

# **Object Detection in Thermal Images using the High-Resolution Network**

Submitted by:

**Spyros Lontos**

**MSc Data Science**

in the Department of Computer Science

**The University of Bath**

**September 2021**

## Abstract

This dissertation aims to research if object detection in thermal imagery can be improved by using the High-Resolution backbone. Using the same Faster R-CNN architecture the results were compared between the tested HRNet backbones and the baseline ResNet-50-FPN. The models were all pre-trained on the COCO 2017 dataset and fine-tuned on the FLIR ADAS thermal dataset that contains annotations for cars, people, and bicycles. Both training and testing was done using Google Colab GPU accelerated notebooks with the MMDetection framework that accommodates new and old research models and backbones. From the testing results obtained the High-Resolution backbone with convolution width of 40 had achieved a 7% improvement when compared to the baseline ResNet-50-FPN backbone. This showed that HRNet is better fit for object detection applications on the thermal domain.

## Acknowledgments

I would like to express my sincere gratitude to my supervisors, Dr Olga Isupova and Dr Alan Hayes for their guidance and assistance throughout the completion of this study. I also want to thank my family and partner Rafailia for their support and patience during my academic journey.

# Table of Contents

<b>TABLE OF CONTENTS .....</b>	<b>I</b>
<b>LIST OF FIGURES .....</b>	<b>II</b>
<b>LIST OF TABLES .....</b>	<b>III</b>
<b>CHAPTER 1 INTRODUCTION .....</b>	<b>4</b>
1.1    BACKGROUND .....	4
1.2    SCOPE .....	8
<b>CHAPTER 2 LITERATURE SURVEY .....</b>	<b>9</b>
<b>CHAPTER 3 METHODOLOGY .....</b>	<b>13</b>
3.1    INTRODUCTION .....	13
3.2    APPROACH .....	14
<b>CHAPTER 4 EXPERIMENTS .....</b>	<b>16</b>
4.1    DATASET .....	16
4.2    ENVIRONMENT .....	18
4.3    DATASET PRE-PROCESSING .....	20
4.4    MODELS .....	21
4.4.1    BASELINE: FASTER R-CNN RESNET-50 FPN .....	21
4.4.2    FASTER R-CNN HRNETV2P MODELS .....	21
4.5    DATA PIPELINES AND PARAMETERS .....	24
4.6    HYPERPARAMETERS .....	26
4.7    TRAINING AND VALIDATION RESULTS .....	30
4.8    TESTING RESULTS .....	37
<b>CHAPTER 5 CONCLUSIONS .....</b>	<b>41</b>
5.1    PROJECT SUMMARY .....	41
5.2    FUTURE WORK .....	42
5.3    REFLECTIONS .....	43
<b>BIBLIOGRAPHY .....</b>	<b>44</b>

# List of Figures

FIGURE 1. A ROAD MAP OF OBJECT DETECTION (ZHENGXIA, ET AL., 2019) .....	5
FIGURE 2. THERMAL IMAGE OF A SEARCH AND RESCUE TRAINING EXERCISE <a href="http://www.thedronesmag.com/review-dji-zenmuse-xt/">HTTP://WWW.THEDRONESMAG.COM/REVIEW-DJI-ZENMUSE-XT/</a> [ACCESSED ON 31 AUG 2021] .....	6
FIGURE 3. SIDE-BY-SIDE COMPARISON OF VISIBLE-LIGHT IMAGE AND A THERMAL IMAGE IN A FIREFIGHTING TRAINING EXERCISE <a href="http://www.thedronesmag.com/review-dji-zenmuse-xt/">HTTP://WWW.THEDRONESMAG.COM/REVIEW-DJI-ZENMUSE-XT/</a> [ACCESSED ON 31 AUG 2021] .....	7
FIGURE 4. AN EXAMPLE OF A HIGH-RESOLUTION NETWORK (WANG, ET AL., 2020) .....	15
FIGURE 5. EXAMPLE THERMAL ANNOTATED IMAGES .....	16
FIGURE 6. FLID ADAS ANNOTATION DISTRIBUTION BETWEEN THE 3 MAIN FOLDERS .....	18
FIGURE 7. ENVIRONMENT LIBRARY VERSIONS USED, PYTORCH 1.9.0   CUDA 10.2 .....	19
FIGURE 8. PRE-PROCESSING STEP. EDITING IMAGE FILE_NAME FIELDS .....	20
FIGURE 9. FASTER_RCNN_HRNETV2P_W32_1X_COCO CONFIGURATION FILE .....	22
FIGURE 10. FASTER_RCNN_HRNETV2P_W18_1X_COCO CONFIGURATION FILE .....	23
FIGURE 11. FASTER_RCNN_HRNETV2P_W40_1X_COCO CONFIGURATION FILE .....	23
FIGURE 12. TRAIN AND TEST DATA PIPELINES .....	25
FIGURE 13. DATA CONFIGURATION .....	25
FIGURE 14. FASTER_RCNN_ReNet-50_FPN_1X HYPERPARAMETER CONFIGURATION.....	26
FIGURE 15. INTERSECTION OVER UNION EXAMPLE <a href="https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2">HTTPS://TOWARDSDATASCIENCE.COM/MAP-MEAN-AVERAGE-PRECISION-MIGHT-CONFUSE-YOU-5956F1BFA9E2</a> [ACCESSED ON 1 SEP 2021] .....	28
FIGURE 16. CORRESPONDING COLOURS FOR EACH TRAINED MODEL. ....	30
FIGURE 17. TRAINING LOSS METRIC FOR ALL 4 RUNS.....	31
FIGURE 18. ADDITIONAL TRAINING METRICS UPPER LEFT: BOUNDING BOX LOSS. UPPER RIGHT: CLASSIFICATION LOSS. BOTTOM LEFT: RPN BOUNDING BOX LOSS. BOTTOM RIGHT: RPN CLASSIFICATION LOSS.....	32
FIGURE 19. VALIDATION LOSS METRIC FOR ALL 4 RUNS.....	33
FIGURE 20. ADDITIONAL VALIDATION METRICS UPPER LEFT: BOUNDING BOX LOSS. UPPER RIGHT: CLASSIFICATION LOSS. BOTTOM LEFT: RPN BOUNDING BOX LOSS. BOTTOM RIGHT: RPN CLASSIFICATION LOSS.....	34
FIGURE 21. AVERAGE PRECISION % AT IoU=.50:.05:.95 (PRIMARY CHALLENGE METRIC) .....	35
FIGURE 22. LEFT: AVERAGE PRECISION % AT IoU=.50 (PASCAL VOC METRIC) RIGHT: AVERAGE PRECISION % AT IoU=.75 (STRICT METRIC) .....	35
FIGURE 23. AVERAGE PRECISION ACROSS SCALES VALIDATION RESULTS LEFT: AVERAGE PRECISION % FOR SMALL OBJECTS: AREA < 32 <sup>2</sup> MIDDLE: AVERAGE PRECISION % FOR MEDIUM OBJECTS: 32 <sup>2</sup> < AREA < 96 <sup>2</sup> RIGHT: AVERAGE PRECISION % FOR LARGE OBJECTS: AREA > 96 <sup>2</sup> .....	36
FIGURE 24. ResNet50 MAP AND MAP_50 RESULTS .....	38
FIGURE 25. HRNetW18 MAP AND MAP_50 RESULTS .....	38
FIGURE 26. HRNetW32 MAP AND MAP_50 RESULTS .....	39
FIGURE 27. HRNetW40 MAP AND MAP_50 RESULTS .....	39

# List of Tables

TABLE 1. PERFORMANCE COMPARISON OF THE 4 TESTED MODELS .....	40
TABLE 2. MAP PERCENTAGE DIFFERENCE OF THE MODELS COMPARED TO THE RESNET50 BASELINE...	40

# Chapter 1

## Introduction

### 1.1 Background

Computer vision techniques have been more and more prevalent over the years. As more advantages of such processing techniques are identified, they are being applied across industries with a remarkable effect. One of these techniques that has seen the most advancement in the last few years is automated and assisted driving. Person detection and road sign detection have been implemented by many car manufacturers and have been proven to have a great effect in decreasing the number of accidents (Zachary, 2020), as also enabling the further development of self-driving vehicles.

Publications on object detection methods have been constantly increasing over the past 20 years but especially after the Major Transitional Period to deep learning-based detection methods. Milestone detectors such as Faster-RCNN (Shaoqing, et al., 2015), Single-Shot Detector (SSD) (Liu, 2016) and You Only Look Once (YOLO) (Redmon, et al., 2016) shown in Figure 1, all stem from the breakthrough of AlexNet (Krizhevsky, et al., 2012) with its deep learning approach that had outperformed all the traditional object detectors at the time.

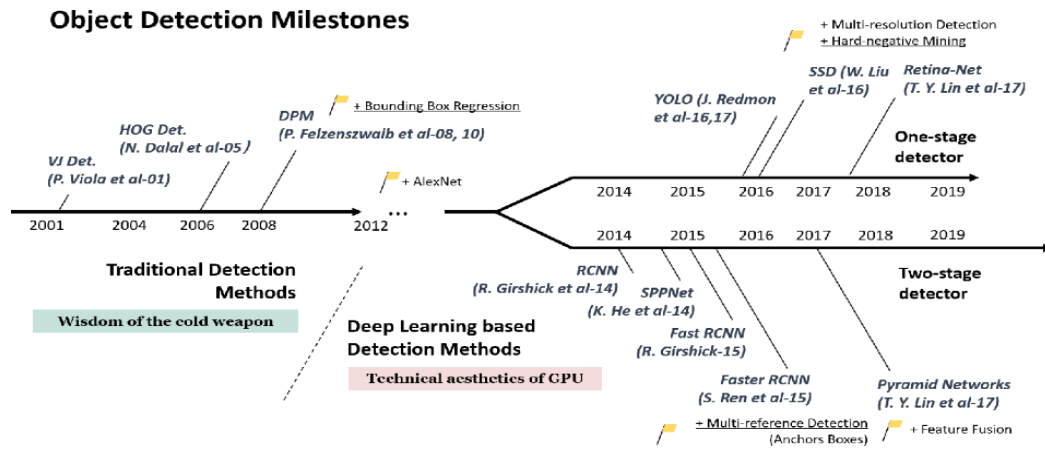


Figure 1. A road map of object detection (Zhengxia, et al., 2019)

As more research was applied to the field, the constant improvement between detectors was immense. With the first implementation of You Only Look Once (YOLO) detector in 2016 presented in the Conference on Computer Vision and Pattern Recognition. The novel detection approach had managed to perform similarly in terms of accuracy with the slower detectors such as RCNN (Girshick, 2014) but it was able to run real-time at 45 frames per second thus significantly increasing its possible applications.

