern recognition systems were developed for character recognition in office automation related tasks [8].

The fundamental idea behind any of these approaches is that any image can be represented as a set of derived features deemed to be relevant to a given task, such as edges, corners, colors or textures. Such a representation of an image is commonly known as a feature vector and the mechanism that converts an input image into a feature vector is known as a feature extractor.

Typical object detection framework consists of several stages. First, feature descriptor is used to represent images as feature vectors. Then these feature vectors are passed as an input to a classification algorithm. Finally, this classifier is applied together with a sliding window to detect and localize objects in an image [7, p.836].

Subsequent sections present a critical overview of the most prominent classical CV approaches to object detection. The main goal of this overview is to get familiar with mechanics of each algorithm, study their downsides and assess feasibility of using traditional CV object detectors for the purpose of this project.

3.1.1 Scale-Invariant Feature Transform (SIFT)

First published by David Lowe in 1999, SIFT method [9] uses local points to identify specific objects among many alternatives. For object detection, SIFT features are extracted from a set of reference images. A new image is matched by individually comparing each feature from this new image to previously extracted ones and calculating Euclidean distance of their feature vectors [10, p. 93]. Figure 2 illustrates a result of local feature extraction process using SIFT method.

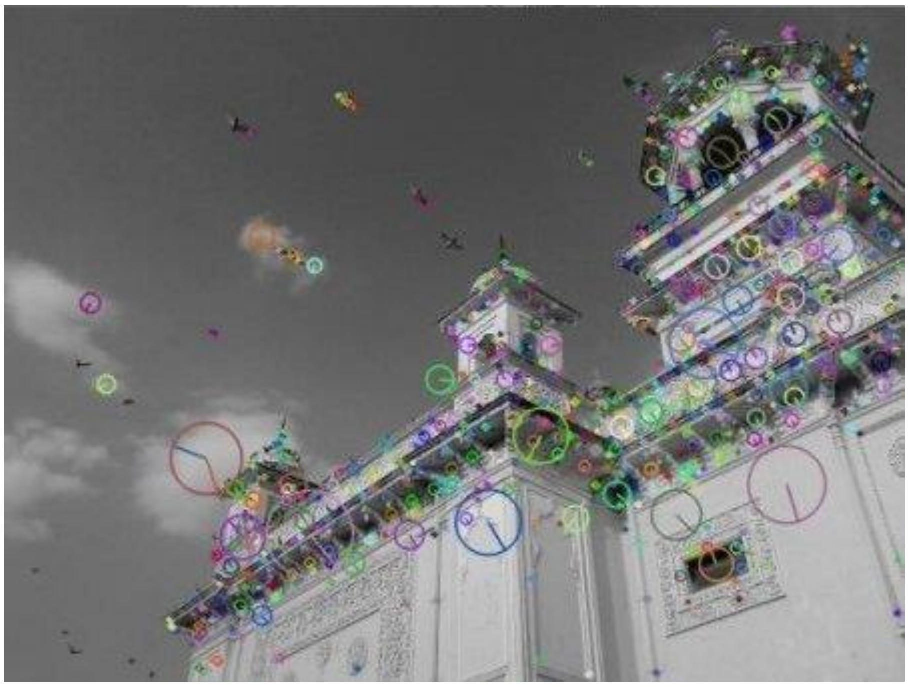


Figure 4. Extracting local features using SIFT method. Copied from [11].

Ability of the SIFT algorithm to extract large numbers of key points from a given image leads to efficient extraction of small objects. Since key points are detected over a range of different scales, small and highly occluded objects can be matched using small local features, whereas large key points allow to achieve high performance for images with sufficient noise [10, p. 106].

However, since SIFT algorithms relies on extraction of numerous key points, it gets sufficiently complex mathematically which results in extensive computational requirements. Moreover, whilst being rotation and scale invariant, this method experiences difficulties when facing lighting changes and blur [12].

3.1.2 Speeded Up Robust Features (SURF)

First presented by Herbert Bay, et al., at the 2006 European Conference on Computer Vision, speeded up robust features is a patented local feature detector and descriptor [13]. Based on the same principles of extracting local features and same algorithmic steps, SURF algorithm achieves considerably lower execution times compared to SIFT, while maintaining the same level of accuracy. Such robustness is achieved due to the quick approximation method called a box blur, which is used to detect points of interest. Blur box algorithm works by computing the average value of all the image values in a given rectangle.

SURF algorithm proved to be an efficient when applied not only to normal images but also for medical imagery [14]. However, as comparative study [15, p. 150] shows, SURF is still prone to make errors on images with different types of affine transformations and blur.

3.1.3 Haar Cascade Classifier

Another technique used in computer vision to perform object detection is called Haar Cascade Classifier. Initially proposed in 2001 by Paul Viola and Michael Jones and therefore also known as Viola-Jones framework, it was the first framework to fulfill requirements of a real-time object detection task.

Original Haar Cascade Classifier consists of four principal stages. First of all, a set of manually determined Haar Features is calculated from the input image. Predominantly, these features help to detect lines and edges. Hence this algorithm is so efficient at its original application of the face detection task. An “integral image” constructed during the second stage allows features to be computed extremely fast [16]. However, most of the calculated features are irrelevant. Therefore, during the next stage, a learning algorithm, called AdaBoost, chooses only features carrying critical visual information about the image [16]. Afterwards, a classifier is trained using only these strong features. Term cascade explains the last stage, when a set of weak classifiers is combined together to form a strong one. This complex classifier allows to rapidly discard background-like regions and concentrate resources on computing more promising object-like regions.

One of the main deficiencies of this method originates from its initial application for face detection. Haar Features describe low-level visual properties of an image such as edges and lines. Therefore, if a target image does not have clear edges and lines, or even if they are slightly modified, an algorithm might not be able to correctly classify such images. Since this project is concentrated on detecting people in relatively low-quality images, Haar Cascade Classifier might not be the best candidate for the task.

3.1.4 Histogram of Oriented Gradients (HOG)

Lastly, one of the most prominent and widely used in traditional computer vision object detection methods is HOG descriptor. This method relies on the use of intensity gradients. First, the whole image is divided into small connected blocks, then a histogram of gradient directions is compiled for the pixels inside each block. The results of calculating gradients for a given input image are illustrated in Figure 5.

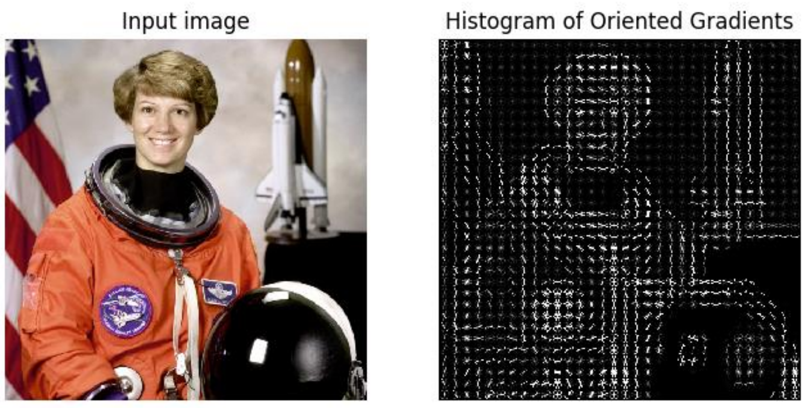


Figure 5. Input and output of HOG descriptor. Copied from [17].

For further accuracy improvement, local contrast normalization is applied across overlapping cells. This approach got widely adopted after Navneet Dalal and Bill Triggs presented their work on pedestrian detection [18] at the Conference on Computer Vision and Pattern Recognition in 2005.

Since HOG descriptor operates on local cells, is has several advantages over other image descriptors. It is invariant to geometric and photometric transformations, except for object orientation. Moreover, local contrast normalization ensures better invariance to changes in illumination, shadowing, and edge contrast [17].

3.2 Deep Learning Object Detection Methods

Merely half a decade ago, computer vision mainly relied on image processing algorithms and methods. Whereas most computer vision tasks might appear easy and be trivially solved even by children, traditional CV algorithm as can be seen from the previous sections, are inherently complex in their nature. The main reason is that there is still no clear understanding of how human vision works. Another facet of the problem is the complexity of the visual world. A robust and accurate computer vision system should work in a variety of different scenes and under a range of changing conditions such as lightning, blurring, noise, etc.

Due to the immense success of deep learning applied to computer vision problems, which started in 2012, object detection methods based on deep learning models became increasingly popular as a default choice in recent years. In 2012 Krizhevsky et al. [19] won a Large Scale Visual Recognition Challenge (ILSVRC) competition by training a large deep convolutional model and achieving significant improvement in accuracy over all other approaches.

Whereas tailored to solve the same computer vision tasks, deep learning models significantly differ from previously described traditional CV algorithms. In the following sections, a brief overview of the most essential fundamentals and concepts of DL for CV will be given in order to provide a basic understanding behind its inner working.

3.2.1 Deep Learning Fundamentals

Deep Learning can be defined as a sub-field of machine learning (ML), aimed at learning a function which can accurately map a set of given inputs to a set of given outputs [20, p. 164]. Applied to an object detection task, a deep learning model tries to map an input image to a set of bounding boxes drawn around each object of interest in that input image. To find such a mapping function, deep learning methods use so-called neural networks.

Main building block of a typical neural network is called a layer. Generally, any neural network consists of an input layer, an arbitrary number of hidden layers and an output layer [20, p. 161]. Each layer itself is composed of nodes or neurons. Depicted below in Figure 6 is a typical deep neural network with k hidden layers.

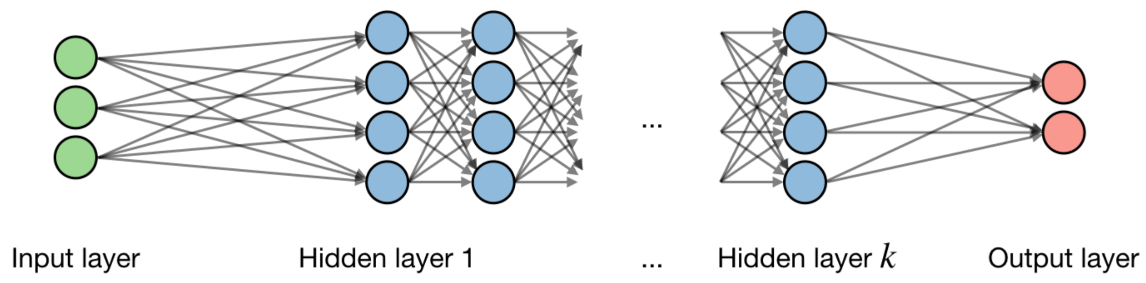


Figure 6. Typical deep neural network architecture. Copied from [21].

Figure 6 also illustrates that neurons in each layer are connected to the neurons in the previous layer as well as to the neurons in the following layer. These connections are called weights.

Interconnected in such a way, neurons allow computation to happen within the network and training signal to propagate through layers. Weights can either amplify or diminish the training signal depending on whether if a current input is helpful for the task or not. Figure 7 provides schematic representation of an artificial neuron.

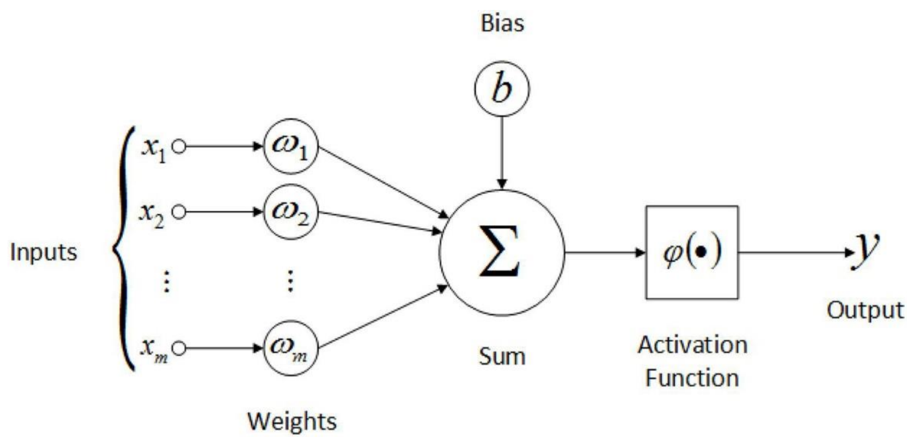


Figure 7. Parts of an artificial neuron. Copied from [22]

Illustrated in Figure 7 is an example of a computation carried by a typical neuron. A neuron takes a set of inputs {***x1***, …, $\boldsymbol{x_m}$} and calculates a product of them and a corresponding set of weights {***w1***, …, $\boldsymbol{w_m}$}. Input-weight products are summed up afterwards and an additional scalar bias term b is added to the sum [23, p.322]. The output is then passed through a non-linear activation function to produce the final output ***y***. Final output of one layer servs as an input to the following layer where similar computation is performed again until the output layer is reached.

The output of the final layer is the same as a prediction made for the task in hand. After prediction was made, it can be compared to the ground-truth, a value which is known to be the right prediction for that particular input. Such comparison of the produced output to the expected output is calculated using a loss function. Most often, cross entropy is used as a loss function for classification tasks, whereas mean squared error is used for regression tasks.

