OBJECT DETECTION AND SYNTHETIC INFRARED IMAGE GENERATION FOR UAV-BASED AERIAL IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF ENGINEERING AND SCIENCE
OF BILKENT UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR
THE DEGREE OF
MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

By
Mehmet Akif Özkanoğlu
September 2023

OBJECT DETECTION AND SYNTHETIC INFRARED IMAGE GENERATION FOR UAV-BASED AERIAL IMAGES

By Mehmet Akif Özkanoğlu

September 2023

We certify that we have read this thesis and that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

İbrahim Körpeoğlu(Advisor)

Sedat Özer(Co-Advisor)

Özgür Ulusoy

Cihan Topal

Approved for the Graduate School of Engineering and Science:

Orhan Arıkan
Director of the Graduate School

ABSTRACT

OBJECT DETECTION AND SYNTHETIC INFRARED IMAGE GENERATION FOR UAV-BASED AERIAL IMAGES

Mehmet Akif Özkanoğlu
M.S. in Computer Engineering
Advisor: İbrahim Körpeoğlu
Co-Advisor: Sedat Özer
September 2023

This thesis contains two main works related to aerial image processing. In the first work (in the first main part of this thesis), we present novel approaches to detect objects in aerial images. We introduce a novel object detection algorithm based on CenterNet which yields the state-of-the-art results in many metrics on many aerial benchmark datasets, when this thesis was written. In this part, we study the effect of different loss functions, and architectures for improving the detection performance of objects in aerial images taken by UAVs. We show that our proposed approaches help improving certain aspects of the learning process for detecting objects in aerial images.

To train recent deep learning-based supervised object detection algorithms, the availability of annotations is essential. Many algorithms, today, use both infrared (IR) and visible (RGB) image pairs as input. However, large datasets (such as VisDrone [1] or ImageNet [2]) typically are captured in the visible spectrum. Therefore, a domain transfer-based approach to artificially generate infrared equivalents of the visible images for existing datasets is presented in the second part of this thesis. Such image pairs, then, can be used to train object detection algorithms for either mode in future work.

Keywords: Image Generation, Object Detection, Deep Learning, GANs.

ÖZET

TÜRKÇE BAŞLIK

Mehmet Akif Özkanoğlu
Bilgisayar Mühendisliği, Yüksek Lisans
Tez Danışmanı: İbrahim Körpeoğlu
İkinci Tez Danışmanı: Sedat Özer
Eylül 2023

Bu tez havadan görüntü işleme ile ilgili iki ana çalışmayı içermektedir. İlk çalışmada (bu tezin ilk ana bölümünde), hava görüntülerindeki nesnelerin tespitine yönelik yeni yaklaşımlar sunuyoruz. Bu tez yazıldığında, birçok hava kıyaslama veri kümesinde birçok metrikte en son teknolojiye sahip sonuçları veren CenterNet'i temel alan yeni bir nesne algılama algoritması tanıttık. Bu bölümde İHA'lar tarafından çekilen hava görüntülerinde farklı kayıp fonksiyonlarının ve mimarilerin nesnelerin tespit performansının arttırılmasına etkisi incelenmektedir. Önerilen yaklaşımlarımızın, havadan görüntülerdeki nesnelerin tespitine yönelik öğrenme sürecinin belirli yönlerinin iyileştirilmesine yardımcı olduğunu gösterdik.

Güncel derin öğrenme tabanlı denetlenen nesne algılama algoritmalarını eğitmek için ek açıklamaların kullanılabilirliği önemlidir. Günümüzde birçok algoritma girdi olarak hem kızılötesi (IR) hem de görünür (RGB) görüntü çiftlerini kullanıyor. Ancak büyük veri kümeleri (VisDrone [1] veya ImageNet [2] gibi) genellikle görünür spektrumda yakalanır. Bu nedenle, bu tezin ikinci bölümünde, mevcut veri kümeleri için görünür görüntülerin kızılötesi eşdeğerlerinin yapay olarak üretilmesine yönelik alan aktarımı tabanlı bir yaklaşım sunulmaktadır. Bu tür görüntü çiftleri, gelecekteki çalışmalarda her iki mod için de nesne algılama algoritmalarını eğitmek amacıyla kullanılabilir.

Anahtar sözcükler: Yapay Resim Üretimi, Nesne Tanıma, Derin Öğrenme, Çekişmeli Üretici Ağlar.

Acknowledgement

First and foremost, I would like to convey my deepest appreciation to my Co-Advisor Assist. Prof. Dr. Sedat Özer for his support and understandings. Without his guidance and dedicated involvement in every stage, this thesis would have never been achieved. Secondly, my sincere gratitude goes to Prof. Dr. İbrahim Körpeoğlu, whose assistance have been invaluable over the past three years. I would also like to thank Prof. Dr. Özgür Ulusoy and Assoc. Prof. Dr. Cihan Topal for joining my master thesis committee and for their invaluable contributions to my thesis.

This thesis benefits from the 2232 International Fellowship for Outstanding Researchers Program of TÜBİTAK (Project No:118C356, PI: Dr. Sedat Özer) as the results and the presented approaches are produced for the TÜBİTAK project under Sedat Özer's supervision. Note that, the financial support received from TÜBİTAK does not mean that the content of the publication is approved in a scientific sense by TÜBİTAK.

I would like to dedicate this work to my ever-supporting family and close friends.

Thank You

Contents

1 Introduction **1**

2 Related Work **5**

3 SkyDataNet: An Object Detection Algorithm for UAV-based Aerial Images **11**

3.1 Sequential Residual Block 14

3.2 The Hourglass Architecture 17

3.3 Unifying Self-Attention Modules (USAM) 17

3.4 Prediction Headers 18

3.5 An Analysis Considering Object Size 19

3.6 Thermal Analysis 20

3.7 Loss 23

3.8 Metrics for Object Detection 25

3.8.1 Intersection over Union (IoU) 25

3.8.2 Precision 25

3.8.3 Recall 26

3.8.4 Precision-Recall Curve 26

3.8.5 Recall-IoU Curve 30

3.9 Datasets for Object Detection 31

3.9.1 SkyData 31

3.9.2 VisDrone 32

3.9.3 NWPU VHR-10 32

3.9.4 DOTA 33

3.10 Experimental Results 33

4 InfraGAN: A GAN-based Domain Transfer Approach for Generating Synthetic Infrared Images 46

4.1 Generator Architecture 50

4.2 Discriminator Architecture 50

4.3 Residual Blocks 52

4.4 Discriminator Loss: 54

4.5 Generator Loss: 55

4.6 Experiments 57

4.6.1 Datasets for Infrared Image Generation 57

4.6.2 Metrics for Image Generation 59

4.7 Results 61

5 Conclusions **79**

A Additional Tasks: Image Registration **91**

A.1 Method I 91

A.2 Method II 93

A.3 Method III 93

List of Figures

3.1 Overview of our architecture. It is the two stacked version of Hourglass network as seen in Figure 3.5b and also includes two additional preprocessing convolutional blocks. The first one is a standard convolutional layer and the other one is sequential residual block. (a) is the original CenterNet exploits two stacked HourGlass without merging two distinct feature maps. For inference, the the second feature map is given into the output headers. Besides, first preprocessing layers are named as STEM used in Figure 3.5b. In contrast to (a), our proposed architecture SkyDataNet (b) coherently merges two feature maps for scale awareness by utilizing self attention mechanism as detailed in Figure 3.4. 12