Object detection has been proven to work exceptionally well on normal colour images, but with little research being done on thermal images the results obtained on a thermal dataset is still relatively low to be used for real-world applications. With some recent research being done using a much-improved detector like YOLOv3 and applied to distinguish people in surveillance footage of thermal images they had only manage to achieve 19% in Average Precision on the Person class which is much lower than the 90% Average Precision which normal coloured images provide (Krišto, et al., 2020). Although, the precision amounts are only specific to their application it still identifies the significant gaps that thermal images have in the implementation of this computer vision technique.



## Chapter 1: Introduction

Thermal imaging is using cameras have sensors that can detect a different range of wavelengths in the electromagnetic spectrum. Instead of depending on other sources illuminating the environment, thermal sensors can detect heat footprints from the objects within the infrared range and are able to create a complete picture even in complete darkness.

Thermal imaging cameras can capture information in a large range of the Infrared field within the electromagnetic spectrum. Near Infrared (NIR) is between 0.75 and 1 microns in wavelength, and it usually requires some source that provides active illumination. Whereas Long Wave Infrared (LWIR) is between 8 and 14-micron wavelength and it can observe the energy emitted from different objects within the scene such as shown in Figure 2. This can significantly help in identifying different objects within the scene especially in harsh environments.



Figure 2. Thermal image of a search and rescue training exercise  
<http://www.thedronesmag.com/review-dji-zenmuse-xt/> [Accessed on 31 Aug 2021]

## Chapter 1: Introduction

There is a great deal of advantages that thermal images can provide over normal colour images. With their unique features being that they can perform well in both day and night whilst they are also unaffected by harsh weather conditions such as rain and fog. Such example is shown in Figure 3 where on the left no information can be distinguished on the picture taken by a normal colour camera whereas on the right a firefighter with their equipment can be clearly seen.

The disadvantage of thermal imaging cameras is can they be relatively expensive when compared to their colour counterparts. This had greatly affected the accessibility of capturing large amounts of thermal images so that they can be further used in computer vision research.

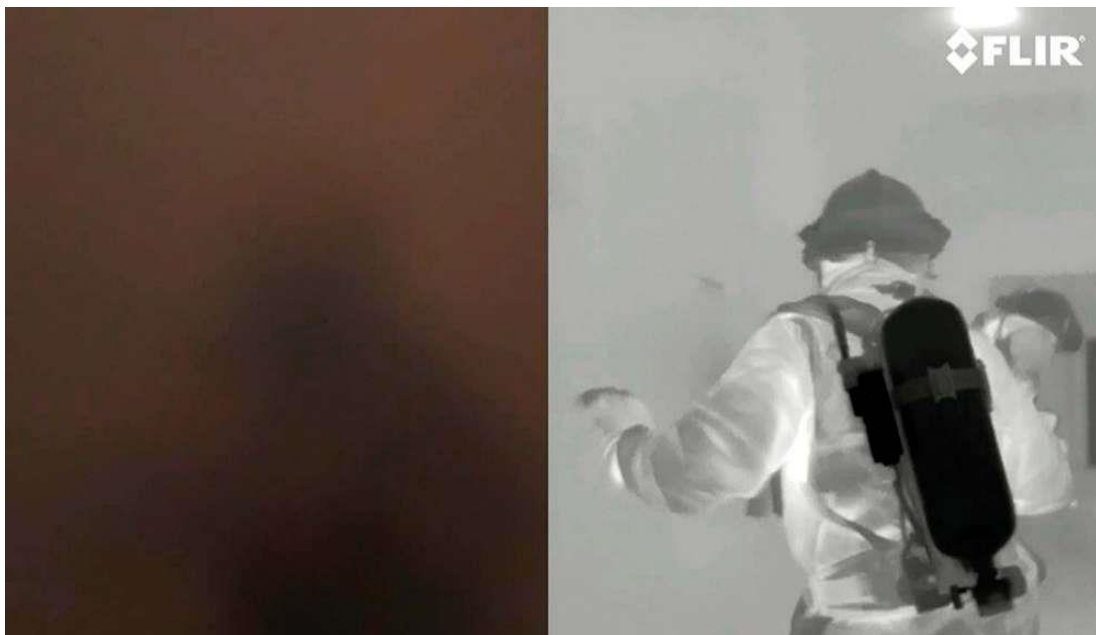


Figure 3. Side-by-side comparison of visible-light image and a thermal image in a firefighting training exercise <http://www.thedronesmag.com/review-dji-zenmuse-xt/>  
[Accessed on 31 Aug 2021]

## 1.2 Scope

The project's main aims and objectives will be to assess how object detection performance on thermal images can be affected when different model structures are used. By assessing the previous approaches and methodologies that have been researched within the field, a clearer picture can be made on the possible gaps, improvements, and evolutions and how these can be further used to adapt a more appropriate implementation of the subject.

Finding any performance benefits could have a significant impact in the extension of thermal specific detection applications. Thermal imaging data can provide significant advantages as previously discussed so identifying any possible points of improvement could be exceptionally valuable for this field.

Through the literature survey performed possible gaps that were identified depend mostly on that thermal imaging data do not contain a lot of information for this position-sensitive vision technique. On several methods applied it was mostly trying to extract more information within the thermal data as the results obtained using the standard colour object detection methods were sub-par.

A possible gap was identified, where typical detection models use feature extractor backbones that initially encode the image to a low-resolution representation. Since thermal imaging data showed that they lack information, this approach could mean that it further enhances the issues.

The project will compare a different feature extractor approach that can maintain high-resolution representation throughout the whole process. It should be able to maintain more semantic information and improve the detection performance when compared to a baseline approach of a typical model backbone.

## Chapter 2

# Literature Survey

Through researching the different object detection methods on thermal imaging data this section aims to provide a comprehensive literature review on the most relevant techniques and methods that have been applied in this field. Comparing in chronological order the different methods will aid in identifying the progress that has been made throughout the years. What gaps have been filled in and what remains that would inhibit the expected results this research aims to find.

In 2013, after the introduction of AlexNet, research introduced by (Woeber, et al., 2013) tried to adapt a different detection approach using a principal component analysis (PCA) based learning procedure to analyse and detect objects within thermal images. Although this piece of work did show that the implementation of their algorithms had been successful, the results obtained were insufficient to be used in their time-critical application of semi-autonomous convoying performing transportation missions. Considered in this work was that more sophisticated algorithms or approaches should have been used to achieve both better matching performance and speed. Although, the results that had been presented were not that promising in replacing deep neural networks, it was an initial approach in adapting thermal imaging data to an application that could significantly benefit from its use. Being able to detect objects within shadow and completely dark areas was a huge aspect in the approach of this research group and they made a significant effort in trying to adapt it to their real-life example.

By 2015, a much different approach by (John, et al., 2015) was using fuzzy C-means clustering, or soft k-means clustering for the detection aspect of their approach. Passing it to a convolutional neural network to learn the different features and perform the binary classification on identifying pedestrians within the thermal images. Their soft k-means approach had a significant limitation in that the number of clusters used, as for each image the number needed to be specified. Using a cluster map created from membership of the pixel intensities, they extracted candidate regions to perform binary classification on the presence of pedestrians. Their approach combined different machine learning techniques and had still managed to achieve greater results than pre-existing solutions. With a lower false-positive rate but at a much slower processing time of over 2.5 seconds per frame, they could identify in both day and night-time the pedestrians within the different scenes.

As the implementation of deep convolutional neural networks and the creation of new more sophisticated models had been making their progress through the object detection field several researchers focused on using these classical convolutional neural network architectures and adapting them to their thermal imagery specific application. Research from (Liu, et al., 2017) tried to account on the negative aspects of thermal images specifically on the difficulty in distinguishing features amongst images. They implemented a domain adaptation method of using pre-trained convolutional neural networks based on normal colour images and then training on thermal imaging data. Due to features obtained by the network not being enough due to lack of spatial information they proposed and adapted an ensemble method based on Kullback–Leibler (KL) divergence using different weak tracers to manage to achieve results for tracking objects. The thermal infrared tracking benchmarks they chose to use were the VOT-TIR 2015 (Berg, et al., 2015), a collection of thermal image sequences which was first proposed in 2015 used to test the different detection trackers developed and the new and enhanced version VOT-TIR 2016 (Felsberg, 2016). When compared to previously created and adapted models their approach had shown some

promising performance, but the major weaknesses that this domain had would first need to be addressed to get to a state so that it could be used in real life applications and situations.

The common adaptation technique of using RGB pre-trained convolutional neural networks as a domain adaptation approach that is to be transferred on infrared-based detection has been all and more common during the passing years of this research field. A paper by (Herrmann, et al., 2018) presented the application of pre-processing methods such as inverting, stretching, equalizing, or blurring the thermal image which had managed to reduce the performance difference between the RGB and thermal imaging object recognition. Within this research, the benchmark dataset which had been used is the public Korea Advanced Institute of Science and Technology (KAIST) dataset (Hwang, et al., 2015). It was explained that disadvantages of thermal sensors are mainly due to their high cost, as the common sensors used to obtain the data to be used are much lower resolution than their visual counterparts. This has the effect of reducing the features that can be extracted within the images which has a significant impact in the results of the computer vision technique.

Of the latest reports done on thermal imaging object detection was presented in the 2019 Conference on Computer Vision and Pattern Recognition. They stayed away from the basic implementations of deep neural networks but had shown to adapt novel techniques to the field, in attempting to improve the detection performances. (Ghose, et al., 2019) proposed working with saliency maps to be used as an aid in pedestrian detection, especially for daytime where it was observed that thermal image detection was at a major disadvantage. Using different state-of-the-art saliency methods such as PiCa-Net (Liu, 2018) and R3-Net (Deng, et al., 2018) they tried to adapt the information obtained to complement the thermal images. They attempted to fuse together the extracted saliency maps with the 3-channel thermal image to provide additional information when passed through the deep region-based neural network. This method had given better results as when compared to the standard performance

of thermal images, with lower overall miss rates within both daytime and night-time thermal images. Although the results seemed very promising it is another indication of how the low-resolution thermal images can lack information, needing additional techniques to properly establish the appropriate object detection processes.