The ultimate goal of the training process is to minimize the loss. This can be done by iterative adjustments of model parameters, i.e. weights and biases. This process is performed with an algorithm known as backpropagation through gradient descent [24, p.69]. Essentially, backpropagation can be viewed as a feedback loop allowing neural network to find the best possible combination of the parameters and therefore achieve the lowest loss. During the training of a neural network, each forward pass includes loss calculation and each backward pass includes calculation of the gradient and then performing backpropagation or integrating the gradient in order to decide how much and in what direction the weights should be changed. This process is repeated until a point of convergence is reached.

Depending on task in hand and given input, different types of neural networks can be used. Historically, when the input is represented as an image, a special type of a neural network called Convolutional Neural Network (CNN) is used.

Originally devised by Yan LeCun in 1989, convolutional neural network is a special type of a deep neural network which is tailored to process grid-like inputs, most often images [20, p. 326]. This name comes from a particular mathematical operation called convolution. Therefore, any neural network which has at least one layer that uses convolution operation instead of typical weight matrix multiplication, can be called a CNN.

Coming from the field of digital signal processing, a convolution operation refers to a combination of two functions which produces a third function as a result. Simply put, it combines two sets of numerical information. In the scope of deep learning, the input is corresponding to the first function and a kernel is corresponding to the second function [20, pp. 327-329]. Essentially, a kernel or as it is often interchangeably called a filter, represents an array of weights or parameters that are being learned by a neural network during the training process. The size of a kernel is usually relatively small, often being defined as 3x3 or 5x5, for example. In the figure below a typical process of applying a convolution operation to the input image is described.

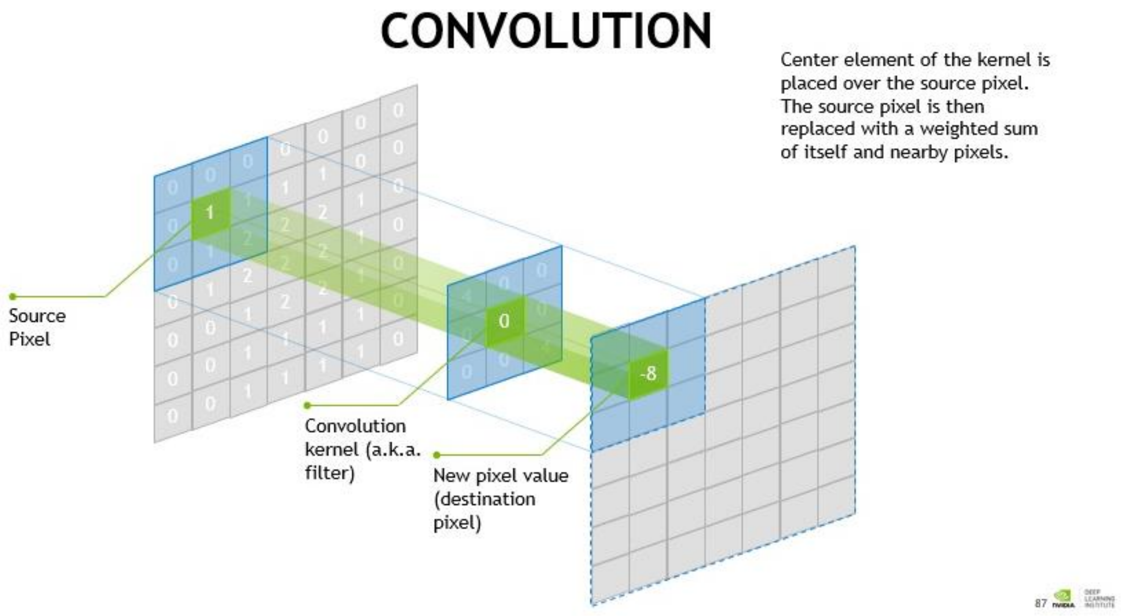

CONVOLUTION
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
Source Pixel
Convolution kernel (a.k.a. filter)
New pixel value (destination pixel)

Figure 8. Convolutional operation applied to the input image. Copied from [25].

It can be seen from the Figure 8, that during the convolution operation, a kernel is sliding or convolving across the input image in both dimensions. At each position a kernel is multiplied element-wise with the corresponding part of the image and the sum of the products is the final output of the convolution operation at that position.

The result of a filter convolving over the whole image is called a feature map [24, p. 124]. Usually a set of multiple feature maps is produced at each level by applying different filters. Although all of the filter values are learned during the training process and never defined manually.

Interestingly, at each layer typical neural network learns different types of features. As a rule, in first layers it learns low-level features such as edges, lines and shapes. The deeper the layer is, the more complex features it learns. Final layers usually represent quite sophisticated features such as textures, specific patterns or elements of the objects Such hierarchical structure allows CNNs to capture complex nature of natural images independent of scale of the objects, their position within the image and different distortions such as blur and noise [24, p. 123].

Additionally, CNNs use pooling layers in between convolutional layers in order to reduce the dimensionality of the intermediate inputs and therefore reduce computational costs and speed up the training process. Other hyperparameters include stride and padding

[24, p.126]. Stride defines the step of the filter. Padding allows feature maps to have the same size of the output after convolution pass as it had before it.

CNNs are suitable for a wide range of CV tasks including object classification, object detection, image segmentation, pose estimation and more. Consecutive sections describe two types of general architectures widely used for object detection task in particular.

3.2.2 One-Stage Object Detectors

As opposed to region-proposal family of models, one stage detectors represent a set of architectures which are trying to solve an object detection task by producing predictions for coordinates of the bounding boxes and probability scores for different classes using only a single forward pass through the network. They generally aim to classify each region of an image either as background or an object. Various positions across an image are considered as a potential object [26]. One-stage models aim to achieve lower inference time but do so by sacrificing accuracy. Most popular one-stage detectors are Single-Shot Detector (SSD) [27] and You Only Look Once (YOLO) [28].

Generally, all one-stage detectors start by dividing an input image into a grid of cells. Each cell has the same task of predicting any bounding boxes whose centers fall within the area of that cell. Predictions for each bounding box consist of x and y coordinates, width and height, and a confidence score. Confidence score is a probability measure between 0 and 1 produced by a classifier, which reflects how likely is that an object is contained within a bounding box. This confidence score is calculated regardless of the class of an object. Finally, a class prediction is produced for each cell independently of other cells.

The main idea that powers one-stage detectors is anchors or default boxes. The concept of an anchor box was first introduced in [29]. According to the authors, anchors are essentially a set of predefined bounding boxes of carefully selected sizes and aspect ratios distributed across the image. Hence a model should decide which subset of anchor boxes to use and later adjust their coordinates and offset to make a final prediction. Below in Figure 9 is a high-level illustration of how a YOLO model splits an image into a grid of cells, initializes a set of default boxes and makes final predictions.

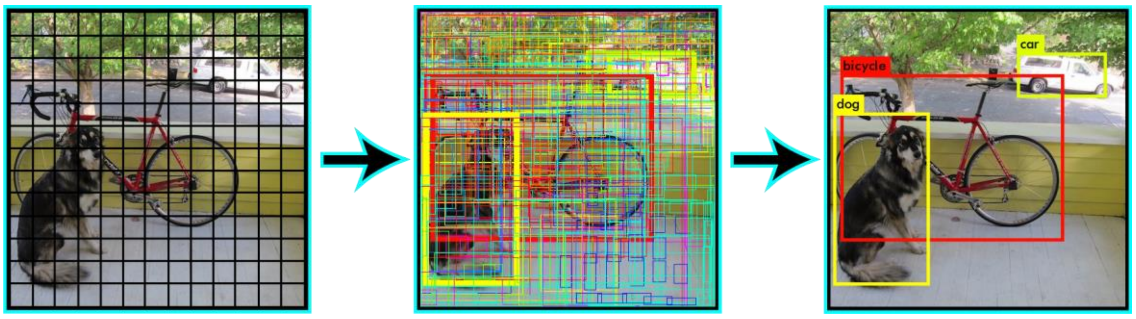


Figure 9. (a) Image split into a grid (b) A set of all default boxes (c) Final predictions. Copied and modified from [28].

One-stage detectors represent an end-to-end type of a network and make predictions using a grid without carrying out any intermediate tasks. Such architectures typically produce faster and simpler models compared to two-stage object detectors. These characteristics make one-stage detectors a plausible candidate for this project.

3.2.3 Two-Stage Object Detectors

The main idea upon which all two-stage object detectors are built is composed of two essential steps [26]. First, a technique called selective search is used to identify a manageable set of regions within the image which might contain different objects. Such object region candidates are commonly known as Regions of Interest (RoI). These RoIs usually produced in various sizes. Second step of the process is to extract specific features from each region using a CNN model. Feature extraction is applied independently to each region for further classification. Additionally, a regression model could be integrated into two-stage architectures to refine bounding boxes predicted by a region proposal network (RPN).

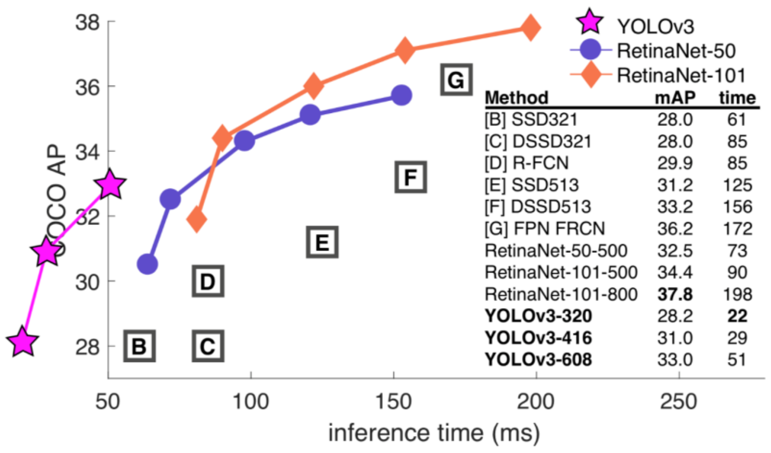
R-CNN is the first widely adopted two-stage detector which significantly improved upon previous models was proposed by Girshick et al. in 2014 [30]. Despite being relatively inefficient due to a high amount of redundant calculations, this architecture served as a basis for a whole family of region proposal object detectors.

Consequently, same group of authors merely a year later proposed Fast R-CNN architecture. Improved architecture addresses main limitations of R-CNN model by merging three models into a single framework [31]. Unified model increased number of parameters shared across the layers, therefore enabling lower training and inference time while achieving better prediction scores than original R-CNN model.

Next iteration of the R-CNN family models, Faster R-CNN, achieved state-of-the-art results by combining RPN into the main CNN model and further increasing the ratio of shared parameters [29]. As a result, nowadays multiple variations of a Faster R-CNN based architectures exist. [26]

Finally, the latest most influential two-stage detector is Mask R-CNN. First presented in 2017 by He et al. [32], this approach augments Faster R-CNN with a separate branch for predicting pixel-level masks of the objects.

Overall, two-stage detectors tend to produce more accurate results with regards to both classification and bounding box localization. As can be seen in Figure 10, while for example YOLO3 has the fastest prediction times, its accuracy is below average compared to other detectors.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 10. Performance comparison between various deep learning object detectors. Copied from [33].

However, whereas two-stage detectors outperform one-stage detectors, it usually comes at a significant computational price. Whereas, as it can be seen from the Figure 10, one-stage RetinaNet model represents a balanced model which is able to provide high accuracy without requiring extensive computational resources. These are important factors that should be considered when choosing a model for the project.

3.3 Deep Learning vs. Traditional Computer Vision Methods

Previous sections provided detailed description of how both traditional CV methods and DL based methods for object detection work. When comparing both approaches, it can be seen that the main and the most important difference is how each method arrives from the input to the output. As illustrated in Figure 11, traditional CV methods heavily rely on a feature extraction phase. During this phase, a set of hand-crafted predefined features is extracted from an input image. Alternatively, DL based methods allow features to be learned in an end-to-end fashion.

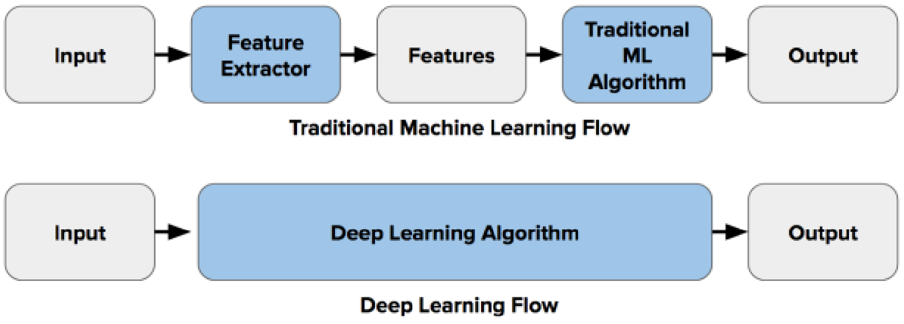


Figure 11. Conventional Machine Learning vs Deep Learning Flow. Copied from [34]

Such end-to-end approach is the main differentiating characteristic of any DL based method. It allows to completely skip the tedious phase of feature engineering. Instead, machine learning practitioner only needs to provide a predefined model with inputs (images) and outputs (a set of class labels and bounding box coordinates). This makes development process faster and eliminates possible errors in an overall pipeline. Moreover, no extended domain knowledge is needed to apply DL based methods of object detection.