3.2 These are headers for three different loss function. The final prediction of bounding boxes are calculated by decoding output of these headers. 13

3.3 Difference between 1D Gaussian and 2D Gaussian kernel. 14

3.4 The architecture of Unifying Self-Attention Modules as utilized between the output headers and the feature map obtained from each stack. S1, S2 refers to feature map coming from first and second HourGlass stack respectively. 15

3.5 The details of sequential residual blocks denoted as L1, L2 and L3 blocks in Figure 3.5b.The main difference is the order of down-sampling layers in the block. STEM (See Figure 3.1 for further details) is the initial block of HourGlass architecture consisting of 1 convoluional layer and single residual layer. The parameters for convolutional layer of STEM is 7x7 sized kernel, 3x3 padding and 2 stride while the parameters of single residual block is 1x1 kernel and 2 stride. Besides, out_c refers to output channel size of STEM layer. In this figure, f is used for filter size, p is for padding, b is number of blocks used in consecutive manner. BN means Batch Normalization. UP1 is actually different version of L2. UP1 has 2 consecutive blocks while the oher has 4 blocks. UP2 denotes bilinear interpolation with 2 scale factor 16

3.6 The analysis of original two stacked network. The result demonstrates that any of these stacks does not focus on the objects with specified sizes. 21

3.7 Another proposed approach to assign first stack for small objects while forwarding only large objects to second stack of HourGlass network. The first plot shows combination of other three plots. The second one is the ground truths detected by the first stack. The second one is detected by the second stack, and, the last one is the non-detected objects. The x axis is for the width of the object while the y axis is for the object's height. Small objects were mainly detected by the first stack. The second stack could not detect small object mostly. However, for the large objects, both of the stacks can detect them. The rows in Table 3.5 - named as "CenterNet (2 HG) - Seperated Cluster"- indicates quantitative results of the proposed approach in Figure 3.7. As seen Table 3.5, this approach did not outperform other approaches mentioned in this thesis. 22

3.8 Sample frames from SkyData. As seen in the frames, The videos are captured from varying altitudes. 27

3.9 The segmentation masks on each class instance on sample frames. These are colored by utilizing the annotated polygons. 28

3.10 The ground truth of an example frame from SkyData 39

3.11 The detection results obtained from SkyDataNet 40

3.12 The detection samples from CenterNet (2HG) 41

3.13 The detection results obtained from SkyDataNet + 2D-Gaussian 42

3.14 The detection results obtained from SkyDataNet (1HG) + 2D-Gaussian + L_s 43

3.15 The detection results obtained from SkyDataNet (1HG) with 2D Gaussian heat map. 44

4.1 Overview of our proposed InfraGAN architecture for training stage. The details of our generator are given in Figure 4.4 and the details of our discriminator are given in Figure 4.5. 48

4.2 Inference stage of InfraGAN architecture. For this stage, the images from the visible domain are fed into the generator detailed in Figure 4.4. 49

4.3 Comparison of two discriminator types: one-level (on the left) and two-level discriminators (on the right). One-level discriminator only classifies the entire image while the two-level one classifies the image and its pixels as being fake or real at the same time. . 50

4.4 Our U-Net based generator architecture. In the figure, Xf is used to represent the total number of filters in each block where X is the total number of filters. 51

4.5 Our U-Net based discriminator architecture. It shows the discriminator architecture where the symbol $\oplus$ means element-wise summation and the symbol $\bigcirc$ means concatenation. See Figure 4.6 for the details of the attention block. 53

4.6 The attention block of the architecture as used in the discriminator. For further details, see Figure 4.5. $\varovee$ refers matrix multiplication. 54

4.7 Comparative results of multiple algorithms on sample image pairs from FLIR dataset. The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively. 67

4.8 Comparative results of multiple algorithms on sample image pairs from KAIST dataet. The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively. 68

4.9 Comparative results of multiple algorithms on sample image pairs from VEDAI dataset. The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively. 69

4.10 Comparative results between simple U-Net generator and other GANs to show the difference of using GAN loss. These qualitative results show that using GAN architecture can help improving the generated image quality. In the figure, the first two columns show the original images, from third column to sixth column shows GAN-based networks' results and the last column shows U-Net results on three samples. 70

4.11 Comparative results between InfraGAN and U-GAT-IT[3]. As seen in the lower right corner, U-GAT-IT[3] does not produce the objects and edges in the image in high resolution and qualitiy. 71

A.1 The chessboard in RGB domain captured by Zenmuse H20-T. The colored points are matching points captured by conventional edge detection algorithm. 92

A.2 The chessboard in infrared domain captured by Zenmuse H20-T. The colored points are matching points captured by conventional edge detection algorithm. 93

A.3 The registered version of both images. The left RGB image is mapped into to original Infrared image. 94

List of Tables

3.1 The detection results over SkyDataV1. This table includes different versions of our proposed architecture 'SkydataNet' and the SOTA works. The parenthesis refers the name of backbone utilized by the method. 2HG means 2 stacked HourGlass while 1HG is single HourGlass network. 36

3.2 The detection results over SkyDataV1. The results compares performances of the methods in different domains. 'RGB+Thermal' denotes that the input of architecure is concatenation of RGB and Thermal domain as six channel whereas '+Thermal' refers to three channeled input in thermal spectrum. 36

3.3 The detection results over DOTA to compare SkyDataNet and CenterNet 37

3.4 The detection results over DOTA to analyze performances of the proposed loss terms on CenterNet 37

3.5 The detection results on DOTA to compare CenterNet and proposed approaches detailed in Section 3.5 37

3.6 The detection results over VHR. This table includes different versions of our proposed architecture 'SkydataNet' and the other works. 38