(Devaguptapu, et al., 2019) showed another approach in the CVPR 2019. It was a pseudo-multi modal approach that used the thermal images to create pseudo-RGB equivalent images to then be combined and used within the deep neural network. Again, this paper has also shown that it needed to extract more information within the thermal images, so their approach was to take advantage of the development of the computer vision techniques on RGB imagery which have been proven successful over the years in the field of object detection. By using models which have been pre-trained on large-scale RGB datasets such as PASCAL-VOC (Everingham, 2009) and MS-COCO (Tsung-Yi, et al., 2014) and performing an image-to-image translation of thermal to RGB equivalent images their multi-modal architecture helped borrow the high-level features and with domain transfer, it would improve the object detection within the thermal domain. Using the FLIR ADAS thermal image dataset (Group, 2018) the multi-modal framework approach had achieved higher average precision results amongst classes when compared to the baseline performance. Trying to exploit the benefits and advancements that had been observed in RGB imagery they attempted to transfer the feature learnability of models that had been previously trained on normal coloured images to then be combined and trained to thermal imaging datasets so that it would reach similar performance.

## Chapter 3

# Methodology

### 3.1 Introduction

Through the literature survey research, the progress that has been observed is fully noted but within the field, there are many gaps that remain. For every approach they needed to add additional methods to manage to achieve noteworthy results. This focus was based on the thermal imaging dataset limitations. A big issue with thermal sensors is that they are relatively expensive. Capturing high-resolution thermal images can be very costly, thus most of the available material that is gathered is typically low-resolution imagery which contains some information but not enough so that the features extracted within the images are sufficient in achieving good performance. Common issues that have been observed is that objects could not be easily distinguished when they are grouped in clusters or when they are far away.

Additionally, though the research that was discussed such as in (Devaguptapu, et al., 2019), their proposed adaptation of a multimodal framework approach they used a combination of Faster-RCNN (Ren, et al., 2015) models with a Residual Network (ResNet) (He, 2016) backbone.

ResNet become one of the most common backbones used within the different detection models. The work introduced by (He, 2016) managed to overcome the optimization issues that very deep neural networks had. It was previously believed that to improve the performance of more convolution layers needed to be used. This work



showed that simply stacking layers, after a certain limit it produced the opposite results with increasing error rates on test and validation accuracy. With the use of skip connectors within their residual blocks it solved the issue of vanishing gradient as it allowed for alternate shortcuts for the gradient to flow through. Although this work did manage to fix this crucial issue its structure may have allowed for another problem. The ResNet backbone connect high-to-low resolution convolution layers in series giving the framework a low-resolution representation of the input. It then recovers the high-resolution representation from the encoded low-resolution (Wang, et al., 2020). This has the effect of trimming spatial information within the images that can have a significant effect to the performance of the detection model.

## 3.2 Approach

A possible solution approach that would attempt to overcome these thermal image low resolution issues could be a backbone structure introduced by (Wang, et al., 2020). It proposed a backbone described as the High-Resolution Network (HRNet). As shown in Figure 4, its unique structure manages to maintain a high-resolution representation of the input image as it connects high-to-low convolution streams in parallel. Used in the pixel-position-sensitive applications such as object detection this high-resolution approach should help in retaining more information within the thermal images so that better results can be achieved. Their multi-level representation was applied and tested to the object detection architectures of Faster R-CNN, Cascade R-CNN (Cai, 2018) with different backbone structures. Backbones extending ResNet and ResNeXt (He, 2017) with their different variations on the number of convolution layers used. With the results shown, their approach performed better than the other backbones as it had managed to achieve higher average precision.

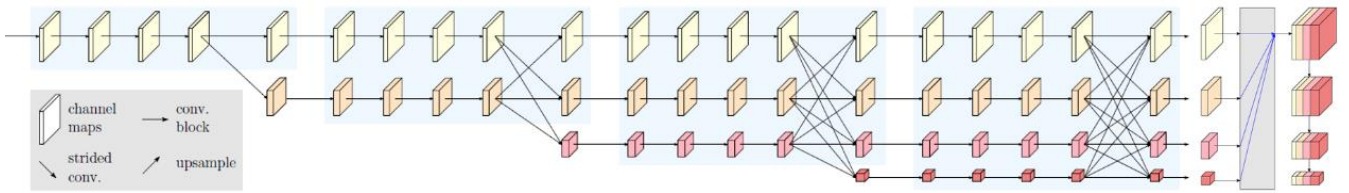


Figure 4. An example of a high-resolution network (Wang, et al., 2020)

Although the results shown by the High-Resolution backbone paper were based on models trained and tested on colour imaging data, it also indicates that it should be able to maintain more information in thermal imaging data. Comparing this HRNet backbone with a baseline such as the ResNet backbone, trained on a thermal dataset would show if this approach can transfer its benefits in the thermal imagery domain.

Maintaining an environment with the same parameters should be able to create an experiment that would clearly indicate how the performance varies between the baseline and the HRNet models used. Identifying performance improvements with this network of any value would be of huge importance as it means that it would provide a beneficiary impact in thermal detection specific applications and could build a new approach in the field.

## Chapter 4

# Experiments

### 4.1 Dataset

For this study the dataset that was used is the FLIR ADAS thermal dataset (Group, 2018) which consists of thermal and non-annotated RGB imagery. Containing over 9000 fully annotated thermal images taken with an IR Tau2 camera, some fully annotated example images are shown in Figure 5 The frames captured each with a resolution of 640x512 were taken in different conditions, with 60% of them being captured during the day, and the 40% at night.



Figure 5. Example thermal annotated images

The dataset contains 3 folders comprising of a training set, validation set and video set containing frames extracted from a 144 second video. For each of the 3 folders there is a structure of subfolders including 16-bit thermal images, 8-bit thermal

images, 8-bit thermal images with annotation boxes and RGB images. As well as an MSCOCO (Tsung-Yi, et al., 2014) formatted json annotation file which has the annotations for the objects present only in the thermal\_8\_bit folder. As per the dataset instructions, “annotations were created only for thermal images. The thermal and RGB camera did not have identical placement on the vehicle and therefore had different viewing geometries, so the thermal annotations do not represent the placement of objects in the RGB image.”

Since the dataset had already split the images into suggested sets, these were used for their specific purposes. The train thermal data was used as the training set for the model, the val thermal data was used as a validation set, and the video data was used as the testing set, to test the final performance of the fine-tuned models. As previously described the RGB images could not be used as the dataset does not contain their corresponding annotations. Additionally, the thermal-16-bit folders were also not used as there was no real purpose in using such high tonal value representations for these images.

The distribution of the annotated classes between the 3 main folders is shown by Figure 6. The dataset contains annotations for 4 classes, them being car, person, bicycle dog. As also shown in the figure there is a significant imbalance between the number of instances of each class within each set which could cause problems in the training and testing process. The train and validation set have a significant amount of more car and person annotations than the other 2 classes, especially on the dog class. Additionally, since the video set contained no instances of the dog class, it was decided that the dog category has insufficient data to be used. Thus, any dog annotations contained within the json files were removed during the pre-processing stage.

## Chapter 4: Experiments

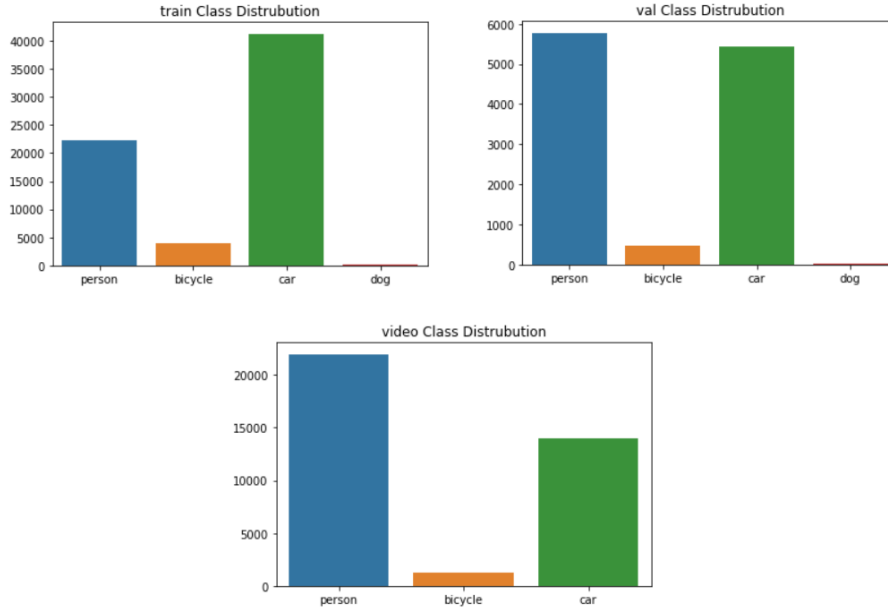


Figure 6. FLID ADAS annotation distribution between the 3 main folders

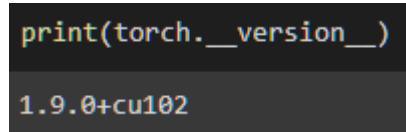
## 4.2 Environment

Training object detection models can be very computationally expensive. It was decided as to use a service provide by Google being Google Colaboratory. It is a free browser-based platform that provides the ability of executing arbitrary python code, in a notebook styled environment. It provides the ability of adding CPU (Central Processing Unit, GPU (Graphical Processing Unit), or TPU (Tensor Processing Unit) processing acceleration which can significantly cut down on processing time especially on computer-intensive processes such as model training. Most typical python libraries are already imported in Google Colab, but custom or version specific libraries need to be imported with every runtime execution of the environment.

## Chapter 4: Experiments

The main library used in this study for performing the training and testing of the models is MMCV (Contributors, 2018). The MMCV was build as a foundational library that supports many research projects with the one chosen for this research being MMDetection (Chen, 2019). Built by the OpenMMLab team, MMDetection is an open-source object detection toolbox that provides an extensive support on both contemporary and new detection frameworks. With its high benchmark efficiency, modular design, and plethora of pre-trained models it is believed that it is the most complete detection toolbox.