However, DL methods have other advantages over traditional CV methods. Most noticeably, CNNs in particular, have proven to have superior performance on a wide range of CV tasks and have also surpassed human-level performance. Typically, the more challenging the task in hand, the bigger would be the difference between performance of a traditional CV model and DL model.

Furthermore, DL methods allows to continuously improve model performance by increasing the size of a training set without hitting diminishing returns as traditional CV models eventually would. Even though such a requirement to have lots of training images could have been seen as a downside a few decades ago, nowadays there is lots of data for almost any vision task, especially if we consider the case of video surveillance. However, labeling such huge datasets is clearly a downside and can turn out to be both expensive and time-consuming.

On top of that, deep learning models for CV have shown to be more robust at handling images containing different transformations including blur, noise and scaling. This essentially means that if a model has been trained on a set of images in one environment, it could be used to analyze images obtained from similar environments. In practice, this allows to use same model to perform human detection across different locations around the campus, be it cafeteria, meeting room or a lobby. technically speaking, deep learning models provided higher generalization. Not only it allows to spend less resources on development, but also facilitates easier scaling.

Finally, due to heavy computational costs, typical deep learning models require special hardware such as GPU video cards to be used for training acceleration. When trained on a standard CPU, such models can take days or even weeks to train. However, nowadays GPU cards are rather inexpensive. Additionally, free cloud resources like Google Colab can be used as an alternative for purchasing actual hardware.

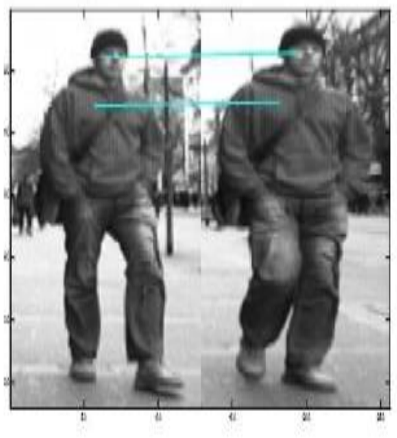
To conclude, whereas deep learning methods for CV tasks have some minor downsides compared to traditional CV methods, their upsides drastically overweight them. Therefore, nowadays deep learning methods should be reviewed and considered alongside traditional methods. Next sections consider in detail two main types of object detection frameworks: one- and two-stage detectors. Knowing principal differences between them will later allow to choose the best suitable architecture for the project.

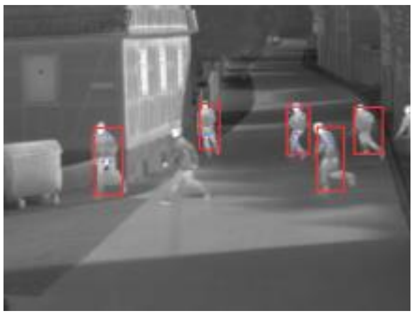
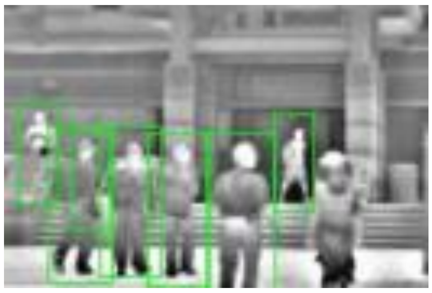
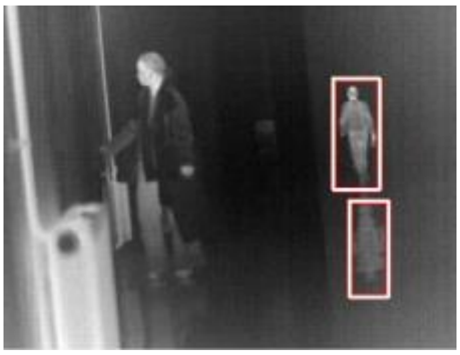
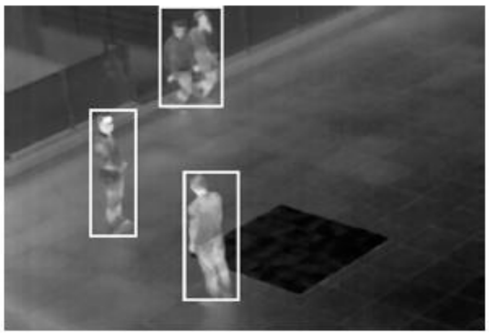
3.4 Related Work

Previous sub-chapters have covered in detail main object detection methods from both, traditional CV and deep learning. Their strong and weak sides have been highlighted. However, before one can make an educated decision about which model to use for the project, it would be useful to get a brief overview of which methods prevail in practice. Therefore, in this section, a survey on recent papers on object detection is presented.

Object detection is a popular CV tasks and hence a popular research topic. To narrow down the amount of reviewed papers, three main criteria are used. Firstly, a paper is required to be describing work on object detection, or other CV task where object detection is a downstream task. Secondly, the paper should be working with thermal images. And finally, the task should include humans as one of the possible classes. The more criteria a paper meets, the better. However, not all the reviewed papers meet all three criteria at once.

In Table 1, a summary of all reviewed papers is given. Each entry includes paper's title and a reference link, year of publication, main method used to perform object detection and an example. Analyzing this information helps to understand current trends in human detection in thermal imagery and their evolution over time. Sample images allow to visually compare tasks and identify which images are most similar to the images used in this project.

	Title, authors	Year	Method	Example
1	Tracking of Humans and Estimation of Body/Head Orientation from Top-view Single Camera for Visual Focus of Attention Analysis [35]	2009	SIFT	
2	Human Detection Using SURF and SIFT Feature Extraction Methods in Different Color Spaces [36]	2014	SIFT, SURF	

3	Feature based person detection beyond the visible spectrum [37]	2009	SURF	
4	Pedestrian Detection in Infrared Images based on Local Shape Features [38]	2007	HOG	
5	A Real Time Human Detection System Based on Far Infrared Vision [39]	2008	Gaussian background model	
6	Human Detection Based on the Generation of a Background Image by Using a Far-Infrared Light Camera [40]	2015	Background subtraction	

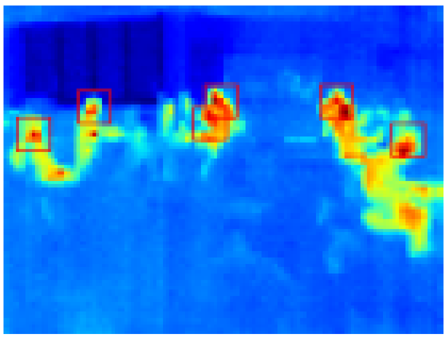
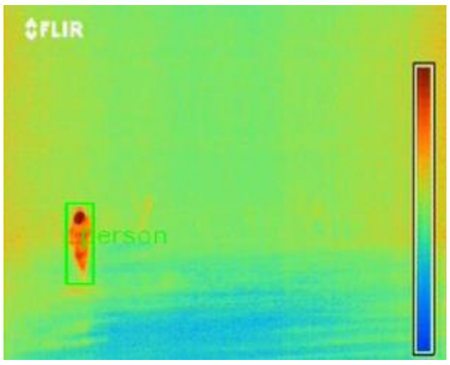
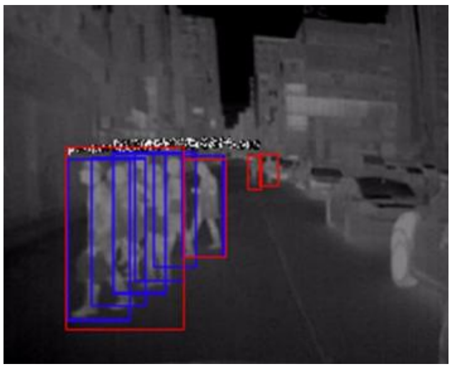
7	Person Detection in Thermal Images using Deep Learning [41]	2018	Convolutional autoencoder	
8	Thermal Image-Based CNN's for Ultra-Low Power People Recognition [42]	2018	CNN	
9	Human detection in thermal imaging using YOLO [43]	2019	YOLO	
10	Human Detection in a Sequence of Thermal Images using Deep Learning [44]	2019	Temporal CNN	

Table 1. Summary of various research papers on human detection in normal and thermal imagery.

Upon critical review of the abovementioned papers, two main factors have been discovered. First of all, six papers out of ten used traditional CV methods as SIFT, SURF and HOG to identify people in both normal and thermal images. The other four papers opted towards different convolutional models. However, an important observation is that most recent papers, published in 2018-2019, exclusively use deep learning methods, whereas up until 2015 only traditional CV methods were used. This highlights a significant trend of choosing deep learning models both for research and development.

The reason for such a paradigm shift can be derived both from section 2.2.3.5 where main differences between traditional CV and deep learning methods are described, and from analyzing sample images in Table 1. For instance, in a sample image from [37] one can see a missed object right in the center of the image, and another one at the right side of the image. Whereas first error type is similar for both types of object detection, the second person is not identified due to the fact that its silhouette is only partially visible. This prevents traditional CV model from making a correct prediction. However, when trained properly, deep learning models can successfully identify partially occluded objects or objects exceeding the limits of the image.

Consequently, in a sample image from [39], another two error types typical for object detection in thermal imagery are present. Firstly, a reflection is identified as a person. This kind of error can be avoided by a deep learning model, if during the training no reflections are labeled as actual humans. Secondly, if a person present in a thermal image has just appeared to come from outside during the cold season and wears a coat, the heat radiated by the body is blocked by the coat and therefore person's silhouette looks different and cannot be picked by traditional CV methods, especially those methods that rely on background extraction.

Lastly, in [40] sample image two people standing close to each other share a single bounding box. Even though they are correctly identified, traditional CV method is not able to separate them as two individuals. On opposite, in sample image from [44], which uses temporal CNN, a whole group of multiple overlapping bounding boxes is correctly identified.

To conclude, it can be suggested that deep learning approaches to object detection became feasible and prevalent during last couple of years. This can be explained by superior performance that CNNs show compared to traditional CV methods. Considering these findings, a deep learning object detector has been chosen for this project. Following chapters provide additional details on methods, implementation details and discuss results that have been achieved.

4 Methods and Tools

In chapter 2 it has been established that DL approaches to solving CV problems might be faster and easier to develop, compared to traditional CV methods, all while achieving better accuracy. Therefore, this and following chapters are set do describe in detail a practical experiment which main goal was to demonstrate the eligibility of that statement.

Whereas starting point and final goal for both, software projects and ML projects, are the same, an internal life cycle differs a lot. Generally, ML projects include more uncertainty and variability than traditional software processes. Additionally, they use a different set of programming tools and test solutions in a different way. Following sections go through each step of a development life cycle for a typical ML project with respect to the experiment conducted for this work.

4.1 Machine Learning Development Life Cycle

Traditionally ML projects are more iterative and experimental by their nature compared to software projects. Therefore, one should be prepared to try out a range of various ideas, approaches and models before arriving at a satisfying solution. Specific number of steps included in a typical ML workflow may slightly vary from source to source. In Figure 12, a set of most common steps for ML project is depicted.

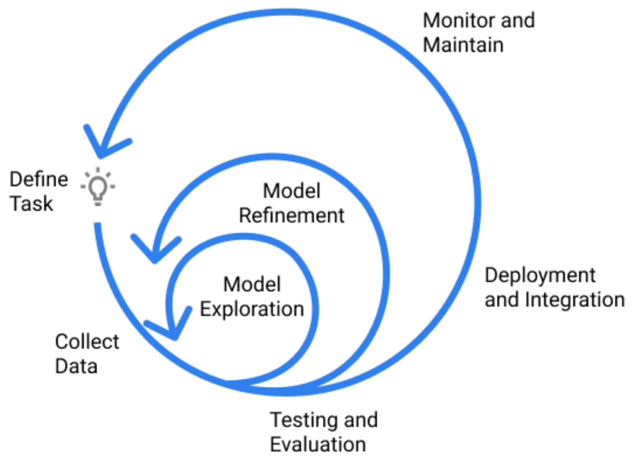


Figure 12. Typical workflow of a ML project. Copied from [45].

Starting from task definition and ending with continuous monitoring of a deployed model, most of the steps presented in Figure 12 might be repeated several times. Roughly half of the steps are aimed at preparing the final model, whereas second half is concerned with deployment and maintenance of the developed model. Tasks related to model deployment, a vast topic of its own, are outside the scope of the project. Therefore, following sections focus on data preparation and model implementation.

4.2 Task Definition

To reinstate previously discussed motivation for the project from business perspective, the main goal of the practical part of the project is to develop an object detection model. This model must accurately identify humans in thermal images and provide coordinates of a bounding box for each identified human.

Correctly chosen metric often defines the success of an ML project and allows to bring actual business value. Having a single metric defined at the very beginning of the project, as Andrew Ng suggests in [46, p. 20], allows ML practitioners to rapidly iterate over new ideas and evaluate them according to that single metric, making steady progress towards the main goal.

Evaluation of a trained object detector is not a trivial task because it requires to simultaneously account for both classification and regression parts of the task. For this project, mean average precision (mAP) is chosen to be a single optimizing metric. It is a score metric commonly accepted in object detection competitions such as ImageNet, PASCAL VOC and COCO.

For each predicted bounding box there are three possible outcomes: true positive (TP) in case an object is correctly classified and localized, false positive (FP) in case an object is incorrectly classified or localized and false negative (FN) in case if an existing object is not identified. The last case of a true negative (TN) prediction, correctly not predicting an object where it does not exist, is irrelevant within the scope of an object detection task.