3.7 The detection results over VHR to analyze performances of the proposed loss terms on CenterNet 38

3.8 Additional Evaluated Results on VHR-10 to compare CenterNet with two stacked HourGlass and CenterNet using other SOTA backbones in different parameters.AdamW means the usage of AdamW[4] as an optimizer for the training. LinearLR indicates the usage of linear learning rate scheduler for each iteration. . . . 45

4.1 Network definitions based on their configurations, where $\mathcal{L}$ refers to the used loss function and A.F. means activation function. Encoder and Decoder is listed for the generator. This table also compares run times of different networks on KAIST dataset. 62

4.2 Comparison of different algorithms on VEDAI dataset [5]. Best result is shown in bold for each metric over all classes for each dataset. 66

4.3 Comparison of different algorithms on KAIST dataset [6]. Best result is shown in bold for each metric over all classes for each dataset. 72

4.4 Comparison of different algorithms on FLIR dataset. Best result is shown in bold for each metric over all classes for each dataset. . 72

4.5 Comparison of four different algorithms on the KAIST dataset for Cyclist class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 73

4.6 Comparison of four different algorithms on the KAIST dataset for People class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 73

4.7 Comparison of four different algorithms on the KAIST dataset for Person class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 74

4.8 Comparison of four different algorithms on the VEDAI dataset for Car class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 74

4.9 Comparison of four different algorithms on the VEDAI dataset for Truck class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 75

4.10 Comparison of four different algorithms on the VEDAI dataset for Pickup class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 75

4.11 Comparison of four different algorithms on the VEDAI dataset for Tractor class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 76

4.12 Comparison of four different algorithms on the VEDAI dataset for Camping Car class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 76

4.13 Comparison of four different algorithms on the VEDAI dataset for Boat class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 77

4.14 Comparison of four different algorithms on the VEDAI dataset for Van class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 77

4.15 Comparison of four different algorithms on the VEDAI dataset for 'Other' class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric. 78

4.16 Comparison of using L1 (InfraGANv2) and using Mean Squared Error, MSE, (InfraGANv1) for pixel by pixel loss on VEDAI and FLIR. 78

Chapter 1

Introduction

This thesis contains two main works including an introduction of an object detection algorithm and generating synthetic infrared equivalent of given visible (RGB) images. Therefore, below, we provide the motivation for each of those works separately:

(A) Object detection: After the sudden rise of deep learning in the 2010s, deep neural networks started to be used in many problems such as object recognition, autonomous systems, and image generation in the field of computer vision. After this remarkable success, the deep networks were quickly adapted to many real applications. More specifically, when you look at the earlier studies in this field, it is obviously seen that most of the publicly published open-source datasets such as COCO[7] and ImageNet[2], which contain multiple classes, are mainly utilized to verify the success of many proposed object detection algorithms. However, such mainstream datasets are not helpful in all possible applications. There are many reasons for that.

The first is that although the most known datasets involve large objects in their images, many applications may require detecting small-scale objects in different aspect ratios, as they are captured from any flying platform such as UAVs. Secondly, these applications may need to support a large variety of camera angles

in the images. For instance, while the images in the datasets such as ImageNet and COCO, include a frame that shows the items on the table from a direct camera angle instead of a high angle such as bird view, the applications such as city surveillance and security-based AI products need to process UAV frames.

The object detection task is examined in the context of aerial images in this thesis. Consequently, the use of two-stage costly algorithms based on Region Proposal Networks (RPN) such as [8, 9] were avoided because of wasting inference time with heavy operations such as filtering out proposals by investigating twice for a single frame. Furthermore, the proposed object detection model follows anchor-free approaches in order to get rid of encoding and decoding operations for the training and inference stage. Encoding operations for anchor-based refer to several stages. The first is to find overlapped anchors with ground truth boxes. Besides, the second is to calculate the values of offsets between the box and overlapped anchors. On the other hand, decoding operations for anchor-based approaches denote several stages like encoding. The first stage is to find the anchor for the output box by using index of predictions. Then, the second is to compute the final prediction bounding box by utilizing prediction values and properties of the corresponding anchor. In the end, both of them mean a computational cost for inference and training.

To summarize, in this thesis, related to object detection, our contributions involve:

1. 2D Gaussian loss function to represent center point more sensitive to object's size and aspect ratio;
2. the introduction and use of Unifying Self-Attention Module to learn the relationship between output stacks;
3. experiments on four different UAV-based benchmark data sets including SkyDataV1.

(B) Synthetic Infrared Image Generation:

Recent developments in sensor types and the increasing demand in autonomous

systems, (as in [10, 11, 12]), have moved many researchers' attention to the data fusion algorithms where using more than one modality is desired. Sample areas and works can be found in [13, 14, 15, 16]. Many autonomous systems use images coming from both visible and infrared (IR) spectrum. Since both IR image (typically containing information from the thermal spectrum) and visible image (containing information from the visible spectrum) summarize different properties of a scene, acquired IR and visible images of the same scene would look similar but not the same. A typical example is text images. For example, reading the text information on a posted sign might be an easy task in a visible image, however, that can easily become a troublesome task if the reading is done in the IR spectrum since the text might not appear on the IR image.

Infrared images have primary importance for various vision tasks such as object detection and object tracking especially in the night mode. Infrared cameras have the ability to distinguish an object in difficult lighting conditions such as in night or in cloudy weather. On the other hand, because of perceiving the thermal temperature level, IR images can also help to differentiate objects based on their body temperature, i.e., they provide extra information in addition to the visible spectrum. As a result, IR images can help to improve the understanding of the surroundings and making miscellaneous vision tasks such as detection, tracking, and self-driving more feasible.

Making a deep *fusion* algorithm to work better than the existing single-modality based algorithms, usually a large data set which is obtained by multiple sensors is needed. However, the typical trend is using single-modality based large public datasets in many vision and pattern recognition applications since major large public datasets (such as ImageNet [2], PASCAL VOC [17] or MS COCO [7]) contain images from only the visible spectrum. That creates a problem to use such datasets in multi-sensor applications since they provide only visible images. Alternative solutions to still using such existing public datasets in fusion algorithms include collecting new datasets with multiple sensors or using domain transfer-based techniques to obtain (generate) the missing modalities. In this thesis, we look for an efficient transfer technique to specifically obtain the IR equivalent of a given visible image.

In this thesis, we propose a generative adversarial network (GAN) based deep architecture to generate the IR equivalent image of a given (input) visible image to help facilitate large and single modality-based datasets in multi-sensor-based applications. We call our solution *InfraGAN*. InfraGAN uses Structural Similarity Index Measure (SSIM) in its architecture to focus on learning certain structural similarities between the IR and visible domains while the pixel-based L1 norm enforces the architecture to look like an IR image.