For setting up the Colab Notebook initially the MMDetection GitHub repository was cloned, and all the frameworks' dependencies where installed. Additionally, for installing up the proper MMCV version the correct CUDA (Sanders, 2010) version and Pytorch (Paszke, 2019) version needed to be identified. The versions used for this research are as shown in Figure 7, with Pytorch version 1.9.0 and CUDA version 10.2. These were used within to modify a URL link to download and install the corresponding MMCV version.



```
print(torch.__version__)  
1.9.0+cu102
```

Figure 7. Environment library versions used, Pytorch 1.9.0 | CUDA 10.2

## 4.3 Dataset Pre-Processing

As previously described the FLIR ADAS thermal dataset provides for each folder a corresponding annotation file in the MSCOCO format. Before it could have been used with the MMDetection toolbox some minor changes needed to be made to these annotation files.

Each file contained in its categories object all 80 categories that the COCO dataset (Tsung-Yi, et al., 2014) uses. These needed be trimmed down leaving only the 3 categories of person, bicycle, and car that the dataset contains.

Furthermore, since it was decided that the dog category annotation instances would not be used these had to be removed from the annotations object within the json files. Additionally, because some file paths within the images object where not actually present in their thermal image folders and where causing problems these needed to be identified and removed.

Lastly, the file\_name fields within the images object as shown in Figure 8 were edited as to only include the image file name and not both the folder path and image file. With all changes applied new annotations json files were created and those where the ones used in the experimental process.



```
"images": [
  {
    "extra_info": {},
    "subdirs": ".",
    "id": 0,
    "width": 640,
    "file_name": "thermal_8_bit/FLIR_00001.jpeg",
    "height": 512
  },
  {
    "extra_info": {},
    "subdirs": ".",
    "id": 1,
    "width": 640,
    "file_name": "thermal_8_bit/FLIR_00002.jpeg",
    "height": 512
  },
],
```

```
"images": [
  {
    "extra_info": {},
    "subdirs": ".",
    "id": 0,
    "width": 640,
    "file_name": "FLIR_00001.jpeg",
    "height": 512
  },
  {
    "extra_info": {},
    "subdirs": ".",
    "id": 1,
    "width": 640,
    "file_name": "FLIR_00002.jpeg",
    "height": 512
  },
],
```

Figure 8. Pre-processing step. Editing image file\_name fields

## 4.4 Models

For the projects experimental process, it was decided that 4 models will be trained. One of the models will be used as a baseline method that the results of the other 3 models will be compared to. All of 4 tested models were chosen from MMDetection’s Model Zoo, they were all pre-trained on the COCO dataset with same learning schedule of 1x. They all inherit the same Faster R-CNN architecture, with the same Region Proposal Network (RPN) and the same Region of Interest (ROI) head. RPN and ROI are both components of the Faster RCNN architecture that were an adaptation on top of the previous Fast R-CNN architecture.

### 4.4.1 Baseline: Faster R-CNN ResNet-50 FPN

Faster RCNN with the ResNet-50 backbone with a Feature Pyramid Network (FPN) was used as a baseline model. ResNet-50 is a variant of the ResNet model that uses 48 convolution layers, 1 maximum pooling layer and 1 average pooling layer with a combined 50 convolution layers hence the name. The FPN first shown in 2017 in the work by (Lin, 2017) is a feature extraction method that generates multiple feature map layers in a pyramid like structure and helps maintain semantic value between layers. FPN is built within the Region Proposal Network and has been proven to greatly improve inference times. This baseline model will be referred as the ResNet50 model.

### 4.4.2 Faster R-CNN HRNetv2p Models

For comparing the High-Resolution network with the baseline method 3 HRNet models with slight variations were used. To create the 3 different models, different values for the width of the high-resolution convolution were used. The first one is the Faster R-CNN HRNetV2p-W18 model, 18 representing the width of the



high-resolution network convolution. The second one is the Faster R-CNN HRNetV2p-W32 with a width of 32, and lastly Faster R-CNN HRNetV2p-W40 with a width of 40. These models will now be referred to as HRNetw18, HRNetw32 and HRNetw40 respectively.

For creating the models initially, they inherit the same base structure as in the ResNet50 model. Then the backbone and neck of the model is modified with the configurations of the HRNet models obtained by the model zoo with their High-Resolution network structure. Figure 9 which the modifications that were applied to the baseline model to create the HRNetw32 model.

```
_base_ = './faster_rcnn/faster_rcnn_r50_fpn_1x_coco.py'
model = dict(
    backbone=dict(
        _delete_=True,
        type='HRNet',
        extra=dict(
            stage1=dict(
                num_modules=1,
                num_branches=1,
                block='BOTTLENECK',
                num_blocks=(4, ),
                num_channels=(64, )),
            stage2=dict(
                num_modules=1,
                num_branches=2,
                block='BASIC',
                num_blocks=(4, 4),
                num_channels=(32, 64)),
            stage3=dict(
                num_modules=4,
                num_branches=3,
                block='BASIC',
                num_blocks=(4, 4, 4),
                num_channels=(32, 64, 128)),
            stage4=dict(
                num_modules=3,
                num_branches=4,
                block='BASIC',
                num_blocks=(4, 4, 4, 4),
                num_channels=(32, 64, 128, 256))),
        init_cfg=dict(
            type='Pretrained', checkpoint='open-mmlab://msra/hrnetv2_w32')),
    neck=dict(
        _delete_=True,
        type='HRFPN',
        in_channels=[32, 64, 128, 256],
        out_channels=256))
```

Figure 9. faster\_rcnn\_hrnetv2p\_w32\_1x\_coco configuration file

For creating the other 2 HRNet model configurations the process follows a similar way. The models inherit everything from the HRNetv32 model, but then the values in the different stages in the backbone are changed, as well as the pre-trained checkpoint in the initial configuration of the backbone and the values of the input channels in the neck. Shown in Figure 10 is the configuration for creating the HRNetv18 model and in Figure 11 is the configuration for creating the HRnet40 model.

```
_base_ = './faster_rcnn_hrnetv2p_w32_1x_coco.py'
model = dict(
  backbone=dict(
    extra=dict(
      stage2=dict(num_channels=(18, 36)),
      stage3=dict(num_channels=(18, 36, 72)),
      stage4=dict(num_channels=(18, 36, 72, 144)),
    ),
    init_cfg=dict(
      type='Pretrained', checkpoint='open-mmlab://msra/hrnetv2_w18'),
  ),
  neck=dict(type='HRFPN', in_channels=[18, 36, 72, 144], out_channels=256))
```

Figure 10. faster\_rcnn\_hrnetv2p\_w18\_1x\_coco configuration file

```
_base_ = './faster_rcnn_hrnetv2p_w32_1x_coco.py'
model = dict(
  backbone=dict(
    extra=dict(
      stage2=dict(num_channels=(40, 80)),
      stage3=dict(num_channels=(40, 80, 160)),
      stage4=dict(num_channels=(40, 80, 160, 320)),
    ),
    init_cfg=dict(
      type='Pretrained', checkpoint='open-mmlab://msra/hrnetv2_w40'),
  ),
  neck=dict(type='HRFPN', in_channels=[40, 80, 160, 320], out_channels=256))
```

Figure 11. faster\_rcnn\_hrnetv2p\_w40\_1x\_coco configuration file

## 4.5 Data Pipelines and Parameters

To create a balanced training environment the same data augmentations methods and data parameters were used for all the tested models. As previously described the FLIR ADAS thermal dataset annotations are in the MSCOCO format, thus the `dataset_type` used in the runtime configurations was ‘CocoDataset’.

Furthermore, there are the data augmentation methods that apply to both the training and test data. First, step in the data augmentation methods is the image scale resizing. This was altered from the default values provided of 1333x800 to 800x800. The first value is the highest indicates the maximum value a side of an image can have, and the second value is the lowest. Since the thermal images of the dataset all have the same resolution of 640x512 it would make sense to lower the minimum value to match the image data resolutions. With further investigation it was chosen not to change the minimum value per implementation details of the paper that introduced FPNs, the feature extractor was developed and tested on image data with a minimum value of 800. This had to be properly accounted for as the baseline ResNet50 model does use FPNs.

The second step of the data augmentation methods is a probabilistic random horizontal flipping of an image. This method was developed as to apply some additional variation within the dataset and to help the model generalize better on unseen data. Then, there is image normalization used in different configuration values also shown on the 2<sup>nd</sup> line in Figure 12, and lastly there is the padding of an image set to a divisor size of 32. The data pipelines used for this study can be seen in Figure 12.

## Chapter 4: Experiments

```
# Dataset settings
dataset_type = 'CocoDataset'
data_root = ''
img_norm_cfg = dict(mean=[123.675, 116.28, 103.53], std=[58.395, 57.12, 57.375], to_rgb=True)

train_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(type='LoadAnnotations', with_bbox=True),
    dict(type='Resize', img_scale=(800, 800), keep_ratio=True),
    dict(type='RandomFlip', flip_ratio=0.5),
    dict(type='Normalize', **img_norm_cfg),
    dict(type='Pad', size_divisor=32),
    dict(type='DefaultFormatBundle'),
    dict(type='Collect', keys=['img', 'gt_bboxes', 'gt_labels']),
]
test_pipeline = [
    dict(type='LoadImageFromFile'),
    dict(
        type='MultiScaleFlipAug',
        img_scale=(800, 800),
        flip=False,
        transforms=[
            dict(type='Resize', keep_ratio=True),
            dict(type='RandomFlip'),
            dict(type='Normalize', **img_norm_cfg),
            dict(type='Pad', size_divisor=32),
            dict(type='ImageToTensor', keys=['img']),
            dict(type='Collect', keys=['img']),
        ]
    )
]
```

Figure 12. Train and Test data pipelines

Paths for the appropriate image data sets such as training, validation and testing were modified accordingly, with a batch size of 4 images per iteration and 2 workers, the data parameter changes made can all be seen in Figure 13.

```
data = dict(
    samples_per_gpu=4,
    workers_per_gpu=2,
    train=dict(
        type=dataset_type,
        ann_file=data_root + '../train/thermal_annotations_v2.json',
        img_prefix=data_root + '../train/thermal_8_bit/',
        pipeline=train_pipeline),
    val=dict(
        type=dataset_type,
        ann_file=data_root + '../val/thermal_annotations_v2.json',
        img_prefix=data_root + '../val/thermal_8_bit/',
        pipeline=test_pipeline),
    test=dict(
        type=dataset_type,
        ann_file=data_root + '../video/thermal_annotations_v2.json',
        img_prefix=data_root + '../video/thermal_8_bit/',
        pipeline=test_pipeline))
```