Knowing a number of true positive, false positive and false negative predictions across the dataset, one can calculate ***precision*** and ***recall***. Formulas 1 and 2 show how each of the metrics is calculated.

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

Formula 1. Formula for calculating precision. Copied from [47].

$$Recall = \frac{TP}{TP + FN}$$

Formula 2. Formula for calculating recall. Copied from [47.]

In Formula 1, precision is a measure of how well a model identifies relevant objects. In Formula 2, recall is a measure of model’s ability to find all the relevant objects.

Consequently, mAP represents a product of both precision and recall of the predicted bounding boxes. By combining detections from all images, a precision/recall curve can be drawn [60]. Inspection of precision-recall curves is highly recommended to get the full picture of both metrics when evaluating a model. An example of such a curve is depicted in Figure 13.

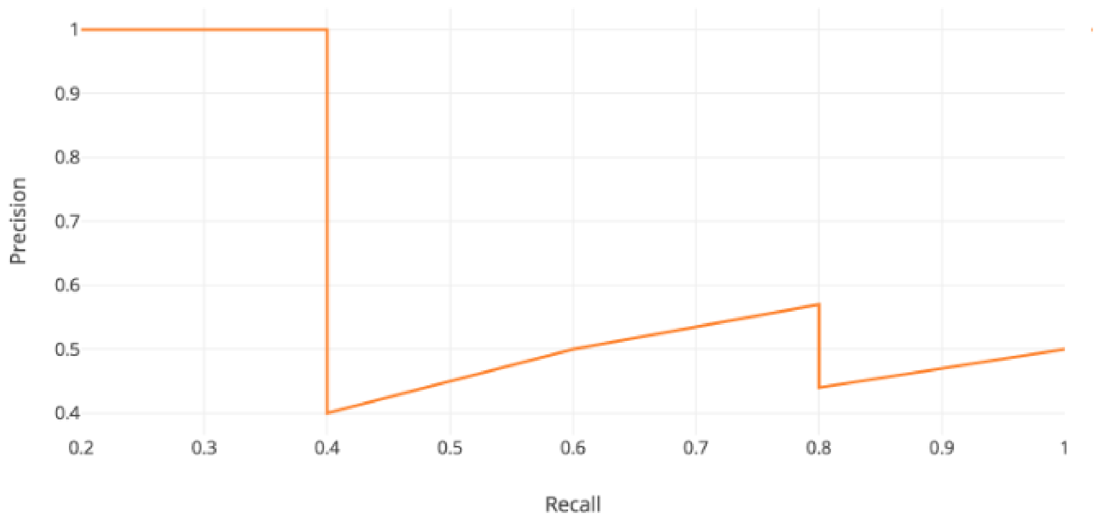


Figure 13. Precision-recall curve. Copied from [61].

The area under the curve, as shown in Figure 13, is used to calculate average precision (AP) for each class in the dataset. Therefore, mean average precision is an average of AP scores across all classes. The mAP score can take values in a range between 0 and 1, where a score of 1 is achieved by a prefect model. For this project there is a single class, “person”, therefore AP must be calculated only once. The goal is to achieve a mAP score of 0.95 or higher.

Apart from having an optimizing metric, it is often advisable to have a satisficing metric [46, p. 22]. In case of this project, it is not only required to achieve a certain performance, which is measured with mAP score, but also achieve sensible inference time. The most accurate model would be useless if generating predictions for a single image takes dozens of minutes. For this project, satisficing inference time is defined to be 10 seconds on a standard CPU or GPU machine. This way all locations can be processed every minute with some buffer time left for any possible overheads. This way it would be possible to provide updates on a space occupancy rate at a granularity of a minute.

Establishing optimizing and satisficing metrics allows to compare different models and their variations against each other and eventually choose the one which is most suitable for the deployment. Ultimately, developing a model which complies with both metrics, should be interpreted as a successful fulfilment of the main goal for the practical part of the project.

3.3 Data Collection

Popular computer science principle states: “Garbage in, garbage out” [50]. From machine learning perspective, this means that the quality of the output produced by a model depends on the quality of the inputs. When considering CV tasks, several principles should be followed to ensure the quality of the collected dataset.

According to [46, p. 15], the most important condition for a ML dataset is to represent as closely as possible examples which a model must predict well on when deployed. Hence, images from the dataset should be ideally coming from the same distribution as images fed into the model during the inference stage.

Secondly, to ensure that model generalizes well to previously unseen data, ML dataset should represent as wide a range of possible variations in images as possible. One of the goals for this project is to create such a model that could be deployed across different locations on the campus: meeting rooms, corridors, lobbies and other common areas. Therefore, a well-designed dataset should include images collected across all such areas.

Last of all, since another goal of the project is to compare performance of traditional and DL approaches to CV tasks, the dataset should include cases which were found in Chapter 2 to be challenging for traditional CV object-detection models. Such cases include presence of small objects, objects being partially outside of image borders, reflections and occluding objects. Consequently, it is crucial for the dataset to include such images in order to access performance of the final model in all the described categories.

Given above constraints, a training dataset of 2000 images was collected across four different locations at Nokia Campus, Espoo. Another 400 images comprise a test dataset. All images were gathered using a thermal camera provided by LEVITEZER Oy. Produced images are grayscale and have resolution of 120x160 pixels. Table 2 provides aggregate statistics of the training dataset sorted by location.

Location	Average people per image	Average object size, pixel	Reflections	Occlu-sions	Outside of borders
Location 1	1.14	3767	No	4.5%	9.5%
Location 2	1.0	1103	Yes	4.5%	23.3%
Location 3	1.26	903	Yes	20%	13.3%
Location 4	3.31	1169	Yes	90%	10.9%

Table 2. Per location statistics of the training dataset.

Furthermore, statistics for an average size of bounding boxes, percentage of images that include reflections, occlusions and bounding boxes outside of image borders are given in the same table.

Based on provided details of the dataset, it is reasonable to conclude that such dataset should allow to accurately access performance of a model with respect to generalization ability and robustness against known challenges in object detection. Since train and test sets were both obtained by shuffling and splitting the original dataset of 2400 images (train set – 80%, test set – 20%), they both have similar distribution and therefore fully represent the original dataset.

4.4 Model Choice

Two main types of DL-based object detection architectures have been discussed and compared in Sections 2.2.3.3. and 2.2.3.4. Whereas two-stage detectors traditionally achieve slightly better accuracy than their one-stage counterparts, this superiority comes at a cost of additional computational complexity. Given iterative nature of ML projects, where same model or its modifications must be trained multiple times, both computational costs and training times accumulate. Therefore, following Occam's razor principle widely adapted by the field of data science, one should always favour the least complex solution.

Furthermore, recently a question of environmental impact of using DL models has been raised by several sources including [51]. A well-known trend pictured in Figure 10 suggests that the best performing models usually also have the largest number of parameters. Which subsequently means greater computational cost and higher energy consumption. Most importantly, big models consume more energy not only during the training but also later, when a model is used for inference. Therefore, additional costs scale accordingly to increase in number of inferences.

While it would be possible to use highly accurate two-stage detectors for the purpose of this project, a decision was made to avoid architectures with a significant overhead and favor models that exemplify sustainable approaches to developing DL solutions. Such models aim to provide state-of-the-art results by leveraging smart designer choices.

Ethical and sustainable development is one of the core values of Nokia [52]. Therefore, a one-stage architecture called RetinaNet was chosen to be used in this project. Next section presents a more detailed overview of its inner working.

4.5 RetinaNet

RetinaNet [53] is an example of an architecturally simple one-stage detector which has demonstrated both superior accuracy and admissible inference time in comparison with other one-stage and two-stage object detectors. Besides, an opensource implementation of the architecture is available on Github [54] under Apache license.

Although, RetinaNet is a fairly simple architecture, in order to be able to use it appropriately and be able to adjust and tune its parameters to fit a custom dataset, a profound understanding of the design is required.

Fundamentally, RetinaNet is composed of the following parts:

- a Feature Pyramid Network built on top of a backbone network;
- a classification subnetwork which performs object classification task manipulating on the output of the backbone network;
- a regression subnetwork which performs bounding box regression task manipulating the output of the backbone network.

Figure 12 shows how different parts of the RetinaNet network are connected to each other and what their corresponding dimensions are.

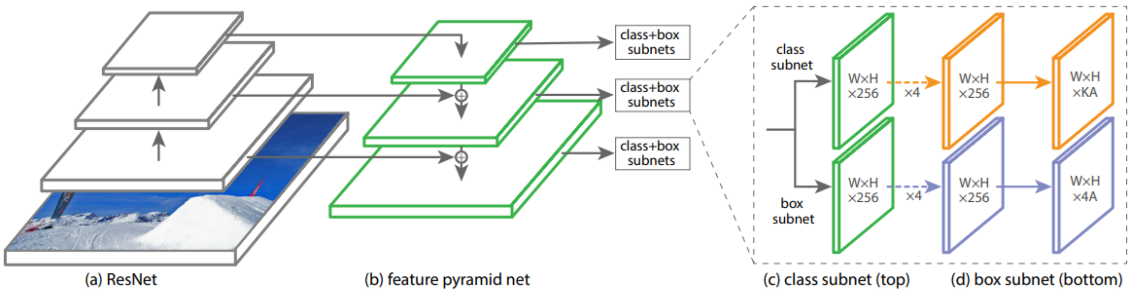


Figure 12. Design of RetinaNet. Copied from [53].

Depicted in Figure 12 is a high-level structure of RetinaNet. First part of the RetinaNet architecture utilizes slightly modified Feature Pyramid Network (FPN) which was previously introduced by [55]. In a default configuration of RetinaNet FPN augments a standard image classification network, for example ResNet [56].

Implemented as a fully convolutional network, the backbone network of RetinaNet takes an arbitrary sized image and produces feature maps of different sizes at different levels of the FPN. At each level of the pyramid, objects of different sizes can be efficiently detected [26]. Higher levels of the pyramid produce feature maps containing grid cells spanning over larger regions of the input image, whereas lower levels of the pyramid produce feature maps containing grid cells covering smaller regions of the input image. As a result, higher levels are better at detecting large objects and lower levels are more suitable for detecting small objects. This makes RetinaNet scale invariant.

Classification subnetwork represents a fully convolutional network (FCN) connected to FPN at each level. Four convolutional layers of 256 filters with kernel size of 3x3 and ReLU activation function are followed by another convolutional layer which has $K * A$ filters [53]. This final convolutional layer is using a sigmoid activation function to produce probability scores.

The shape of the feature map in the output layer is $\mathrm{WxHxK} * \mathrm{A}$, where *W* and *H* correspond to the width and height of the input feature map, *K* is the number of classes and *A* is the number of anchor boxes.

Regression subnetwork is attached to the FPN in the same manner and in parallel to the classification subnetwork. Moreover, these two subnetworks share an identical design with the exception of the last convolutional layer, which in regression subnetwork has 4*A filters [53].

Consequently, the shape of the feature map in the output layer of the regression subnetwork is $\mathrm{WxHx4} * \mathrm{A}$, where 4 corresponds to the four coordinates of the bounding box. Finally, the last convolutional layer uses linear activation function to produce continuous values of the bounding box.

Neural networks trained with stochastic gradient descent always require an appropriate loss function to be chosen during the design and configuration stage. This poses a challenging problem because the loss function should accurately capture the characteristics of the problem and be motivated by qualities that are important to the task.

RetinaNet loss function is composed of two terms [57]:

- first term for localization ($Lloc$)
- second term for classification ($Lcls$)

Combining two parts gives the final formula which can be written as: $L = \lambda Lloc + Lcls$, where λ is a coefficient that balances localization and classification losses.

Specifically, RetinaNet introduces a variation of the classification loss called the focal loss, which is one of the most innovative parts of the model design. Authors of the original paper mention that the practical issue which hurts performance of most object detectors

the most is a class imbalance [53]. Classification imbalance is greatly based on the fact that an extensive part of the locations in any given image can be effortlessly classified as a background and thus does not represent any useful training signal.
In order to address the class imbalance issue, a focusing parameter is introduced into the focal loss formula. This parameter allows to bring down the contribution of the easily classified regions and force the network to put additional resources into classifying hard regions.

In order to calculate the loss for training, predictions produced by both classification and regression subnetworks should be compared with the ground-truth bounding boxes. Since there are several predicted bounding boxes and several ground-truth bounding boxes, it is necessary to understand mechanisms of matching predictions with ground-truths.

According to RetinaNet logic, a ground-truth box is considered to be a match with a candidate bounding box if their intersection-over-union (IoU) is higher than 0.5 [53]. However, if IoU between a candidate anchor box and a ground-truth bounding box is below 0.4, such anchor box is considered to be a background and have no match among ground-truths. Figure 13 illustrates how IoU is calculated.

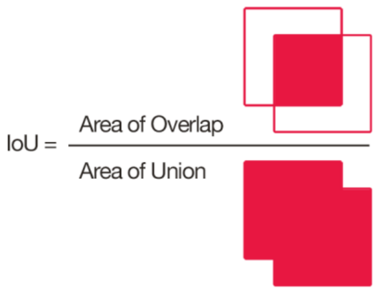


Figure 13. Formula for calculating IoU. Copied from [58]

As shown in Figure 13, IoU is calculated by dividing the area of overlap between predicted bounding box and ground-truth bounding box by a union of these areas. If an anchor box predicts an object instead of a background and vice versa, it is penalized by the loss function. Additionally, in situations when IoU lies between 0.4 and 0.5, an anchor box is recognized to have no match. However, unlike in previous case, no penalty is produced by the loss function.