The overview of our architecture is given in Figure 4.1 which shows the training stage of overall architecture. Furthermore, Figure 4.2 shows the inference stage of the overall architecture. When comparing Figure 4.1 and Figure 4.2, it is evident that the discriminator is utilized solely during the training stage, whereas the generator network plays a role in both the training and inference stages. Naturally, the generator strives to generate more realistic images during the inference stage by trying to beat the discriminator in the training stage.

We compare and evaluate our proposed architecture's performance on three different datasets: on one aerial dataset and on two ground-taken datasets (having both visible and IR pairs). We use five metrics to compare our algorithm's performance to two other baseline networks. Our experimental and comparative results show that our proposed architecture generates better IR equivalents when compared to both recently proposed ThermalGAN [18] and Pix2Pix [19] architectures in all of those five metrics.

In the section on synthetic image generation, our contributions include:

1. introduction of a GAN-based domain-transfer technique to obtain the IR equivalent of a given visible image where *both* generator and discriminator parts use an encoder-decoder architecture;
2. use of an additional loss term based on SSIM, (see Eq. 4.8) for improved results;
3. experiments on three different large benchmark data sets.

Chapter 2

Related Work

After the rise of deep learning in the 2010s, Neural networks achieved a significant impact on object detection and segmentation. Then, visual recognition literature has focused on proposing new architectures instead of conventional computer vision methods by extracting hand-crafted features. In this respect, convolutional neural networks has a leading effect on several image recognition challenges after Alex-Net [20] emerged as a top algorithm with a sensationally large margin. [21] proposes more complex and deeper network compared to [20]. In addition to its complexity, [21] employed uniform convolutional layers throughout its architecture instead of using a combination of 11x11, 5x5, and 3x3 convolutional layers as in [20]. Therefore, [21] reduces trainable parameters so that it provides faster learning. As a result, whereas [21] contributed to improving the accuracy of deep models, it also inspired subsequent convolutional architecture design.

Subsequently, [22] evolved residual blocks, allowing the founding of extremely deep networks. The use of skip connections in residual blocks eliminates the problem of vanishing gradients, which leads to the ability to train deeper networks effectively. [23] and [24] introduced a novel approach to attenuate vanishing gradients by reusing features throughout consecutive layers. In pursuit of this goal, they promoted dense blocks where each layer processes the feature maps from all preceding layers within the same blocks. Thus, they increase the model

efficiency in various computer vision tasks, including image classification, object detection, and semantic segmentation.

[25] is another convolutional neural network architecture which is designed for mobile devices and optimized for limited resources. Moreover, it provides efficient and lightweight solutions for deep learning tasks while achieving high performance in terms of evaluation metrics in image classification and object detection. [26] is another resource-aware convolutional neural network architecture. It developed a compound scaling method which uniformly scales the depth, width, and resolution of the network by applying depth-wise convolutions, expansion layers, and point-wise convolutions. Ultimately, this method is derived from the idea that leveraging any of these dimensions can enhance the model's performance.

[27] and [28] proposed deformable convolutions to capture spatial transformations and object deformations within images. Therefore, it allows the deep layers to be more adaptive to complex and irregular objects. They mitigated the issue of fixed-size kernels in conventional convolutions by adjusting receptive fields with the help of learnable offsets.

[29] single-stage object detection framework that produces a collection of default bounding boxes for each spatial point with various aspect ratios and sizes. Next, for these default boxes, the network predicts class scores and bounding box offsets. [30, 31, 32] are the real-time single-stage detectors that directly output bounding boxes and class confidences from a single pass through DarkNet-based convolutional neural network. It creates a grid from the input image and predicts objects of different sizes inside each grid cell. By using predefined anchors, It aims to generate actual predictions by computing offsets between predefined anchors and object sizes.

[33, 34] present novel architectural improvements building upon [31]. Its improvements involve adapting transformers layers to the overall architecture and escalating the number of detection layers in order to enhance detection performance for UAV images. [35] is an improved version of the [31]. It promoted key features to enhance object detection capabilities in addition to maintaining

real-time inference requirements. [35] employs a number of data augmentation methods such as mosaic data augmentation to generalize the model for various scenes. Moreover, the most significant other improvements that [35] utilized are decoupled head and anchor-free prediction headers. The first one is to separate object locations and class confidences into two separate subnetworks. By separating into the subnetworks, the authors aim to obtain better localization and classification performance. The second approach is to remove predefined anchors from the prediction headers, simplifying prediction steps. Also, the model is more capable of detecting objects of varying sizes and aspect ratios. [36] developed the focal loss to eliminate class imbalance problems in main-stream object detection datasets such as [7]. As an architectural innovation, [36] also applies a Feature Pyramid Network (FPN) at multiple scaled objects.

[37] introduces a heatmap for key points to detect the objects. As its name suggests, it detects using corner points such as the top-left and bottom-right corners of bounding boxes. In this way, it removes the need for predefined anchors simplifying the detection step, because it does not need to encode or decode bounding boxes. Similar to [37], [38] detects the center points of the objects along with their width and height. As well as [37], [38] also utilizes a heatmap to represent the center points of the objects and predicts their offsets for precise bounding box prediction with eligible IoU. [39] propose another approach by unifying a multi-stage detector with a single-stage detector. Consequently, it claimed that this approach decreased the high false negative rate of multi-stage detectors while increasing the precision of object proposals obtained by the single-stage detector.

[8] is a widely used two-stage object detection method. It proposed the concept of RoI (Region of Interest) pooling, leading to efficiently detect the objects by only processing on their regions. In [8], first, it extracts a feature map by passing the entire image through the convolutional backbone. These feature maps are then applied for RoI pooling in order to obtain fixed-size feature vectors for each region proposal. [9] is an improvement over the [8]. It involves two main stages: a Region Proposal Network (RPN) that produces candidate object bounding boxes indicating the region includes an object instead of being background. Then,

the subsequent network refines these proposals to generate class labels and final bounding boxes by filtering out false positives. [40] is an extended version of [9] that includes an extra branch to generate object masks. With the help of this mask generation branch, Mask R-CNN also manages to segment the object masks within the predicted bounding boxes.

After transformers' spectacular success in Natural Language Processing (NLP) as in [41], [42] is one of the leading deep learning architectures that applies the transformer model for computer vision tasks. It promoted an innovative architecture to image classification and similar vision tasks by dividing entire image into fixed-size patches, and then processing them as if these patches were sequences of data by using attention and transformers modules. After that, [43] directly applies transformers to the object detection problem. Besides, it also introduces a novel loss function for more accurate localization and confidence scores for the class. It can be classified as an anchor-free approach because the bipartite matching loss mechanism associates ground truth objects with predictions by ensuring that each of the predictions is matched with the most aligned ground truth box. This is the reason why it does not need to use predefined anchors.