Figure 13. Data configuration

## 4.6 Hyperparameters

Some hyperparameters on the runtime training configurations needed to be changed from the default one provided by MMDetection/configs/\_base\_ directory. An example with these changes applied are shown in Figure 14, which are the hyperparameters used to train the Resnet50 model. Differences between the hyperparameters between the 4 models were only made on the path of their corresponding working directory and the link to their pre-trained model weights.

```
runner = dict(type='EpochBasedRunner', max_epochs=12)
total_epochs = 12
checkpoint_config = dict(interval=1)
work_dir = '../models/ResNet50'

optimizer = dict(type='SGD', lr=0.0025, momentum=0.9, weight_decay=0.0001)
optimizer_config = dict(grad_clip=None)

lr_config = dict(
    policy='step',
    warmup='linear',
    warmup_iters=500,
    warmup_ratio=0.001,
    step=[7, 9, 11])

log_level = 'INFO'
log_config = dict(
    interval=100,
    hooks=[
        dict(type='TextLoggerHook'),
        dict(type='TensorboardLoggerHook')
    ])

load_from = 'https://download.openmmlab.com/mmdetection/v2.0/faster_rcnn/' \
            'faster_rcnn_r50_fpn_1x_coco/faster_rcnn_r50_fpn_1x_coco_20200130-047c8118.pth'

resume_from = None

workflow = [('train', 1), ('val', 1)]
evaluation = dict(interval=1, metric='bbox', classwise=True)

custom_hooks = [dict(type='NumClassCheckHook')]
dist_params = dict(backend='nccl')
```

Figure 14. Faster\_RCNN\_ReNet-50\_fpn\_1x hyperparameter configuration

Further discussion on the value of the hyperparameters used in comparison to the default ones provided by MMDetection’s runtime configuration will be made. Firstly, the Stochastic gradient descent (SGD) (Bottou, 2012) optimizer was kept the same as well as the value for momentum of 0.9 and weight decay of 0.0001. The value that was altered was the learning rate. As it was described in MMDetection’s documentation each model was pre-trained on the same configuration of 8 GPUs with a set amount of 0.2 on the learning rate. As per HRNet’s Training instructions found in their GitHub repository “learning rate should be adjusted when the number of GPUs is changed” (Xiao, 2019). Since the Google Colab environment only uses 1 GPU the learning rate was divided by 8 and the new value of 0.0025 was used.

Furthermore, for the learning configuration, the same step policy and linear warmup configuration were used. The only changes applied were on the steps that learning rate changes would occur. It was altered from reducing the learning rate by a factor of 0.1 at epochs 8 and 11 to now applying the changes at epoch 7, 9, and 11. This was intended as to smooth out the converging learning process of the models nearing the end of the total number of epochs used, which were 12.

The final changes made were on the workflow of the training process. Previously the default configuration only contained training within its workflow. As to obtain additional information during the training process a validation step was also added as to be executed at the end of each epoch. Since the dataset used was of type COCO the evaluation metric that could have been used was ‘bbox’ with the added class-wise evaluation as to provide the additional average precision for each class at every validation step.

The 12 metrics that will be obtained in the evaluation step to characterize the performance of the trained model are the common COCO metrics (Tsung-Yi, et al., 2014). This includes metrics based on the average precision, which measures how

accurate the models of the prediction. To find the precision the rate of true positives over the true positives + false positives are measured. What defines true positive, and a false positive is the Intersection over Union (IoU) threshold used. Shown also in Figure 15 IoU measures the area of overlap over the area of union on the prediction boxes and the ground truth. If for example a threshold of 0.5 was used, the IoU needs to exceed 0.5 for it to be accounted as a true positive, else it is a false positive.

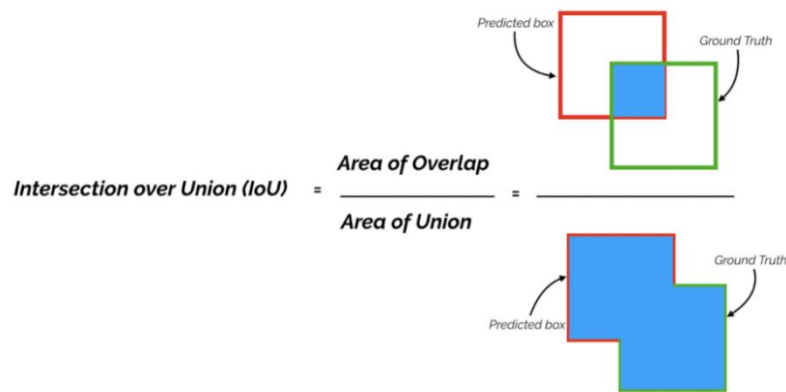


Figure 15. Intersection over Union example  
<https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2> [Accessed on 1 Sep 2021]

There are 6 metrics on average precision, with 2 of them being just measuring the average precision at different thresholds. Average precision with threshold 0.5 is described as the PASCAL VOC metric, one of the most common metrics used, and the other is the strict metric which has a threshold of 0.75. Another average precision metric described as the COCO metric, or else the primary challenge metric, which uses a 10 incremental IoU thresholds from 0.5 to 0.95 and takes the average between them. This is much different from the other methods but it rewards detectors with better localization. Furthermore, there are the average precision metrics at different scales. The scales are described as small, medium, and large. With small

## Chapter 4: Experiments

corresponding to objects with area less than 32 squared pixels, medium being between 32 and 96 squared pixels, and finally large with any object with a larger area of 96 squared pixels. These metrics show how the model performs in detecting different sized object during the validation process. Average precision is referred as the metric obtained for each class. The metric that shows an average of the precision on the dataset is referred as the mean average precision or mAP.

Additionally, there are 6 recall metrics, recall being how well all the positives have been found. It is measured by finding the rate of True positives over the number of true positives and false negatives. These metrics could have provided additional information of the models training processes, but they were not used further though the study.



## 4.7 Training and Validation Results

All 4 models were then trained on the Google Colab GPU accelerated environment with the training and validation results stored in each model's working directory. The results were stored in TensorFlow's (Derek, 2016) event styled files which were visualized using TensorBoard, TensorFlow's visualization toolkit.

For each of the graph metrics that will be presented the x-axis represents the step counter. Step counter contains each iteration for each epoch during the training process. And the y-axis represents the value of the observed metric. The graphs that will be shown in this section had a 0.8 smoothing as to make it easier to distinguish and compare the converging results. The fainter lines behind them are the actual line graphs at their recorded values.

The models shown in each of the following figures will be represented by a different colour. As shown in Figure 16 these colours are to be used as keys for all the metrics graphs that will be presented. With the ResNet50 model taking an orange colour representation, HRNetw18 with a dark blue, HRNetw32 with a red colour, and HRNetw40 with a light blue.

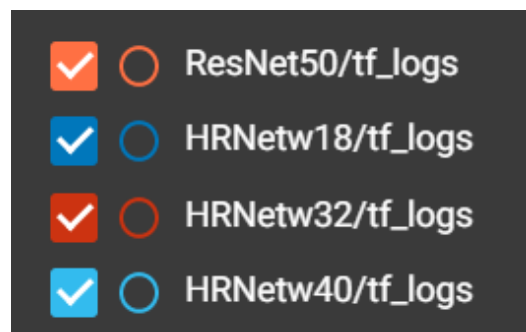


Figure 16. Corresponding colours for each trained model.

Represented by Figure 17 are the results of the training loss obtained by the models. It can be observed that throughout the whole training process the HRNetw40 model had the lowest training loss, with HRNetw32 being closely second. This indicated that those 2 models were able to make more accurate predictions on the training data in comparison to the other 2. Furthermore, the training loss has been shown to even out for all 4 models after about 17 thousand steps.

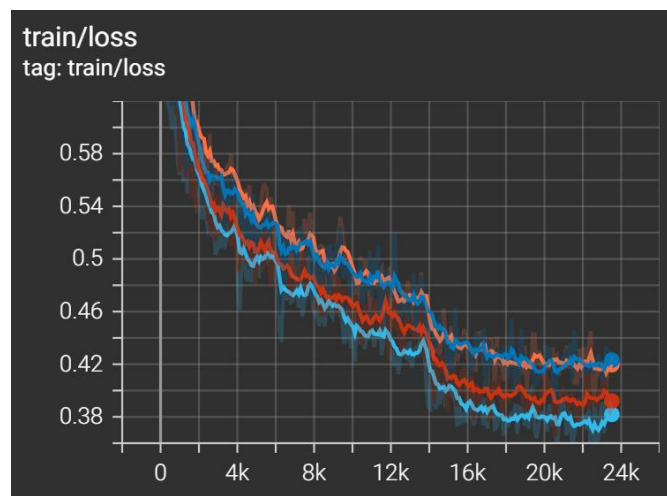


Figure 17. Training loss metric for all 4 runs

This was also shown to be the case for all the other training losses that were recorded. Shown in Figure 18 there are 4 graphs. Upper left: bounding box loss. Upper right: classification loss. Bottom left: RPN bounding box loss. Bottom right: RPN classification loss. Again, the HRNetw40 and HRNetw32 models had the lowest loss values for all 4 metrics throughout training, they were able to make better bounding box predictions having a lower regression loss or L1 loss, and better classifications measured in the cross-entropy loss. As well as better results for the regression and classification of their region proposal networks.