After a network has been trained up to the desired accuracy levels, it can be used for the inference. During the inference, given an input image, a trained network predicts what objects are present in an image and where are they located.

In order for RetinaNet to generate predictions, at most a 1,000 anchor boxes with the highest probability scores are selected at each FPN level. At this point an object in an image can have multiple corresponding anchor boxes. A non-maximum-suppression (NMS) algorithm is used to select an anchor boxes with the highest predicted probability score for each class independently of others [53].

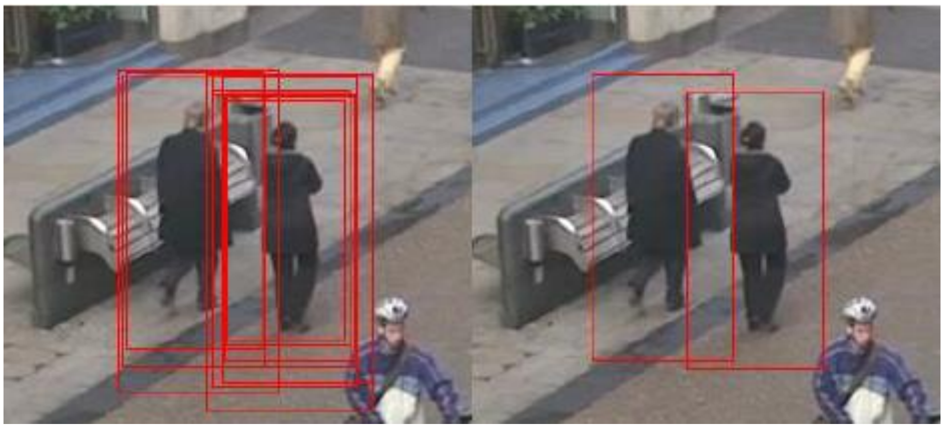


Figure 14. An example of refining anchor boxes with NMS algorithm. Copied from [59]

In the Figure 14, a process of applying NMS to a set of candidate anchor boxes is illustrated. After applying NMS, any overlapping anchor boxes are removed, and previously existing redundancy is eliminated.

Ultimately, the regression subnetwork produces center coordinates and offsets for each of the remaining anchor boxes. These coordinates are used to refine anchors and get final bounding box predictions for each object.

4.6 Software tools

Whereas nowadays there is a wide range of programming languages allowing to develop ML solutions including R, C++, Java, Julia, Scala and many more, the most commonly used language is Python. Being a high-level, general purpose programming language, Python offers ML practitioners an abundance of simple yet powerful tools to tackle various problems. Therefore, Python is chosen to be used throughout the project.

The central most tool for any DL project is a DL framework, which is essentially a high-level interface allowing to access abstractions of different algorithms. By using such a framework, one can develop complex DL models by combining common pre-built and optimized components without delving into implementation details of underlying algorithms.

As of today, there is a plenty of different DL frameworks to choose from. For instance, Google's TensorFlow, arguably the most prominent DL framework, provides the most extensive set of tools for productizing DL models. However, at the same time, it requires practitioners to work with low levels of abstraction and subsequently write more extensive code and conduct vigorous testing. Another popular DL framework, PyTorch from Facebook, is widely used in academia, since it enables researches to implement highly customizable modules.

However, another DL framework, Keras [60], is chosen for this project. Keras was first introduced in 2015 by François Chollet, a world-known expert in DL. Being a high-level API, Keras runs on top of other backends, including TensorFlow. Whilst this results in a less configurable and flexible environment, it simultaneously provides best prototyping capabilities, allowing practitioners to swiftly implement their ideas by writing concise and readable code. Among other advantages, Keras provides seamless support for using both CPU and GPU, which becomes an important consideration, since CV tasks usually require extensive computational power and rely on GPUs for acceleration.

5 Implementation

Due to a highly iterative nature of a ML development cycle, it becomes crucial to establish a reproducible and interactive pipeline. This allows to promptly perform training of a model, evaluate it and perform error analysis. Based on evaluation results and error analysis, one can determine how to adjust hyperparameters, data pre-processing or a training procedure in order to eliminate the most significant source of error. Afterwards, an experiment is repeated until achieving both satisfying and optimizing metrics.

For this project, an experiment pipeline has been developed on Google Colaboratory or 'Colab' for short. platform [61]. It is a cloud based Jupyter environment available for free via browser. In Google Colab users can write and execute custom Python code. More importantly, no setup is required and free computational resources including GPU are provided on a limited basis. Such capabilities allow to minimize development efforts and facilitate rapid prototyping. Therefore, all experiments for this project have been carried out in Google Colaboratory.

First, all required utility libraries such as ***pandas***, ***OpenCV***, ***NumPy***, ***urllib*** and **os** are imported. They provide functionality to download and read images, calculate evaluation metrics and measure execution times.

Given a training dataset which resides in Google Drive, it can be downloaded and used in Google Colab as shown in Listing 1.

```
drive_url = 'https://drive.google.com/uc?export=download&id=' +
DATASET_DRIVEID
file_name = DATASET_DRIVEID + '.zip'
urllib.request.urlretrieve(drive_url, file_name)
```

Listing 1. Downloading training dataset to Google Colab.

Folder with training images should also include a .csv file with labels for each image submitted in Pascal VOC format. Each line in an annotation file describes a single bounding box and includes path to the image, top left *x* coordinate, top left *y* coordinate, bottom right *x* coordinate, bottom right *y* coordinate and a label assigned to that bounding box. Additionally, another csv file with class mapping needs to be provided. For this case, there is only one class – person.

After a training dataset has been provided, a training procedure can be initiated by invocating a script from a pre-installed ***keras_retinanet*** library. A corresponding command line command is shown in Listing 2.

```
!keras_retinanet/bin/train.py --freeze-backbone
--random-transform --weights {model} --batch-size {bs}
--steps {steps} --epochs {n_epochs} --tensorboard-dir {log_dir}
csv {annotations_path} {classes_path}
```

Listing 2. Initializing training with specific arguments.

In Listing 2, apart from the command itself, a list of additional arguments is specified. First argument, `--freeze-backbone`, is used to freeze the weights of a backbone network. This approach, known as ***transfer learning***, is often used in situations when the dataset at hand is smaller than the dataset on which the backbone network has been trained. Freezing pre-trained weights helps to avoid overfitting and decreases training time, since only weights in last layers are adjusted during the training.

Next argument, `--random-transform`, enables ***data augmentation***. Data augmentation is another method of mitigating effects of using a small training dataset. Before an image is fed to the network, a set of random transformations is applied to the image, producing a set of slightly modified images and consequently increasing the size of the original training dataset. For this project, a set of possible transformations includes rotation, transition, shear, scaling and horizontal flipping. Additionally, visual transformations such as change in contrast, brightness and saturation are randomly applied to each input image.

Following `–weights` argument specifies a path to the pre-trained weights. If omitted, the training is initiated with random weights. For this project, weights of ResNet50 pre-trained on ImageNet dataset are used.

Subsequently, `--batch-size`, `--steps` and `--epochs` arguments specify how many images should be included in a single batch, how many training steps should be performed per epoch and for how many epochs the training should run. Larger batch sizes allow for more precise calculation of the gradient during backpropagation at the expense of a longer training time and possible generalization issues. On the other hand, smaller batch sizes are faster to process but often steer weight updates toward a wrong direction and delaying convergence. Correctly chosen batch size allows to balance both training time and accuracy of a gradient calculation. Additionally, an upper boundary for a batch size is limited by the amount of available memory. Since RetinaNet is a computationally heavy network, a batch size of 8 is chosen. When batch size is determined, `--steps` argument can be calculated by dividing number of all images in the dataset by a batch size. Finally, `--epochs` argument specifies how many full passes over the whole training dataset are made. RetinaNet is known to converge relatively fast, therefore this argument is set to 30.

Afterwards, `--tensorboard-dir` argument is used to specify directory to which TensorBoard training logs are stored. TensorBoard is a set of applications for visualizing and tracking TensorFlow runs. Among other capabilities these tools enable users to inspect relevant metrics such as training and validation loss. During the experiment TensorBoard has been used to monitor loss over time, spot possible training issues such as small learning rate or overfitting and recognize convergence moment.

Last arguments, `csv, {annotations_path}` and `{classes_path},` indicate that a custom dataset is used to train the model and points to the files containing labels and classes. When `csv` argument is passed to the training script, a special parser is invoked to read labels and create an instance of a training generator.
After training has been started, a snapshot of the current model is saved to the disk at the end of every epoch. Simultaneously, TensorBoard outputs updated plots with both classification and regression losses. Training curve flattening out and loss being stable for several epochs indicate that a model has reached minimum and training process is finished.

Finally, after a model has been trained, it must be evaluated on a test dataset. Test dataset consists of images a model hasn't seen during the training. Therefore, it provides an ultimate measure of model's fitness. Evaluation metric is a mAP described in a Section 3.2. Additionally, a visualization function overlays each test image with predicted bounding boxes and probability scores. Figure 15 shows an example of visualizing results of the inference performed for a randomly chosen image.

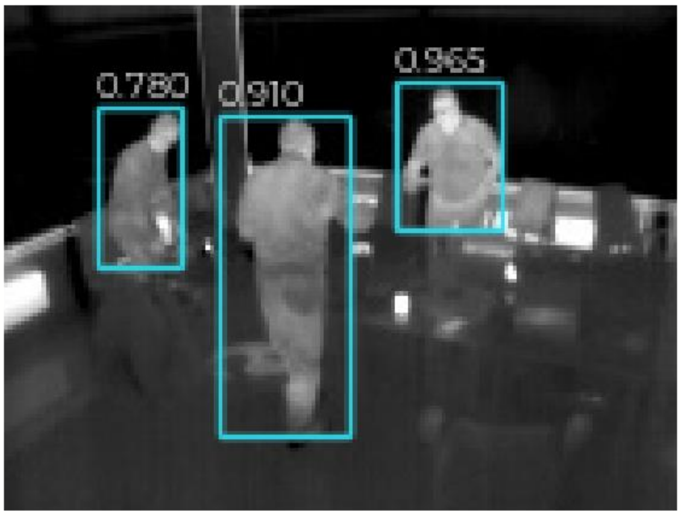


Figure 15. Visualizing predicted bounding boxes.

As seen in Figure 15, three persons have been detected in the input images with corresponding confidence scores of 0.78 (person on the left), 0.910 (person in the middle) and 0.965 (person on the right). Such representation of model's predictions, shown in

Figure 15, allows to rapidly perform a visual error analysis, assess accuracy of localization, and understand what kind of errors a model tends to make.

6 Results

After optimal hyperparameters were identified, an instance of a RetinaNet network was trained on a total of 2000 images. Out of this training dataset, 85% of the images were used to adjust model parameters and calculate training loss. Training loss indicates the progress of the model over time against data it has already seen. Other 15% of the images were used to calculate validation loss of the model at the end of every epoch. Validation loss similarly to the training loss allows to track the progress of the model over time but against previously unseen data.

However, validation loss is calculated by testing the model against previously unseen data. By plotting both learning curves against each other, as illustrated in Figure 16, it is possible to see whether a model is underfitting, overfitting or balanced.

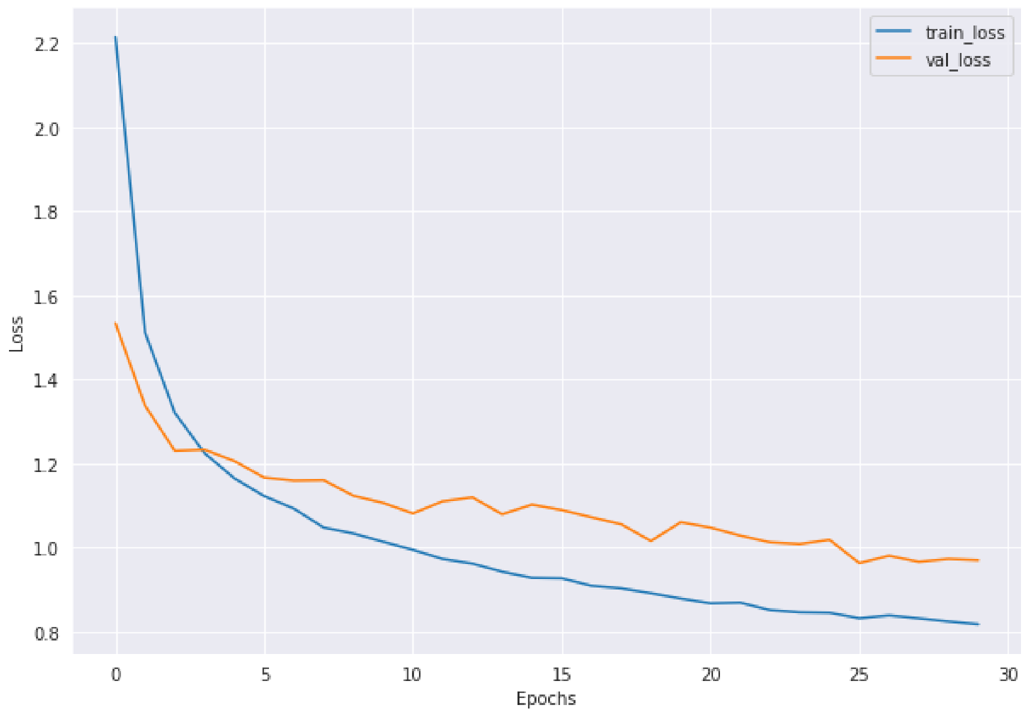


Figure 16. Learning (training and validation) curves plotted over time.