[44] improved the deformable version of [43] to eliminate the gradual convergence and excessive complexity issues. It unifies relation modeling capacity of transformers and the skill of deformable convolution in the sparse spatial sampling. To do this, they developed deformable attention module which utilizes sampling locations of deformed objects in order to extract the notable key elements of feature maps in the image.

[45] presented a hierarchical vision transformer splitting an entire image into non-overlapping windows; therefore, the model does suffer from computational complexity due to high*resolution images. Since, the architecture employs these windows in a hierarchical manner. In addition to this, [45] applies the shifting operation to these windows, allowing to capture dependencies and relations between non-adjacent windows. Consequently, by aggregating the features across consecutive layers, [45] is able to interpret the features within both the local windows and the entire image. [46] enhances the training convergence of the models

inspired from [43] by proposing a denoising (DN) training method. It refers to a slow convergence issue in [43] resulting from unstable bipartite matching loss. To tackle this issue, noisy labels and boxes are utilized for decoders in transformers. Then, [47] introduces a novel contrastive denoising approach to improve detection performance over [46]. They feed negative and positive samples of the same bounding box by classifying the smaller noise as positive and the larger noise as negative. At the end this approach allows the model to filter out duplicated bounding boxes for a single object.

Although a lot of transformers-based architectures are applied to computer vision tasks such as segmentation and detection, [48] proves that pure convolutional networks still are able to achieve better performance on other transformers-based networks such as [45]. In this respect, [48] justified its ability by modernizing standard residual blocks by inspiring from vision transformers and attention modules. In other words, there are also several semi-supervised approaches like [49, 50]. [49] introduces semi-supervised object detection by utilizing pseudo labels generated by teacher networks. During end-to-end semi-supervised training, it enhances the precision of pseudo labels, resulting in better convergence of the object detection model. [50] adapts a semi-supervised object detection training approach to other transformers-based detectors like [43]. Therefore, they developed a Stage-wise Hybrid Matching strategy as an improvement over the one-to-one assignment strategy in other transformers-based detection algorithms.

Generating infrared images from a given visible image has not been widely studied in the recent literature and there are only a few published earlier works available (as in [18, 51, 52, 53]) reporting statistical and comparative results. There are also a relevant class of works that focus on fusing both IR and visible images to obtain a single fused image as in [54, 55]. However, our line of work differs from such fusion-related literature as those works take both IR and visible image pairs as input, while we take only visible image as input and generate its IR equivalent as output. Most of those relevant literature to generate IR images introduces GAN-based architectures as in [18, 52, 53, 3]. Among those, ThermalGAN [18] uses a U-Net based architecture in its generator and uses a standard convolutional network (a basic discriminator) in its discriminator to

classify the entire image being fake or real. In that work, the authors used ThermalWorld dataset [18] which consists of 5098 of visible and thermal image pairs along with their segmentation annotations for ten classes: truck, car, van, person, boat, bus, cat, building, tram, dog.

In another work [52], the authors generated IR images to track objects. There, the authors studied the performance of Cycle-GAN [56] and Pix2Pix [19] based networks for generating IR images. They used both paired and unpaired datasets in their experiments. In [53], the authors introduced using multiple generators. Each generator would focus on learning the properties of a different scene. Another Res-Net [22] based classifier was used to choose which generators' output would be most suitable for the given input image. They used a dataset[1] containing 40479 image pairs. There are also recent GAN based SOTA works utilizing attention modules such as [3]. It presented a novel unsupervised image-to-image translation. Even though it is not developed for infrared image generation, we included its comparative results in this thesis in Table 4.2 and Figure 4.11.

Our work differs from the above-mentioned works by introducing a novel GAN based solution where our discriminator uses an encoder-decoder based architecture and our generator uses an additional loss term based on SSIM in that architecture. Furthermore, while there are not many works in the literature presenting statistical results on large datasets, we utilize three different datasets in our experiments to compare our proposed solution to two recently proposed GAN-based architectures from the literature (Pix2Pix and ThermalGAN), in addition to comparing our results to simple U-Net architecture. The main differences of our network from the most relevant literature include: (i) the use of pixel-based loss using SSIM in the loss function and (ii) the inclusion of a decoder part in the discriminator architecture using residual blocks for the generation of IR images.

[1] `https://www.kaggle.com/c/planet-understanding-the-amazon-from-space` (last accessed on December 2022)

Chapter 3

SkyDataNet: An Object Detection Algorithm for UAV-based Aerial Images

We introduce a new object detection algorithm in this section to detect tiny objects in SkyData. Our algorithm (called SkyDataNet) uses the center point idea (similar to Centernet [38]). The overview of SkyDataNet is given in Figure 3.1. In our architecture, we use two hour-glass architecture and the outputs of each of those is given to a prediction block (called as *Conv Headers* in the figure). The HourGlass architectures are used as the backbone in our proposed detection architecture. The first two blocks in the network (shown in blue and red colors) act as a down-scaling network to reduce the input dimensions. Due to the fact that SkyDataNet uses two stacked HourGlass as backbone, we classified our model as two staged object detector even though it does not involve Region Proposal Networks as in [9]. In the other two staged object detectors, the first region proposal networks predicts more bounding boxes so as not to miss any false negatives by empowering the model in terms of recall. Then, the second network refines the outputs of RPN to produce more precise boxes. However, in SkyDataNet, we followed two two-stacked approaches as a two-stage detector to increase improvements of the detector. By increasing the depth of networks, we