## Chapter 4: Experiments

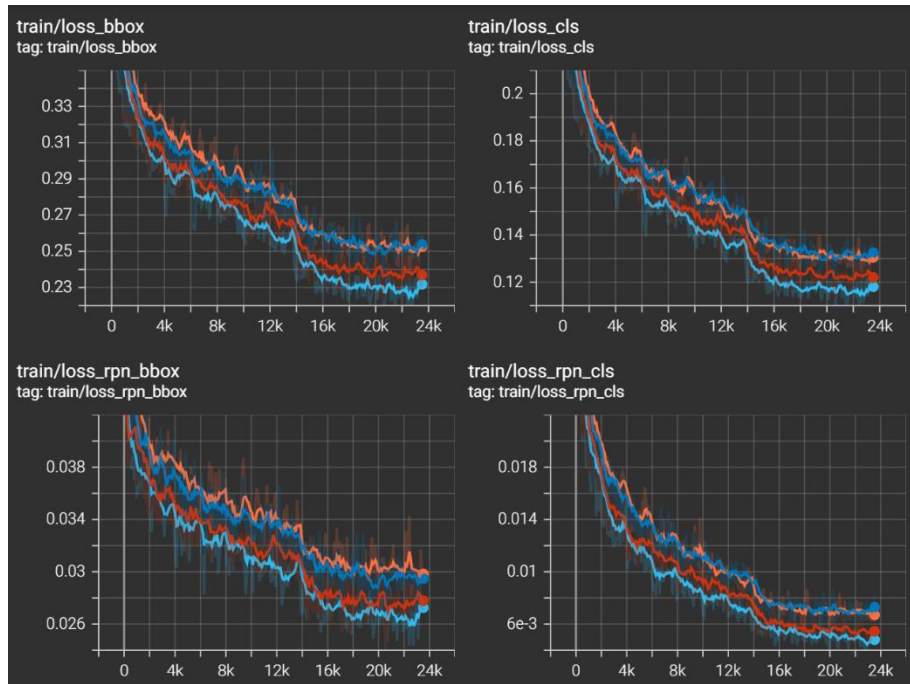


Figure 18. Additional training metrics

Upper left: bounding box loss. Upper right: classification loss.

Bottom left: RPN bounding box loss. Bottom right: RPN classification loss.

As validation was also added in the workflow of the training process at end of each epoch validation metric results were also recorded. These include the validation loss shown in Figure 19 exhibits the model performances on unseen data. The results where a bit different from training, as the HRNetw18 regardless of its previous training metrics had managed to surpass the HRNetw32 model. After the 15 thousandth step mark it had achieved better validation loss results. In the end of the training process, it was able to generalize better and making more accurate predictions on the unseen data.

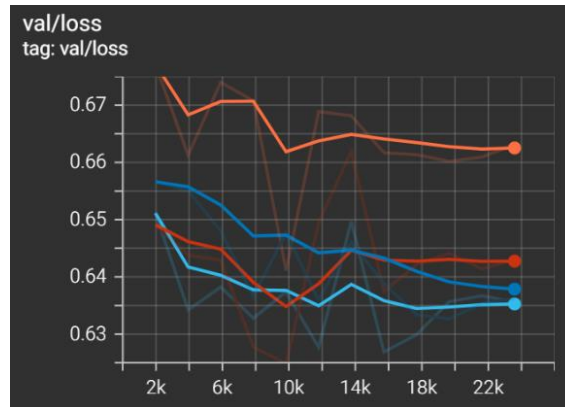


Figure 19. Validation Loss metric for all 4 runs

In Figure 20 there are again the other previously mention metrics, but with their performance now shown on the validation set. Upper left: bounding box loss. Upper right: classification loss. Bottom left: RPN bounding box loss. Bottom right: RPN classification loss. For the bounding box loss, the performance sequence between the models matches the validation loss. This includes both bounding box loss and RPN bounding box loss, with HRNetw40 having the least loss and second being HRNetw18. This is different with validation's classification loss as it is observed that the best performing one was HRNetw18. As for RPN classification loss all 3 HRNet model were relatively close, and it was noticed that the ResNet model was lacking very much behind.

## Chapter 4: Experiments

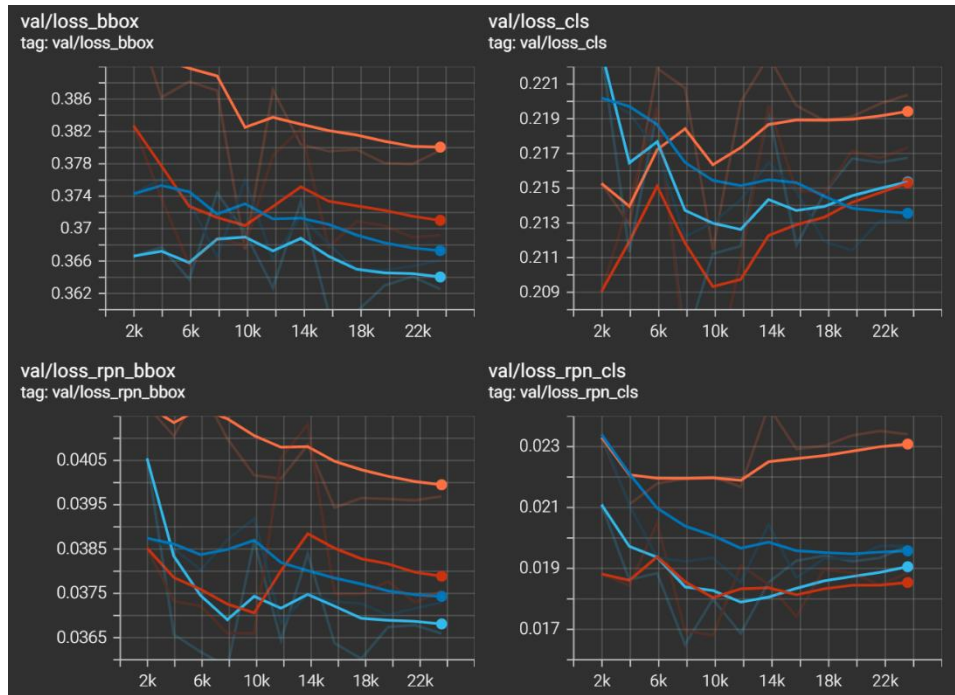


Figure 20. Additional validation metrics

Upper left: bounding box loss. Upper right: classification loss.

Bottom left: RPN bounding box loss. Bottom right: RPN classification loss.

As previously described in the hyperparameters experimental section, the bbox COCO evaluation metric was also set to be recorded. Shown in Figure 21 there are the recordings of the mean average precisions made on the COCO primary challenge metric. With the results obtained it was clear that over the 12 epochs the HRNetw40 had the best results as it got the best overall recordings of its average precision. With HRNetw18 with the second-best performance, HRNet-w32 with the third and last being ResNet-50. The High-resolution network models seemed to make better localized predictions on the validation set.

## Chapter 4: Experiments

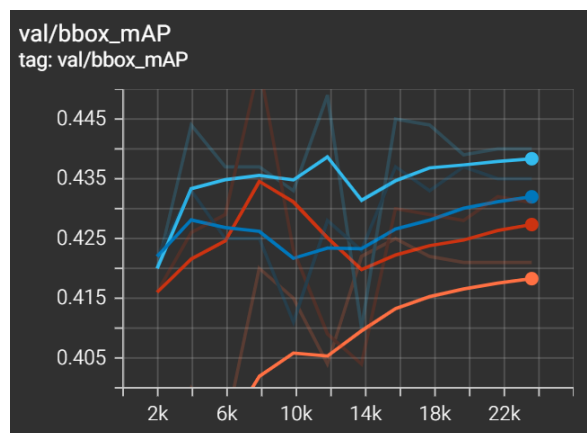


Figure 21. Average Precision % at IoU=.50:.05:.95 (primary challenge metric)

Average precision was also obtained at IoU thresholds of 0.5 and 0.75. As shown in Figure 22, Left: Average Precision % at IoU=.50, Right: Average Precision % at IoU=.75. It followed the exact same sequence of performance between the models as in the previous Figure 21 that was described.

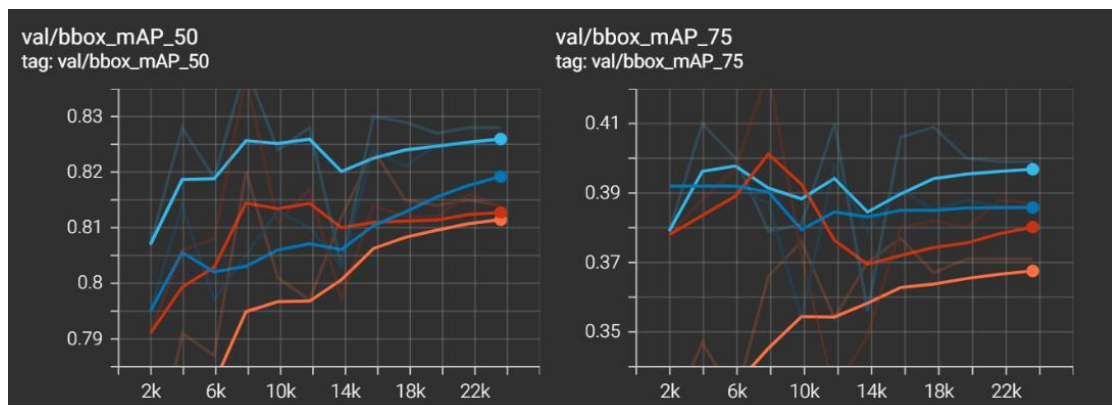


Figure 22. Left: Average Precision % at IoU=.50 (PASCAL VOC metric)  
Right: Average Precision % at IoU=.75 (strict metric)

## Chapter 4: Experiments

Average precision across scales where also recorded. Shown in the following Figure 23 are the 3 graphs. Left: Average Precision % for small objects. Middle: Average Precision % for medium objects. Right: Average Precision % for large objects. From this figure it can be recognized that the HRNetw40 was by far the best at making predictions on medium sized objects. For the small and large objects, the performance between the model was varying throughout the training process, but it can clearly be seen that the ResNet50 model had the worst performance of all.

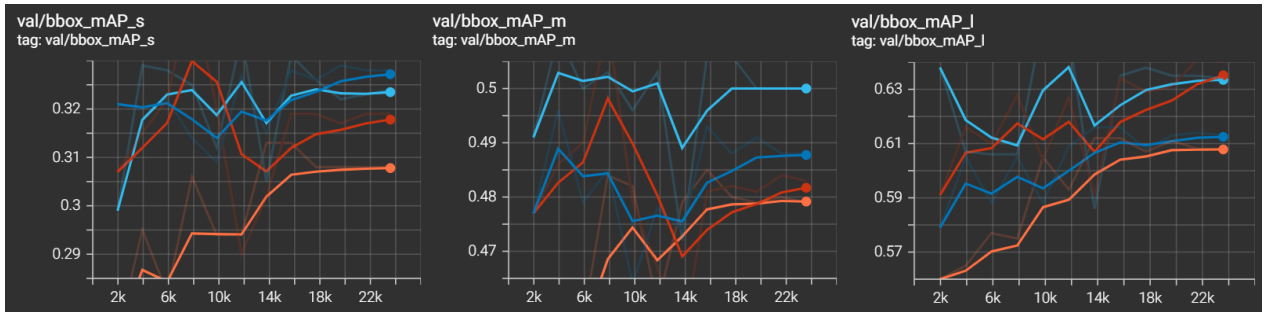


Figure 23. Average Precision Across Scales validation results  
Left: Average Precision % for small objects: area < 32<sup>2</sup>  
Middle: Average Precision % for medium objects: 32<sup>2</sup> < area < 96<sup>2</sup>  
Right: Average Precision % for large objects: area > 96<sup>2</sup>

## 4.8 Testing Results

At the end of each training/validation epoch the state of the trained model was saved as a checkpoint to the working directory. For the testing process of the experiment the best performing saved epoch for each model was to be chosen. As performance was greatly varying on the training and validation results the method of deciding on a best model choice of each model was some what of a manual way of adapting early stopping. Early stopping is a form of regularization method that is used to stop the model from overtraining as such the model does not overfit on the training data and lose its generalization abilities on other unseen data.