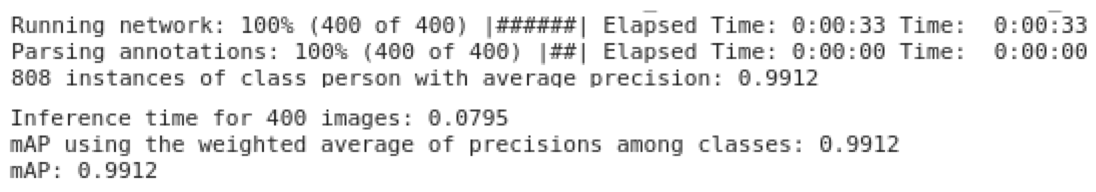
It can be seen in Figure 16 how both, training and validation curves steadily decline as training progresses. Since loss is essentially a measure of error and the goal is to minimize it, the lower the value on the y axis, the better. Furthermore, as it has been previously explained, the loss used in RetinaNet is a sum of classification and localization losses.

It is clear from Figure 16 that both curves reached reasonably low loss values, meaning the model is not underfitting. Moreover, the learning curves stay relatively close to each other and do not start to diverge over time which would indicate overfitting. On contrary, both curves flatten during last epochs, signalling that the model has successfully converged.

To assess model's robustness, ability to generalize and resilience against different environmental changes, a series of carefully tailored test cases were conducted. Their detailed description and corresponding results are presented in the following sections.

6.1 Main test case

To determine an overall model's capacity to correctly detect persons in thermal images, an instance of RetinaNet neural network has been trained according to the training protocol explained in Chapter 4. The model was trained on 2000 images taken across four different location. Another 400 images equally sampled from the same locations were used to test the model. Nearly perfect mAP score of 0.9912 was achieved. Listing 3 shows the output of the testing script.



```
Running network: 100% (400 of 400) |######| Elapsed Time: 0:00:33 Time:  0:00:33
Parsing annotations: 100% (400 of 400) |##| Elapsed Time: 0:00:00 Time:  0:00:00
808 instances of class person with average precision: 0.9912

Inference time for 400 images: 0.0795
mAP using the weighted average of precisions among classes: 0.9912
mAP: 0.9912
```

Listing 3. Testing results for the main test case.

From Listing 3 it can be seen that for this test case, both optimizing and satisficing metrics, established in Section 3.2 are met. Less than 1% of all testing images included either classification or localization error. With accuracy requirement being set to 0.95, proposed model achieved better than expected performance. Figure 17 presents samples of predictions made by the model trained for this experiment.

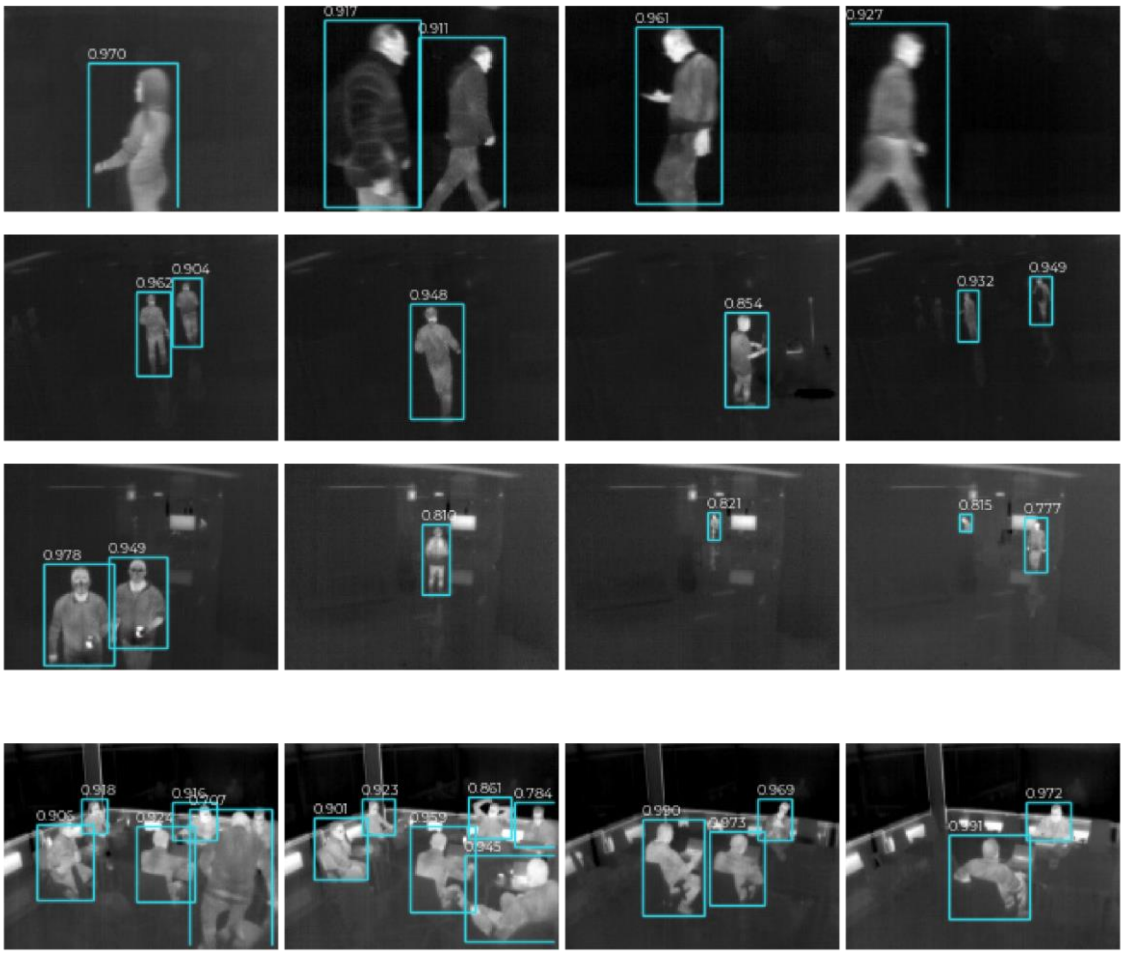


Figure 17. Sample predictions for the main test case.

From top to bottom: location 1, location 2, location 3, location 4.

Out of all the images presented in Figure 17 only first image in the last row contains an error – a single false negative prediction. In the rest of the images all objects are correctly identified and localized. Specifically, fourth image in the second row shows correct omission of reflections. Third image in the third row shows that proposed model is able to handle very small objects. Fourth image in the same row shows model's capability to handle occlusions. Lastly, second image in the last row showcases model's robustness in analysing images with a large number of objects.

To measure the average inference time, proposed model was evaluated in two different setups. In both scenarios, first the model weights were loaded into memory, after that prediction function was timed and results were aggregated across 400 runs. When executed on a GPU machine (NVIDIA Tesla T4), the model showed an average time of 0.27 second per image. This result is significantly lower than 10 second limit established for the inference time. When the model was executed on a CPU machine (Intel Xeon CPU

2.30GHz), average time was 5.76 seconds per image with standard deviation of 0.33. Even though the average inference time on a CPU machine is expectedly higher than on a GPU-enabled machine, it is also well under 10 second limit. Running proposed model on a CPU machine would allow to simultaneously process images from approximately ten different locations every minute.

Another important aspect to consider during model assessment is the distribution of confidence scores. In the scope of object detection task, confidence score is simply the probability for an object of a class to exist in the specific part of the input image. The higher the probability, the more confident a model is about this specific prediction. Therefore, for true positive predictions confidence scores are desired to be as high as possible. For false positive predictions lower confidence scores are expected. When these two conditions are satisfied, the detection threshold can be increased in order to avoid false positive predictions with low confidence scores. Hence overall accuracy of the model can be improved without re-training. The distribution of confidence scores for the model trained in the main experiment is shown in Figure 18.

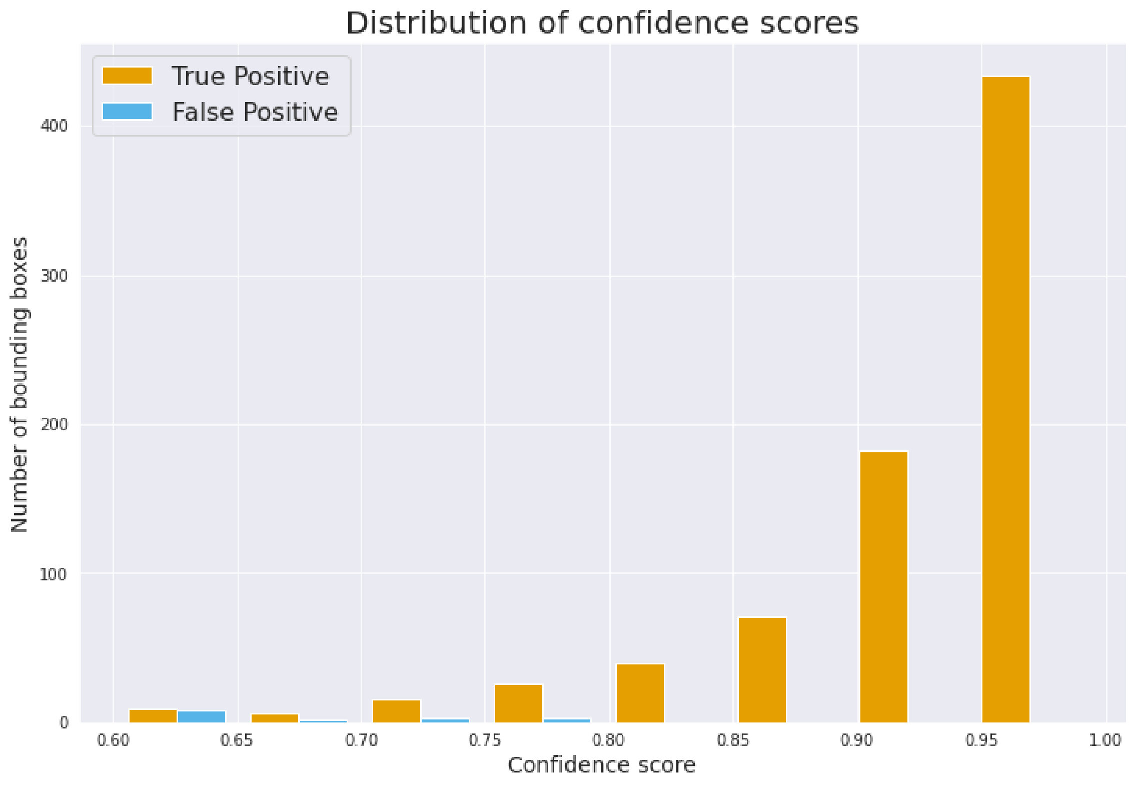


Figure 18. Distribution of confidence scores for true positive and false positive detections.

Upon inspection of the Figure 17, it is clear that confidence scores for true positive detections are indeed higher that confidence scores for false positive detections. For true positive detections most of the confidence scores lie above 0.95 with the average score being 0.925. Consequently, for false positive detections all the scores lie in the interval between 0.6 (threshold value) and 0.8. The average score is 0.67. However, since the model error rate is only 1%, increasing detection threshold would discard otherwise correctly identified objects. Therefore, the optimal solution is to keep detection threshold at its current value of 0.6.

6.2 Hold out test case

In the previous test case testing images are sampled from the same exact locations which were used to train the model. In this experiment, four different models are trained on a reduces set of images. Each model is trained on three locations and tested on the fourth, holdout location. By framing the test case in such a manner, a generalization capability of the model can be determined. Good performance signals a high generalization capability, whereas low performance means that such a model is only useful to make predictions for the environments it was exposed to during the training phase. Therefore, it would not be possible to apply this pre-trained model in a new location. From the scaling perspective, a model which generalizes well is highly desirable and can reduce both, development efforts and deployment times.

Table 3 shows a summary for four corresponding test cases. Each row contains test cases set up and final mAP scores.

Test case	Trained on	Tested on	mAP score
1	Locations 2, 3 and 4	Location 1	0.9496
2	Locations 1, 3 and 4	Location 2	0.9873
3	Locations 1, 2 and 4	Location 3	0.8478
4	Locations 1, 2 and 3	Location 4	0.5233

Table 3. Testing results for the holdout test cases.

By inspecting results in Table 3 and comparing them against typical images from each location, three important observations become evident.