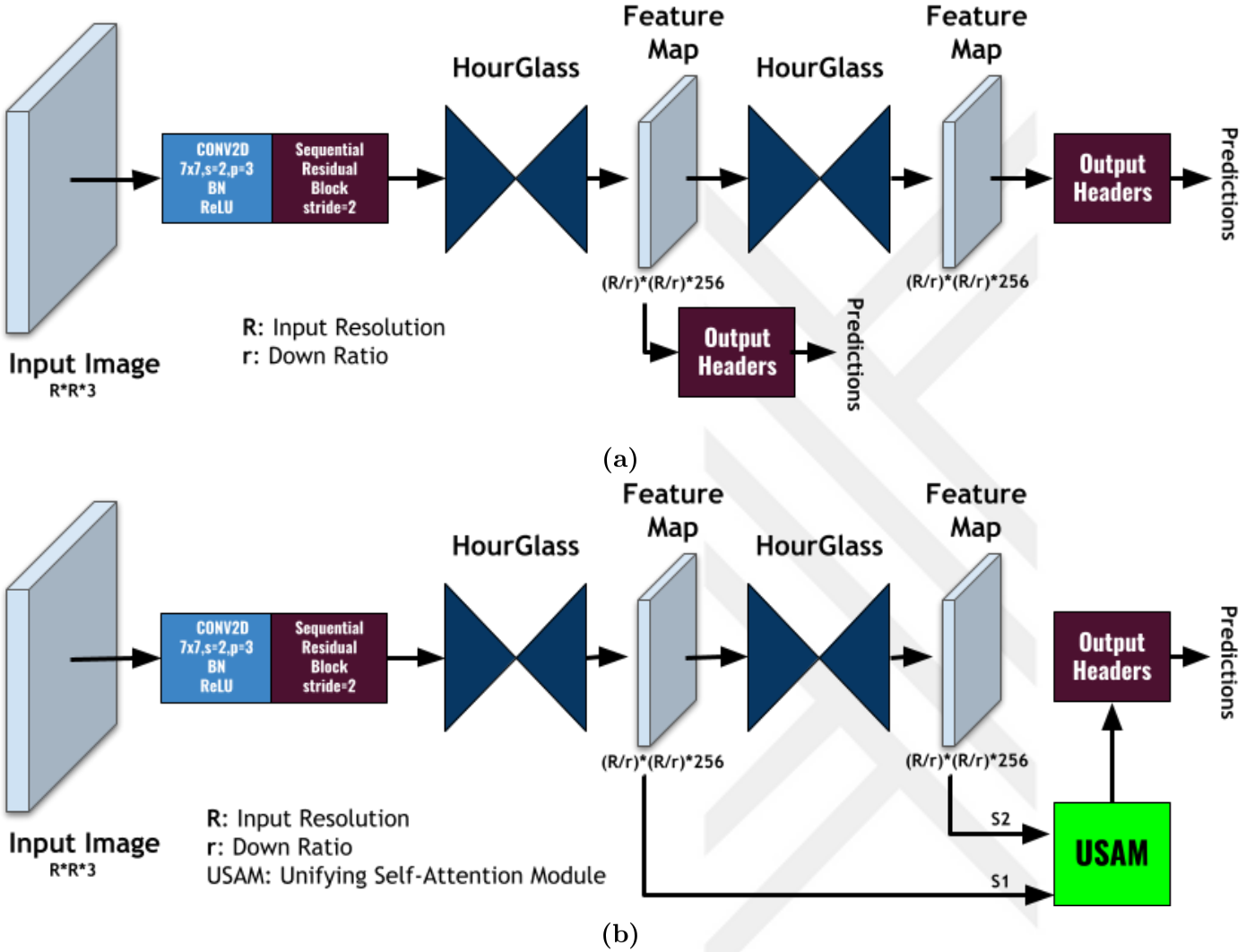


Figure 3.1: Overview of our architecture. It is the two stacked version of Hourglass network as seen in Figure 3.5b and also includes two additional preprocessing convolutional blocks. The first one is a standard convolutional layer and the other one is sequential residual block. (a) is the original CenterNet exploits two stacked HourGlass without merging two distinct feature maps. For inference, the the second feature map is given into the output headers. Besides, first preprocessing layers are named as STEM used in Figure 3.5b. In contrast to (a), our proposed architecture SkyDataNet (b) coherently merges two feature maps for scale awareness by utilizing self attention mechanism as detailed in Figure 3.4.

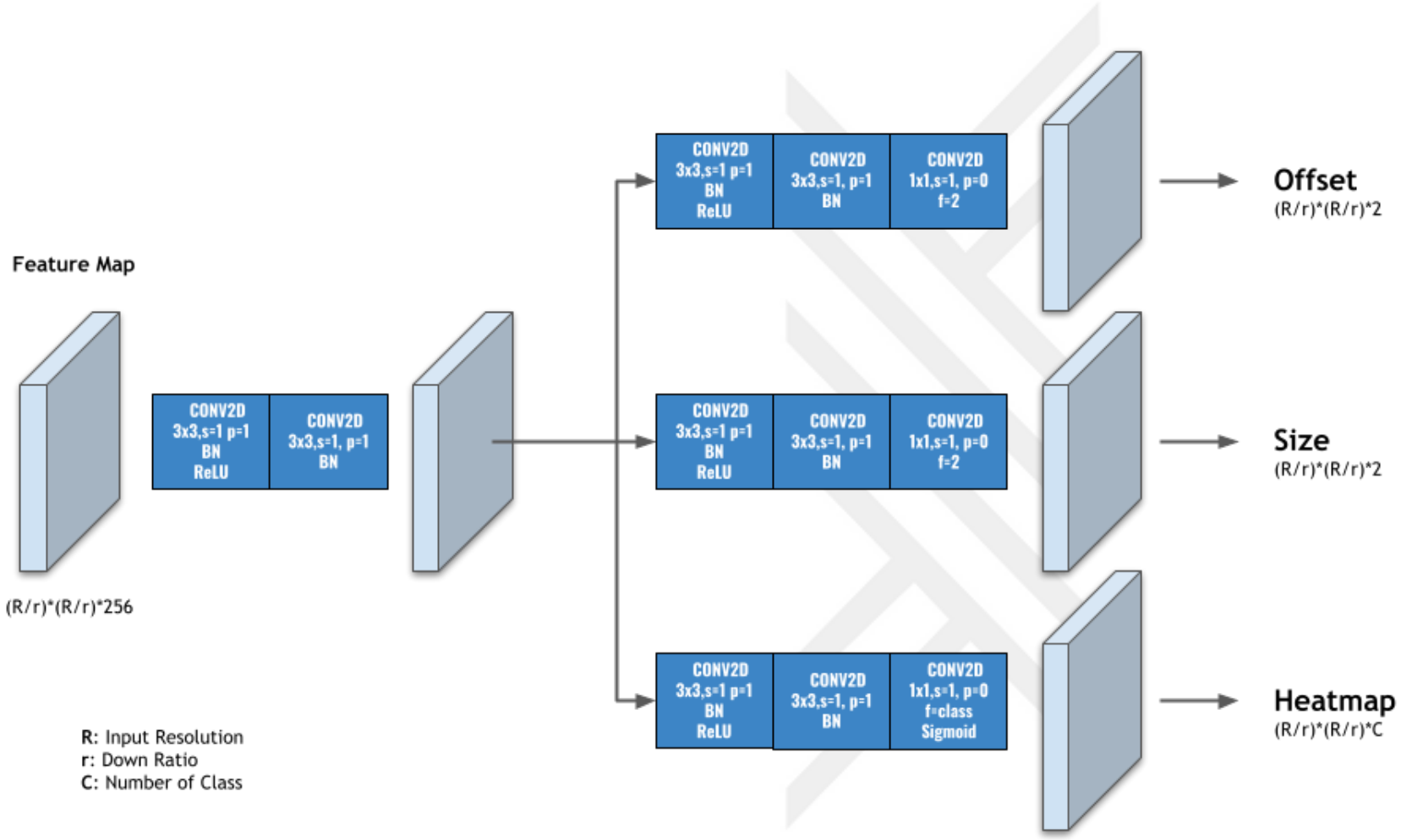


Figure 3.2: These are headers for three different loss function. The final prediction of bounding boxes are calculated by decoding output of these headers.