As for the values to be accounted for to indicate what would be defined as the best model 2 metrics were taken into consideration, bbox\_mAP (mAP at IoU=.50:.05:.95) referred as simply mAP and bbox\_mAP\_50 (mAP at IoU=.50) referred as mAP\_50. The checkpoint with the highest collective values for those 2 metrics and with sufficient training was chosen as the best one and was further used in the testing process. On the following graphs no smoothing was applied thus the line graphs shown follows the exact recorded measurements.

In Figure 24 there are the results of the average precision metrics on the ResNet50 model. Within these results the best values obtained were at step 15.72K, with mAP of 0.425 and mAP\_50 at 0.824, which corresponded to the model saved at the end of the 8<sup>th</sup> epoch.



## Chapter 4: Experiments

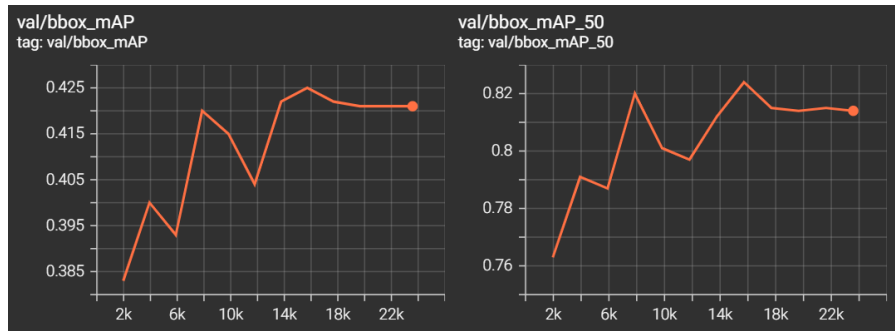


Figure 24. ResNet50 mAP and mAP\_50 results

Results for the HRNetw18 model are shown in Figure 25. At steps 15.72K and 19.65K mAP had the same value of 0.437. Compared to the mAP\_50 metric at 15.72K steps it had a value of 0.824, but on the 19.65K step it had a value of 0.825. Although the difference was minimal step 19.65K corresponding to the 10<sup>th</sup> epoch was chosen as to being the best one.

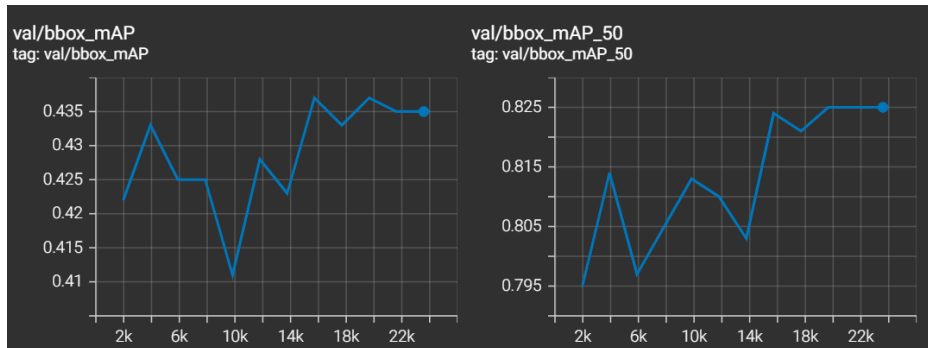


Figure 25. HRNetw18 mAP and mAP\_50 results

For the HRNetw32 model the results obtained are represented by Figure 26. Within these results something odd had occurred. Close to the 8 thousandth step specifically at the 4<sup>th</sup> epoch the best values for the 2 metrics were recorded. This was very strange as it was very early in the training process and both the validation and training loss still seemed to be improving even after that step. For this reason, those

## Chapter 4: Experiments

specific values were accounted as outliers thus the 2<sup>nd</sup> best values were used for the final choice. These were at epoch 11 with a mAP value of 0.432 and mAP\_50 value of 0.816.

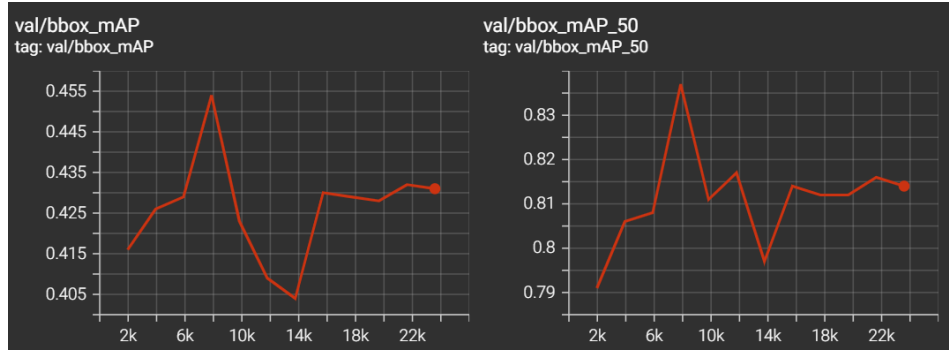


Figure 26. HRNet32 mAP and mAP\_50 results

Lastly, for the model HRNet40 model with the results shown in Figure 27 the best epoch was chosen based on the combined performances of the average precision metrics. At step 7.681K the mAP\_50 had the highest value of 0.839 whereas mAP had only reached a 0.437. Furthermore, a similar thing occurred where at the 11.79K step a high value of mAP was recorded but a smaller one for mAP\_50. For this reason, these 2 points weren't presumed to be the best ones. Thus, at the 15.72K step (epoch 8) it was decided that since both metrics were relatively high that checkpoint was presumed as the best one.

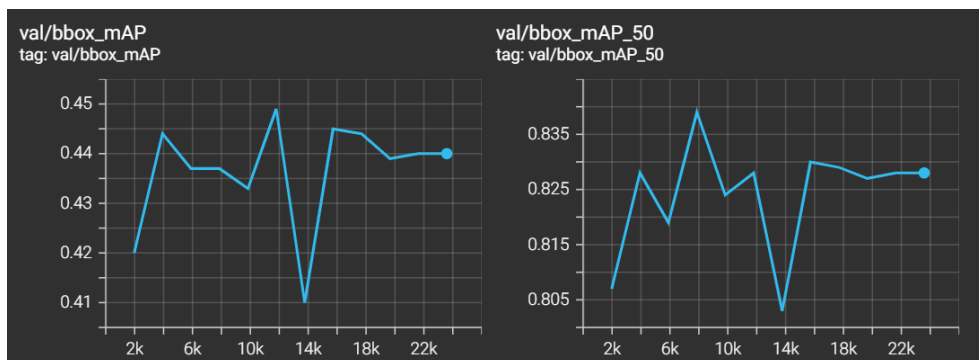


Figure 27. HRNet40 mAP and mAP\_50 results

With the best epochs for each model chosen these were then tested on the FLIR ADAS thermal video set. The images extracted from the 144 second video were never seen by the models and were used to properly qualify the performance of the final models. The test results obtained are presented by Table 1, which shows for each model the average precision metrics obtained for each class as well as the average between all the classes (AP at IoU=.50:.05:.95).

Faster R-CNN					
backbone	Learning schedule	AP across each class			mAP
		Person	Bicycle	Car	
ResNet-50-FPN	1X	0.121	0.194	0.507	0.274
HRNetV2p-W18	1X	0.132	0.179	0.519	0.277
HRNetV2p-W32	1X	0.14	0.2	0.524	0.288
<b>HRNetV2p-W40</b>	<b>1X</b>	<b>0.146</b>	<b>0.205</b>	<b>0.531</b>	<b>0.294</b>

Table 1. Performance Comparison of the 4 tested models

Since the ResNet50 model was used as the baseline approach the mean average precision results for the other models were compared between it as to obtain the percentage difference. As shown in Table 2, all the HRNet models had an improvement over the baseline method. With HRNetw40 having over a 7% performance increase. It proved that HRNet can be considered a good backbone candidate even on the thermal image detection domain.

Backbone	mAP Percentage Difference
ResNet-50-FPN	0%
HRNetV2p-W18	1.09%
HRNetV2p-W32	4.98%
<b>HRNetV2p-W40</b>	<b>7.04%</b>

Table 2. mAP percentage difference of the models compared to the ResNet50 baseline

# Chapter 5

## Conclusions

### 5.1 Project Summary

This project aimed to compare the detection results of the High-resolution network backbone and a common object detection baseline model. From the literature review research that was performed it was identified that thermal imaging data lack information due to their low resolutions. Thus, additional methods had to be adapted to extract more semantic information as to be used in the position-sensitive application of object detection.

Through the methodology and experimental sections, it was described how the training and testing these different backbones will aid to prove how their performance differs. Maintaining the same environment through the training process the results obtained are valid and are properly compared.

With the testing results that had been previously presented by Tables 1 and 2 it was shown that the HRNet backbone outperformed the ResNet50 baseline method. In fact, all 3 High-Resolution network models that were tested had an improvement over the baseline. With the HRNetw40 model (40 being the width of the high-resolution convolution) having over a 7% improvement.

This showed that this backbone approach is suitable to be used in thermal imaging object detection applications. It can provide a beneficiary effect to the detection performance of a model, proving this to be a solution from the gaps identified through the literature review.

## 5.2 Future Work

Future work suggestions are listed below. Proposing additional testing methods on the High-Resolution network as to further expand the backbone's research on thermal imaging data.