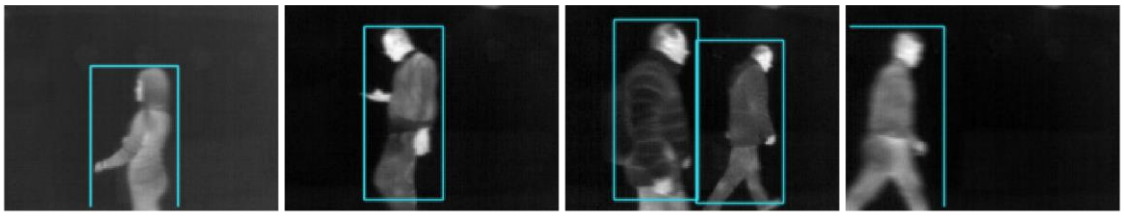
For the first two test cases, final mAP scores fall close to the mAP score achieved by the model in the main test case (mAP of 0.9912). In practice it means that these two models can be deployed with no further tuning yet provide reliable predictions, although during the training phase neither model was exposed to the images from the target environment.

Third test case which achieved mAP score of 0.8478 indicates that whilst the model has learned important features from the training images, its prediction power is below previously established optimizing metric. Hence, such a model cannot be directly applied for reliable object detection. However, in some cases a suboptimal model is better than no model. In ML, a situation which requires generating predictions in a new environment with no data available at the start is known as a cold-start problem.

Typically, a cold-start problem can be solved by providing a suboptimal solution at the start and gradually improving upon that solution as relevant data is being accumulated. Deploying a suboptimal object detection model in at a new location allows to produce good estimates for crowd counting straight away. When enough data is collected from that new location, initial model can be fine-tuned to reach required levels of performance.

Last holdout experiment, where the model is trained on images from locations 1, 2 and 3 and tested on images from location 4, shows an insufficient mAP score of 0.5233. This result indicates that close to half of all predictions made by the model are incorrect. Clearly such a model cannot be used and require another development cycle to be initiated to achieve better performance.

The results of the holdout experiment can be naturally explained by the complexity of the target location images used for testing. As seen in the Figure 19, images representing locations 1 and 2 are relatively simple.



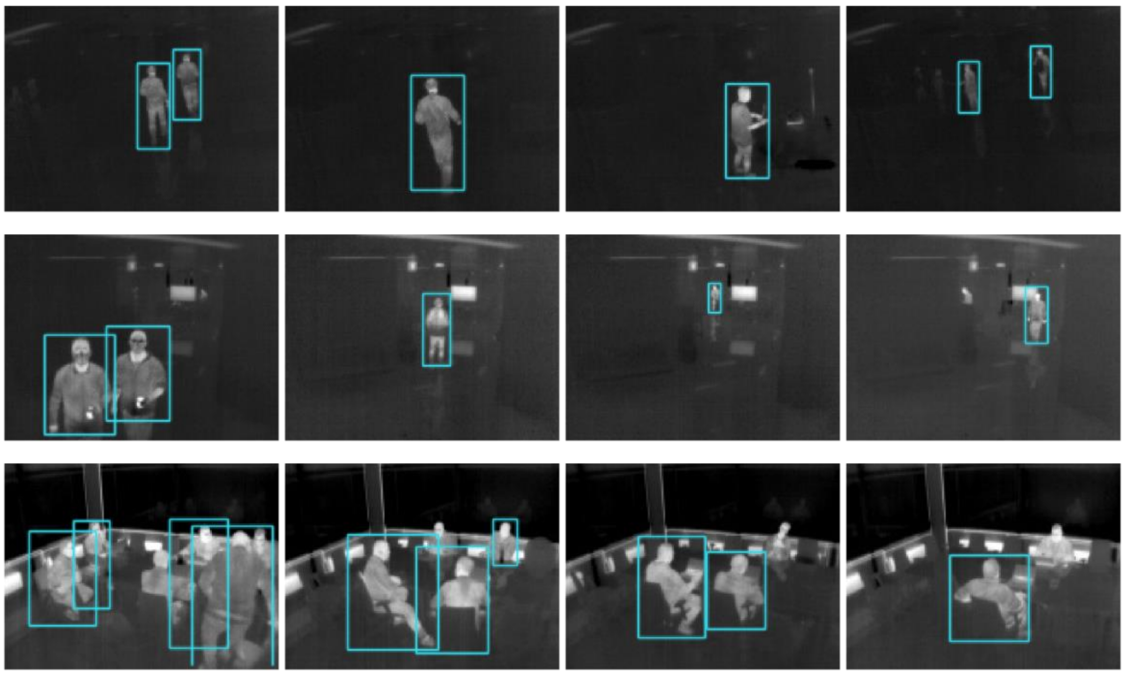


Figure 19. Sample predictions for the holdout test case.
From top to bottom: location 1, location 2, location 3, location 4.

Typically, as seen in images in the first row in Table19, there is only a single object in an image, no occlusions, very few reflections and very few examples of objects crossing image borders.

Subsequently, as shown in Figure 19, images from location 3 are of a medium complexity. In images from this location background includes other objects which in thermal imagery appear to be of the temperature close to the temperature of human bodies: working lamps, computer monitors and TV screens. Additionally, multiple reflections present in the images make object detection more challenging compared to images from first two locations.

Finally, images recorded in the fourth location are the most complex of all. There are more objects per image, occlusions are more prevalent, and background includes additional objects. Therefore, it is understandable that the model which has not been presented with images of a similarly high complexity during the training cannot accurately perform the task when faced with such images.

To conclude, the results of the holdout experiment suggest that proposed model is capable of a high degree of generalization when presented with images of the similar or lower complexity compared to the images it is trained on. This allows to ensure that a model trained on a subset of the images collected from the most challenging environments can be successfully re-deployed across many novel locations without additional development investments.

6.3 Error analysis

When assessing model's performance, it is critical to recognize its limitations. Conducting an error analysis of the model's predictions provides DL practitioners with insights into model's behaviour and highlights most promising directions for the model improvement. A sample of 400 testing images has been used to perform the error analysis of the proposed model. Results of the analysis are summarized in the Table 4.

Location	mAP score	Recall	Precision
Location 1	0.9488	90%	97.9%
Location 2	0.9995	100%	97.8%
Location 3	0.9662	96%	97.5%
Location 4	0.9977	98%	98.4%
Average	0.9785	96%	97.9%

Table 4. Error analysis results.

Table 4 shows that overall and per-location mAP scores of the model are about 0.95 and higher, indicating that required performance was reached in every location. At the same time, by inspecting Table 4, it becomes evident that the weakest part of the model is its recall in the first location. Essentially, this means that model performance can be further improved by reducing the number of missed objects in that location.

To understand what causes false negative detections among images recorded in the first location, a manual inspection of such images was conducted. Figure 20 summarizes the findings of this inspection.

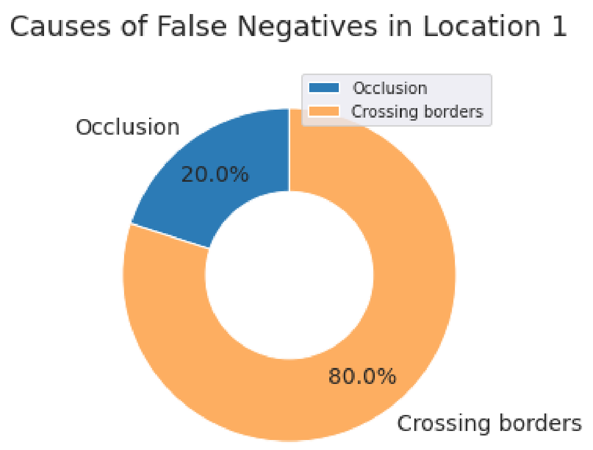


Figure 20. Sources of false negative detections in images from Location 1.

Figure 20 clearly shows that most of the false negatives in images from the first location are caused by target objects crossing image borders. In such images only a part of the person's body is visible, which drives the model to miss such objects. Therefore, a rational approach for improving model's accuracy is to collect more images where only a part of the object is included within image.

However, extra sources of incorrect predictions occur across images obtained from other locations. Figure 21 provides a detailed overview of these sources.

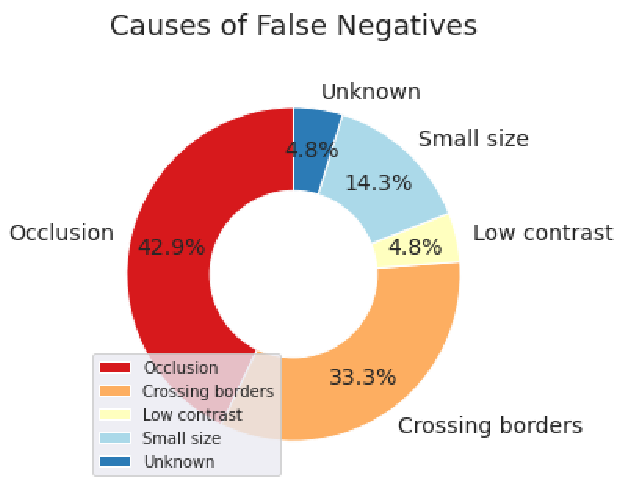


Figure 21. Sources of false negatives across all locations.

According to Figure 21, two leading sources of false negatives are occlusions and bounding boxes that cross image borders. Extending training dataset with corresponding images can potentially drive down the number of false negative detections. Similarly, factors which contribute to false positive detections can be seen in Figure 22.

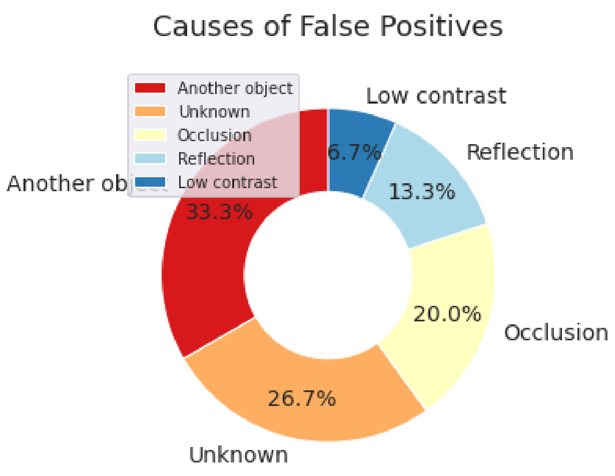


Figure 22. Sources of false positives across all locations.

Figure 22 shows that in addition to previously described sources of incorrect predictions, low contrast of the input images contributes to 6.7% of all false positives. This type of error can be illuminated by implementing an extra pre-processing step. By applying histogram equalization to an input image, the brightness and contrast across the image can be adjusted so that the model can correctly recognize target objects.

7 Conclusion

The primary objective of this project was to examine existing approaches to object detection task in thermal imagery. Both, traditional CV methods and DL-based methods were examined in detail. An exhaustive review of advantages and drawbacks of both techniques has suggested that as of today DL object detectors should be capable to outperform traditional counterparts, whilst simultaneously requiring substantially less development effort and domain knowledge.

Based on the results of the background study, a DL-based object detection system capable of processing thermal images was successfully implemented. The object detection network has fulfilled all initially established requirements. The model has shown a high mAP score of 0.9912. At the same time, average inference times achieved during model testing were correspondingly 0.27 second on a GPU-enabled machine and 5.76 second on a CPU machine.

Upon conducting a range of extensive test experiments, developed model demonstrated high generalization capabilities and robust performance. When trained on a carefully sampled set of images, the network can be deployed in novel locations without need to be re-trained. Furthermore, proposed model is able to efficiently handle reflections, occlusions, small sized objects and changes in brightness and contrast.

Obtained results demonstrate the fact that the object detection system developed as the part of this project is sufficient and ready to be productized. Deployment of the model across different location in Nokia Espoo campus would facilitate analysis of utilization rates and patterns in different premises, help to optimize crowd flows and drive down maintenance costs.

8 Future work

A step towards eventual productization for this project would be to deploy the final model into an embedded module such as Jetson Nano or similar instead of a conventional GPU. Investigating how performance and inference time change upon transition would be of equally interesting. Bringing computation to the edge would allow to reduce latency and network congestion whilst enabling real-time analysis and decision making.

Another possible direction for the future development includes changing a backbone network of a RetinaNet model from ResNet50 to an alternative network. Plausible candidates for the replacement are MobileNet and DenseNet networks. Comparing their performance against proposed solution would allow to choose the best suitable model based on the trade-off between accuracy and inference time.

Ultimately, current project could be extended even further to handle the task of crowd density estimation in highly challenging environments. The examples of possible environments are shown in Figure 23.

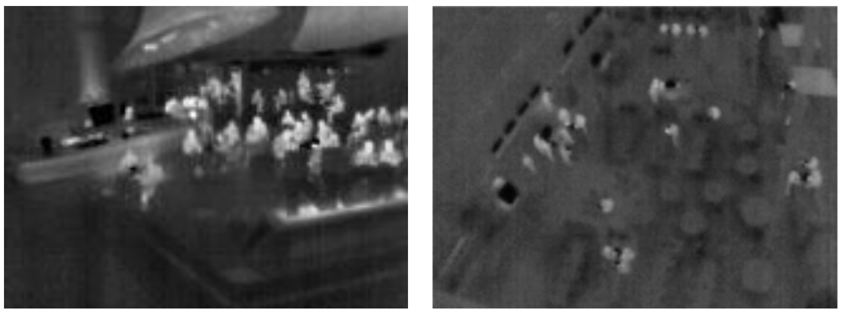


Figure 23. Thermal images taken from (a) lobby, side view and (b) cafeteria, top view.

As it can be seen in Figure 23, such images are increasingly more complex than the images used in this project. Therefore, more sophisticated approaches such as object segmentation or a combination of traditional CV and DL methods might be required to tackle the problem. Developing solution capable of analysing bigger crowds from further away would enable a whole range of new applications at a reduced cost due to less camera installations required.