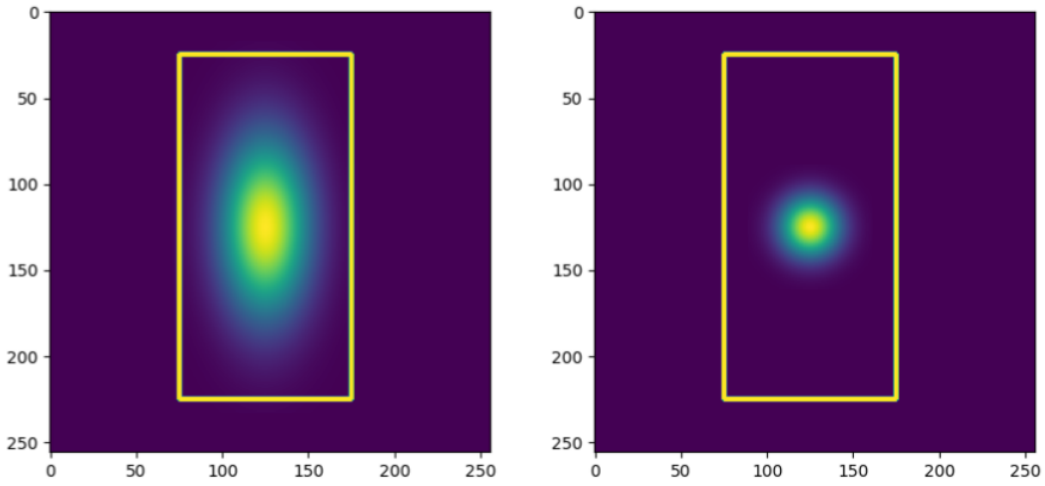


Figure 3.3: Difference between 1D Gaussian and 2D Gaussian kernel.

aimed at a more capable detector while avoiding two consecutive subdetectors. On the other hand, we tackled several experiments to force the first stack to focus on small objects while forwarding large objects into the second stack. Next, we explain the blocks in our network.

3.1 Sequential Residual Block

In the backbone of our network, namely HourGlass architectures, we use sequential residual networks [22] (See Figure 3.5), since the residual blocks help preserving the gradients along all layers (to reduce the effect of vanishing gradients). When examining the details of the residual blocks in Figure 3.5b, namely L1, L2 and L3, The first one, L1, is used to lower the resolution of the preceding feature map coming from previous layer by leveraging convolution layer with 2 stride kernel whereas the second (L2) block utilizes two consecutive residual layers with skip connections. At the end, the last one (L3) employs the residual block with skip connection as the initial layer and another residual layer with convolutional connection with stride 1. Since, it is used to equalize the filter size of the previous feature map with the next block.

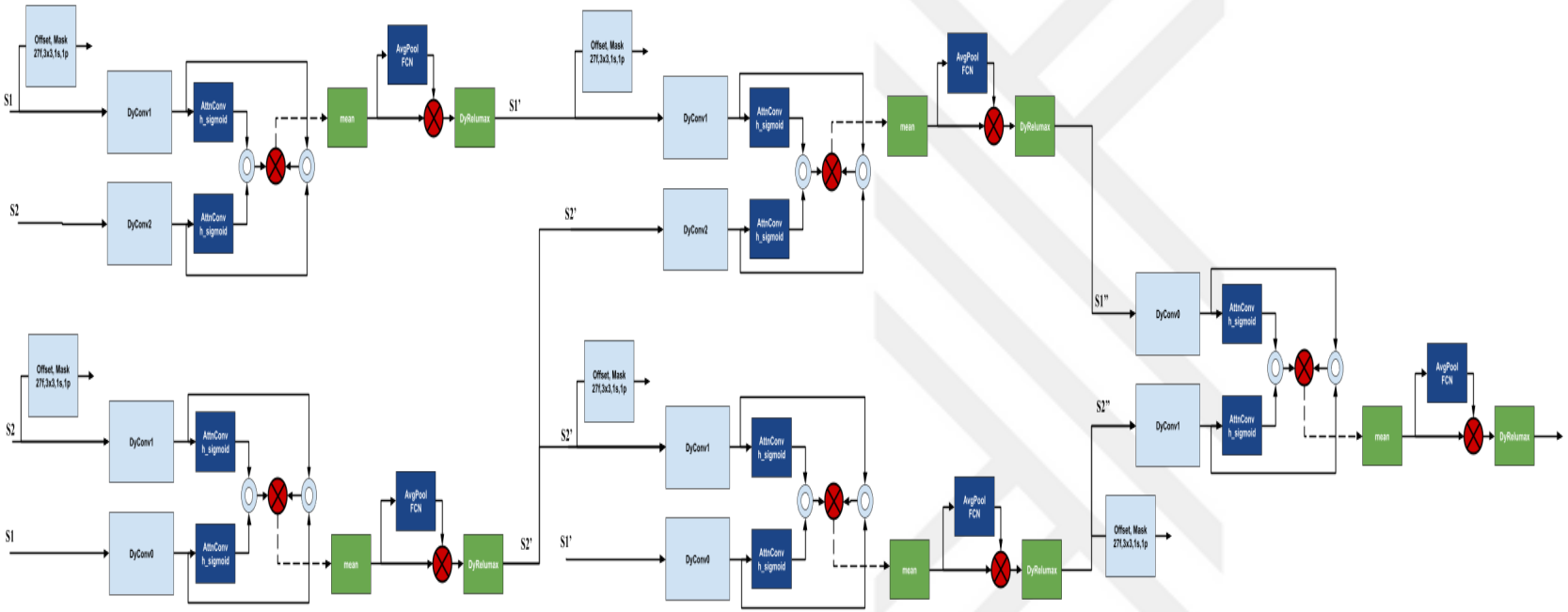
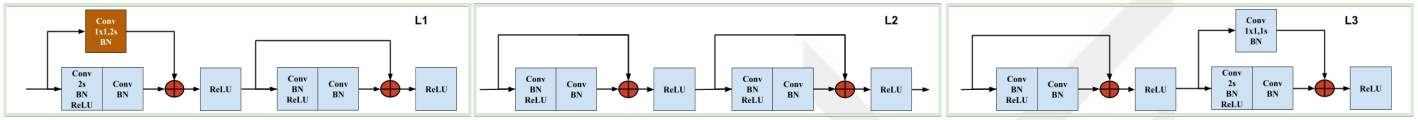
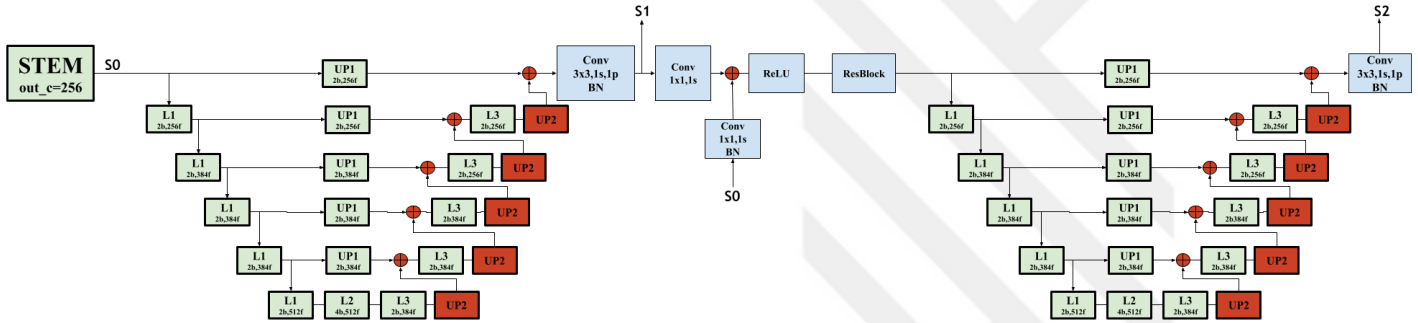