- Introduce baselines with higher convolution layer numbers e.g. (ResNet 101, ResNet152) to be compared with the tested HRNet models. From the work proposed (Xiao, 2019), different model widths were compared between baselines with higher convolution layer numbers.
- Investigate inference speed between the models. As different applications have different requirements, with some focused-on speed and others focus on accuracy, more appropriate model applications can be identified.
- Implementing a different thermal imaging dataset, possible differences could be on the number of classes contained within the dataset as to observe if the models can generalize better on higher number of objects. Additionally, using higher thermal image resolutions as to observe if the High-Resolution networks can perform better by retaining more information during the encoding process.
- Investigating model architectures other than Faster R-CNN and testing their performance on the same data and parameters as to compare their differences.

## 5.3 Reflections

Training the different models. The initial approach was to only use the Pytorch library and follow its corresponding fine-tuning methods for training and testing all the models needed. Unfortunately, only the ResNet50 baseline was able to be trained on Pytorch as there was no method of implementing the High-Resolution network models with their pre-trained weights.

After finding the MMDetection library that supported all the models needed for this study's experiment, using it proved much more difficult than expected. Training the models was very strenuous as Google Colab resource limitations were a significant hindrance. Usually after 4-5 hours of training the runtime disconnected with an "Exceeded Usage Limit" warning preventing reconnection the runtime with a GPU. This usage limit usually lasted up to 2 days causing a significant amount of down time.

Furthermore, attempting to use my personal Windows 10 device proved futile. Although I was able to test everything on the Pytorch implementation there was a significant issue specifically on MMDetection. Given my device's hardware configuration there was no way of importing the appropriate MMCV library version. With an RTX-3070 graphics card the pytorch library required CUDA capability of sm\_86 which is given in versions higher than Pytorch 1.6.0. Additionally, the MMCV library with CUDA capabilities was not supported on Windows 10 on any version of pytorch higher than 1.6.0. This meant that there was no way of using MMDetection with my specific setup.

The only approach that proved to work was upgrading to Google Colab Pro. It had given the ability of training without any disconnecting issues, whilst also using a much faster GPU with even longer runtimes.

# Bibliography

Berg, A., Ahlberg, J. & Felsberg, M., 2015. A Thermal Object Tracking Benchmark. In: *Proceedings of the 12th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. Karlsruhe: IEEE, pp. 1-6.

Bhattarai, M. & MartíNez-Ramón, M., 2020. A Deep Learning Framework for Detection of Targets in Thermal Images to Improve Firefighting. *IEEE Access*, Volume 8, pp. 88308-88321.

Bottou, L., 2012. Stochastic gradient descent tricks. In: *Neural networks: Tricks of the trade*. s.l.:Springer, pp. 421-436.

Cai, Z. a. V. N., 2018. Cascade R-CNN: Delving Into High Quality Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. s.l.:s.n.

Chen, K. a. W. J. a. P. J. a. C. Y. a. X. Y. a. L. X. a. S. S. a. F. W. a. L. Z. a. X. J. a. o., 2019. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*.

Contributors, M., 2018. *MMCV: OpenMMLab: Computer Vision Foundation*. [Online] Available at: <https://github.com/open-mmlab/mmcv> [Accessed 1 September 2021].

Deng, Z. et al., 2018. *R3net: Recurrent residual refinement network for saliency detection..* Stockholm, AAAI Press.

Derek, M. A. a. P. B. a. J. C. a. Z. C. a. A. D. a. J. D. a. M. D. a. S. G. a. G. I. a. M. I. a. M. K. a. J. L. a. R. M. a. S. M. a., 2016. Tensorflow: A system for large-scale machine learning. In: *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*. s.l.:USENIX Association, pp. 265-283.

Devaguptapu, C., Akolekar, N., Sharma, M. M. & Balasubramanian, V. N., 2019. Borrow From Anywhere: Pseudo Multi-Modal Object Detection in Thermal Imagery. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*. Long Beach, California: IEEE, pp. pp. 0-0.

Duan, K. a. B. S. a. X. L. a. Q. H. a. H. Q. a. T. Q., 2019. CenterNet: Keypoint Triplets for Object Detection. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. s.l.:s.n.

## Bibliography

- Everingham, M. a. J. W., 2009. *The PASCAL visual object classes challenge 2007 (VOC2007) development kit.*, s.l.: Citeseer.
- Felsberg, M. a. K. M. a. M. J. a. L. A. a. P. R. a. H. G. a. B. A. a. E. A. a. A. J. a. Č. Z. L. a. V. T. a. L. A. a. F. D., 2016. The Thermal Infrared Visual Object Tracking VOT-TIR2016 Challenge Results. Volume 9914, pp. 824-849.
- Ghose, D. et al., 2019. *Pedestrian Detection in Thermal Images using Saliency Maps*. Long Beach, California, IEEE.
- Girshick, R. a. D. J. a. D. T. a. M. J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. s.l.:s.n., pp. 580--587.
- Group, F. A., 2018. *flir*. [Online] Available at: <https://www.flir.in/oem/adas/adas-dataset-form/> [Accessed 8 May 2021].
- He, K. a. Z. X. a. R. S. a. S. J., 2016. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. s.l.:s.n.
- Herrmann, C., Ruf, M. & Beyerer, J., 2018. CNN-based thermal infrared person detection by domain adaptation. In: M. C. Dudzik & J. C. Ricklin, eds. *Autonomous Systems: Sensors, Vehicles, Security, and the Internet of Everything*. s.l.:SPIE, pp. 38-43.
- He, S. X. a. R. G. a. P. D. a. Z. T. a. K., 2017. Aggregated Residual Transformations for Deep Neural Networks. In: *In Proceedings of the IEEE conference on computer vision and pattern recognition*. s.l.:s.n., pp. 1492-1500.
- Hwang, S. et al., 2015. *Multispectral Pedestrian Detection: Benchmark Dataset and Baseline*. Boston, IEEE.
- John, V., Liu, Z., Mita, S. & Qi, B., 2015. Pedestrian detection in thermal images using adaptive fuzzy C-means clustering and convolutional neural networks. In: *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*. Tokyo: IEEE, pp. 246-249.
- Joseph, R. & Ali, F., 2018. *YOLOv3: An Incremental Improvement*, s.l.: s.n.
- Karen, S. & Andrew, Z., 2015. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [Online] Available at: <https://arxiv.org/abs/1409.1556> [Accessed 2021].
- Krišto, M., Ivacic-Kos, M. & POBAR, M., 2020. Thermal Object Detection in Difficult Weather Conditions Using YOLO. *IEEE Access*, Volume 8, pp. 125459-125476.
- Krizhevsky, A., Sutskever, I. & Hinton, G. E., 2012. *ImageNet Classification with Deep Convolutional Neural Networks*. Red Hook, NY, Curran Associates Inc..



## Bibliography

Li, M., Zhao, X., Li, J. & Nan, L., 2020. ComNet: Combinational Neural Network for Object Detection in UAV-Borne Thermal Images. *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1-12.

Lin, T.-Y. a. D. P. a. G. R. a. H. K. a. H. B. a. B. S., 2017. Feature Pyramid Networks for Object Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. s.l.:s.n.

Liu, N. a. H. J. a. Y. M.-H., 2018. PiCANet: Learning Pixel-Wise Contextual Attention for Saliency Detection. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Salt Lake City: s.n.

Liu, Q. et al., 2017. Deep convolutional neural networks for thermal infrared object tracking. *Knowledge-Based Systems*, 134(0950-7051), pp. 189-198.

Liu, W. a. A. D. a. E. D. a. S. C. a. R. S. a. F. C.-Y. a. B. A. C., 2016. Ssd: Single shot multibox detector. In: *European conference on computer vision*. s.l.:Springer, pp. 21-37.

Paszke, A. a. G. S. a. M. F. a. L. A. a. B. J. a. C. G. a. K. T. a. L. Z. a. G. N. a. A. L. a. o., 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, Volume 32, pp. 8026-8037.

Redmon, J., Divvala, S., Girshick, R. & Farhadi, A., 2016. *You Only Look Once*., s.l.: Proceedings of the IEEE conference on computer vision and pattern recognition..

Ren, S., He, K., Girshick, R. & Sun, J., 2015. Faster R-CNN: Towards Real-Time Object Detection. *Advances in neural information processing systems*, Volume 28, pp. 91-99.

Rezatofighi, H. a. T. N. a. G. J. a. S. A. a. R. I. a. S. S., 2019. Generalized intersection over union: A metric and a loss for bounding box regression. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. s.l.:s.n., pp. 658-666.

Sanders, J. a. K. E., 2010. *CUDA by example: an introduction to general-purpose GPU programming*. s.l.:Addison-Wesley Professional.

Shaoqing, R., Kaiming, H., Ross, G. & Jian, S., 2015. Faster R-CNN: Towards Real-Time Object Detection. *Advances in neural information processing systems*, Volume 28, pp. 91-99.

Sun, K. a. Z. Y. a. J. B. a. C. T. a. X. B. a. L. D. a. M. Y. a. W. X. a. L. W. a. W. J., 2019. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514*.

## Bibliography

Tsung-Yi, L. et al., 2014. Microsoft coco: Common objects in context.. In: *ECCV 2014. Lecture Notes in Computer Science*. Springer, Cham: s.n., pp. 740-755.

Wang, J. et al., 2020. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1 - 1.

Woeber, W., Szuegyi, D., Kubinger, W. & Mehnen, L., 2013. A principal component analysis based object detection for thermal infra-red images. In: *Proceedings ELMAR-2013*. Zadar: IEEE, pp. 357-360.

Xiao, J. W. a. K. S. a. T. C. a. B. J. a. C. D. a. Y. Z. a. D. L. a. Y. M. a. M. T. a. X. W. a. W. L. a. B., 2019. Deep High-Resolution Representation Learning for Visual Recognition. *TPAMI*.

Zachary, S., 2020. *Tesla Autopilot Accidents: 1 Out Of 4,530,000 Miles; US Average: 1 Out Of 479,000 Miles*. [Online] Available at: <https://cleantechnica.com/2020/08/01/tesla-autopilot-accidents-1-out-of-4530000-miles-us-average-1-out-of-479000-miles/> [Accessed 6 May 2020].

Zhengxia, Z., Zhenwei, S., Yuhong, G. & Jieping, Y., 2019. *Object Detection in 20 Years: A Survey*, s.l.: arxiv.