References

1 Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, et. al. Deep Learning for Generic Object Detection: A Survey [online]. 2019 Aug; [cited 2019 Sep 15]. Available from: https://arxiv.org/abs/1809.02165.

2 Michael Vollmer, Klaus-Peter Möllmann. Infrared Thermal Imaging: Fundamentals, Research and Applications, Second Edition [e-book]. WILEY-VCH, 2017 Dec; [cited 2019 Sep 19]. Available from: https://onlinelbrary.wiley.com/doi/book/10.1002/9783527693306

3 Amanda Berg. Detection and Tracking in Thermal Infrared Imagery [licentiate's thesis online]. Linköping University; 2016; [cited 2019 Sep 19]. Available from: https://www.diva-portal.org/smash/get/diva2:918038/FULLTEXT01.pdf

4 Nigel J. W. Morris, Shai Avidan, Wojciech Matusik, Hanspeter Pfister. Statistics of Infrared Images. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition; 2007; [cited 2019 Sep 19]. Available from: https://www.researchgate.net/publication/4260057_Statistics_of_Infrared_Images

5 Grevelink Evelyn. A Closer Look at Object Detection, Recognition and Tracking [online]. Intel; 2017 Dec; [cited 2019 Sep 25]. Available from: https://software.intel.com/en-us/articles/a-closer-look-at-object-detection-recognition-and-tracking

6 Hussein Adnan Mohammed. Object Detection and Recognition in Complex Scenes [master thesis online]. 2014; [cited 2019 Sep 25]. Available from: https://sapientia.ualg.pt/bitstream/10400.1/8368/1/Master%20Thesis.pdf

7 Andreopoulos Alexander, Tsotsos John. 50 Years of object recognition: Directions forward. Computer Vision and Image Understanding; August 2013, Vol. 117, (No. 8): pp. 827–891.

8 Lee Andy. Comparing Deep Neural Networks and Traditional Vision Algorithms in Mobile Robotics [online]. 2016; [cited 2019 Oct 3]. Available from: https://www.semanticscholar.org/paper/Comparing-Deep-Neural-Networks-and-Traditional-in-Lee/1b6f569b79721037425fca034c7ae47904fb9276.

9 Lowe D. G. Object recognition from local scale-invariant features [online]. Proceedings of the Seventh IEEE International Conference on Computer Vision, IEEE; 1999 Sep; [cited 2019 Oct 6]. Available from: https://ieeexplore.ieee.org/document/790410.

10 David G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. International Journal of Computer Vision; November 2004, Vol. 60 (No. 2): pp. 91-110.

11 Introduction to SIFT (Scale-Invariant Feature Transform) [online]. OpenCV - Python tutorials. 2019; [cited 2019 Oct 6]. Available from: https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html

12 Comparison between SIFT and SURF image forgery Algorithms [online]. International Journal of Computer Applications, Vol. 164, (No 2); April 2017; [cited 2019 Oct 6]. Available from: https://pdfs.semanticscholar.org/dde9/2b750cb9db11292b0a7ff83dc0f0565da16e.pdf

13 Bay Herbert, Tuytelaars Tinne, Van Gool Luc. Speeded-Up Robust Features (SURF). Computer Vision and Image Understanding; June 2008, Vol. 110, (No. 3): pp. 346-359.

14 Wojnar Anna, Pinheiro António M. G. Annotation of medical images using the SURF descriptor [online]. 9th IEEE International Symposium on Biomedical Imaging (ISBI); May 2012; [cited 2019 Oct 10]. Available from: https://ieeexplore-ieee-org.ezproxy.metropolia.fi/document/6235501.

15 Juan Luo, Gwun Oubong. A Comparison of SIFT, PCA-SIFT and SURF. International Journal of Image Processing (IJIP); April 2009, Vol. 3, (No. 4): pp. 143-152.

16 Viola P., Jones M. Rapid object detection using a boosted cascade of simple features [online]. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004 May. [cited 2019 Oct 12]. Available from: http://www.merl.com/publications/docs/TR2004-043.pdf

17 Histogram of Oriented Gradients [online]. [cited 2019 Oct 15]. Available from: http://scikit-image.org/docs/dev/auto_examples/features_detection/plot_hog.html.

18 Dalal Navneet, Triggs Bill. Histograms of Oriented Gradients for Human Detection [online]. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005; [cited 2019 Oct 15]. Available from: https://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf

19 Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks [online]. NIPS 2012; [cited 2019 Oct 21]. Available from: https://dl.acm.org/citation.cfm?id=3065386

20 Goodfellow et al. Deep Learning. MIT Press, 2016.

21 Stanford. CS 229 - Machine Learning. Deep Learning Cheat Sheet [online]. 2018 Sep; [cited 2019 Oct 23]. Available from: https://stanford.edu/~shervine/teaching/cs-229/cheatsheet-deep-learning

22 Jayesh Bapu Ahire. The Artificial Neural Networks Handbook: Part 4 [online]. 2018 Nov; [cited 2019 Oct 23]. Available from: https://medium.com/@jayeshbahire/the-artificial-neural-networks-handbook-part-4-d2087d1f583e

23 Aurélien Géron. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2017.

24 François Chollet. Deep Learning with Python. Manning, 2018.

25 Scott Martin. What's the Difference Between a CNN and an RNN? [online]. NVIDIA blog, 2018 Sep; [cited 2019 November 3]. Available from: https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/

26 Xiongwei Wua, Doyen Sahooa, Steven C.H. Hoia. Recent Advances in Deep Learning for Object Detection [online]. 2019 Aug; [cited 2019 Nov 10]. Available from: https://arxiv.org/pdf/1908.03673.pdf

27 Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, et al. SSD: Single Shot MultiBox Detector [online]. 2016 Dec; [cited 2019 Nov 10]. Available from: https://arxiv.org/abs/1512.02325

28 Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi. You Only Look Once: Unified, Real-Time Object Detection [online]. 2016 May; [cited 2019 Nov 10]. Available from: https://arxiv.org/abs/1506.02640

29 Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (6): 1137–49, 2017.

30 Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation [online]. 2014 Oct; [cited 2019 Nov 14]. Available from: https://arxiv.org/abs/1311.2524

31 Ross Girshik. Fast R-CNN [online]. 2015 Sep; [cited 2019 Nov 14]. Available from: https://arxiv.org/abs/1504.08083

32 Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick. Mask R-CNN [online]. 2018 Jan; [cited 2019 Nov 14]. Available from: https://arxiv.org/abs/1703.06870

33 Lilian Weng. Object Detection Part 4: Fast Detection Models [online]. 2018 Dec; [cited 2019 Nov 15]. Available from: https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html

34 Haritha Thilakarathne. Deep Learning Vs. Traditional Computer Vision [online]. 2018 Aug; [cited 2019 Nov 21]. Available from: https://naadispeaks.wordpress.com/2018/08/12/deep-learning-vs-traditional-computer-vision/

35 Ovgu Ozturk, Toshihiko Yamasaki, Kiyoharu Aizawa. Tracking of Humans and Estimation of Body/Head Orientation from Top-view Single Camera for Visual Focus of Attention Analysis [online]. IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops, 2009; [cited 2019 Nov 25] Available from: https://ieeexplore.ieee.org/document/5457590

36 Osameh Biglari, Reza Ahsan, Majid Rahi. Human Detection Using SURF and SIFT Feature Extraction Methods in Different Color Spaces [online]. Journal of mathematics and computer Science, Vol. 11; 2014; [cited 2019 Nov 25]. Available from: https://www.semanticscholar.org/paper/Human-Detection-Using-Surf-And-Sift-Feature-Methods-Biglari-Ahsan/9787d08a742197df608cd6169d1297cae1795505

37 Kai Jungling, Michael Arens. Feature based person detection beyond the visible spectrum [online]. 2014 May; [cited 2019 Nov 25]. Available from: https://www.researchgate.net/publication/220669141_Feature_based_person_detection_beyond_the_visible_spectrum

38 Li Zhang, Bo Wu, Ram Nevatia. Pedestrian Detection in Infrared Images based on Local Shape Features [online]. 2007; [cited 2019 Nov 25]. Available from: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.309.9773&rep=rep1&type=pdf

39 Yannick Benezeth, Bruno Emile, Helene Laurent, Christophe Rosenberger. A Real Time Human Detection System Based on Far Infrared Vision [online]. ICISP 2008,

LNCS 5099, pp. 76–84; 2008; [cited 2019 Nov 25]. Available from: https://link.springer.com/content/pdf/10.1007/978-3-540-69905-7_9.pdf

40 Eun Som Jeon, Jong-Suk Choi, Ji Hoon Lee, Kwang Yong Shin, Yeong Gon Kim, Toan Thanh Le, et al. Human Detection Based on the Generation of a Background Image by Using a Far-Infrared Light Camera [online]. 2015 Mar; [cited 2019 Nov 25]. Available from: https://www.researchgate.net/publication/274091798_Human_Detection_Based_on_the_Generation_of_a_Background_Image_by_Using_a_Far-Infrared_Light_Camera

41 Erik Valldor. Person Detection in Thermal Images using Deep Learning [master thesis online]. 2018 Jun; [cited 2019 Nov 30]. Available from: http://uu.diva-portal.org/smash/get/diva2:1275338/FULLTEXT01.pdf

42 Andres Gomez, Francesco Conti, Luca Benini. Thermal ImageBased CNN's for Ultra-Low Power People Recognition [online]. Proceedings of Low Power Embedded Systems Workshop. ACM, New York, NY, USA. 2018 May; [cited 2019 Nov 30]. Available from: https://www.researchgate.net/publication/326645928_Thermal_image-based_CNN's_for_ultra-low_power_people_recognition

43 Marina Ivašić-Kos, Mate Krišto, Miran Pobar. Human Detection in Thermal Imaging Using YOLO [online]. 2019 Apr; [cited 2019 Nov 30]. Available from: https://www.researchgate.net/publication/333360405_Human_Detection_in_Thermal_Imaging_Using_YOLO

44 Xinran Wang. Human Detection in a Sequence of Thermal Images using Deep Learning [master thesis online]. 2019 Feb; [cited 2019 Nov 30]. Available from: https://library.itc.utwente.nl/papers_2019/msc/gfm/WangXinran.pdf

45 Jeremy Jordan. Organizing machine learning projects: project management guidelines [online]. 2018 Sep [cited 2020 Feb 2]. Available from: https://www.jeremyjordan.me/ml-projects-guide/

46 Andrew Ng. Machine Learning Yearning [online]. 2018 [cited 2020 Feb 8]. Available from: https://github.com/ajaymache/machine-learning-yearning/blob/master/full%20book/machine-learning-yearning.pdf

47 Precision and recall. en.wikipedia.org. 2020 Mar, [cited 2020 Mar 8]. Available from: https://en.wikipedia.org/wiki/Precision_and_recall

48 Rajalingappaa Shanmugamani. Deep Learning for Computer Vision [online]. Packt Publishing, 2018.

49 Jonathan Hui. mAP (mean Average Precision) for Object Detection [online]. 2018 Mar, [cited 2020 Mar 8]. Available from: https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173

50 Garbage in, garbage out. en.wikipedia.org. 2020 Jan [cited 2020 Feb 14]. Available from: https://en.wikipedia.org/wiki/Garbage_in,_garbage_out

51 Emma Strubell, Ananya Ganesh, Andrew McCallum. Energy and Policy Considerations for Deep Learning in NLP [online]. 2019 Jul, [cited 2020 Feb 16]. Available from: https://arxiv.org/abs/1906.02243

52 Nokia website. Sustainability [online]. 2020 Jan, [cited 2020 Feb 16]. Available from: https://www.nokia.com/about-us/sustainability/

53 Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollar. Focal Loss for Dense Object Detection [online]. 2018 Feb, [cited 2020 Feb 16] Available from: https://arxiv.org/pdf/1708.02002.pdf

54 Keras implementation of RetinaNet. Github repository. Available from: https://github.com/fizyr/keras-retinanet

55 Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan et al. Feature Pyramid Networks for Object Detection [online]. 2017 Apr, [cited 2020 Feb 16]. Available from: https://arxiv.org/abs/1612.03144

56 Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition [online]. 2015 Dec, [cited 2020 Feb 16]. Available from: https://arxiv.org/abs/1512.03385

57 Fabio M. Graetz. RetinaNet: how Focal Loss fixes Single-Shot Detection [online]. Nov 2018, [cited 2020 Feb 16]. Available from: https://towardsdatascience.com/retinanet-how-focal-loss-fixes-single-shot-detection-cb320e3bb0de

58 Adrian Rosebrock. Intersection over Union (IoU) for object detection [online]. 2016 Nov, [cited 2020 Feb 16]. Available from: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

59 Victor Prisacariu. FastHOG-a real-time GPU implementation of HOG [online]. 2009 Aug, [cited 2020 Feb 16]. Available from: https://www.researchgate.net/publication/228881235_fastHOG-a_real-time_GPU_implementation_of_HOG

60 Keras documentation [online]. 13 Oct 2019, [cited 1 Mar 2020]. Available from: https://keras.io/

61 Google Colaboratory Frequently Asked Questions [online]. [cited 1 Mar 2020]. Available from: https://research.google.com/colaboratory/faq.html