Figure 3.4: The architecture of Unifying Self-Attention Modules as utilized between the output headers and the feature map obtained from each stack. S1, S2 refers to feature map coming from first and second HourGlass stack respectively.



(a) The details of sequential residual blocks denoted as L1, L2 and L3 blocks in Figure 3.5b.The main difference is the order of downsampling layers in the block.



(b) The detailed blocks of HourGlass Network. L1 and L3 are composed of pre-downsampling and post-downsampling convolutional layers, respectively, whereas L2 consists of two sequential residual blocks with identical skip connection, as seen in Figure 3.5a.

Figure 3.5: The details of sequential residual blocks denoted as L1, L2 and L3 blocks in Figure 3.5b.The main difference is the order of downsampling layers in the block. STEM (See Figure 3.1 for further details) is the initial block of HourGlass architecture consisting of 1 convoluional layer and single residual layer. The parameters for convolutional layer of STEM is 7x7 sized kernel, 3x3 padding and 2 stride while the parameters of single residual block is 1x1 kernel and 2 stride. Besides, out_c refers to output channel size of STEM layer. In this figure, f is used for filter size, p is for padding, b is number of blocks used in consecutive manner. BN means Batch Normalization. UP1 is actually different version of L2. UP1 has 2 consecutive blocks while the oher has 4 blocks. UP2 denotes bilinear interpolation with 2 scale factor

3.2 The Hourglass Architecture

Hourglass architecture[57] is an encoder-decoder based network that connects low resolution features to high resolution features thorough skip connections in a similar way to the U-Net architecture [58]. The Hourglass module uses same padded fully convolutional layers along the skip connections (shown in Figure 3.5a), instead of using identical skip connections as in the U-Net architecture. Figure 3.5b shows the architecture of the Hourglass network which includes down-sampling and up-sampling layers. In the figure, the red blocks are up-sampling layers which increase the resolution with the rate of 1/2; and L1 blocks, sequential residual blocks, are the down-sampling layers (where stride is 2) that lower resolution with the rate of 1/2. For up-sampling, we used nearest-neighbor interpolation to reduce the number of trainable parameters (as opposed to using deconvolutional layers).

3.3 Unifying Self-Attention Modules (USAM)

Figure 3.4 shows our Unifying Self-Attention Module which combines two distinct feature maps separately observed from the output of each HourGlass stack similar to [59]. We combine the outputs (feature maps) of two distinct stacks at the same resolution, while the earlier work in [59] unifies output of consecutive layers at different resolutions. In our way, we aim to preserve features belonging to small-scale objects in output prediction headers. Besides that difference, our module also differs from [59] such that our module does not utilize upsampling between the feature maps. The 'AttnConv' module in Figure 3.4 is composed of 2D adaptive average pooling, one fully connected layer with one neuron and hard sigmoid function in order to observe stack-based attention weights. Hard sigmoid is used to compensate the computational cost of self-attention mechanism over HourGlass architecture, because, [60] reported that hard sigmoid reduces the computational time, even though original sigmoid function slightly outperforms in detection accuracy. Modulated deformable convolutional network [28] is

preferred to adaptively change receptive field to increase the spatial relationships among the pixels in DyConv blocks. [61] suggests that object detection bounding box regression subheader may have large margin depending on the object, because of that DyReLU 2[61] is used to dynamically select the most convenient features for the corresponding object. At the end, USAM block contains three consecutive stages consisting of dynamic and attention convolutional layers as detailed. These stages apply the attention mechanism along different aspects. Namely, the first attention focuses on the stack awareness between the outputs of two HourGlass networks, the second focuses on the spatial awareness among spatial pixels, whereas the last enhances the task awareness within the output channels.

3.4 Prediction Headers

Figure 3.1 demonstrates the overall architecture of SkyDataNet. The first two preprocessing blocks reduce the input resolution (where the down-ratio parameter r is set to 4 in the figure). Then, each HourGlass (backbone) network yields a feature map and that feature map is then fed into the prediction headers as shown in Figure 3.2. Consequently, we have the prediction layers that utilize combined features from each stack of HourGlass in our architecture as opposed to the prediction layer utilizing the last stack of HourGlass backbone in CenterNet [38]. Our objective is to process features at distinct levels, because, in classical and sequential networks, (such as VGG [21]) as we go deeper, the feature maps shrink and consequently the smaller-objects' information can get lost along the process.

Increasing the prediction layers in a network and using skip connections help learning the variance of scale in classes better as also our previous work reported in [33]. Similar to that idea, we also increased the number of prediction headers to two in SkyDataNet. Each prediction header contains three parallel branches. The first branch is used to identify the center point of bounding boxes. In other words, this branch is defined as a heatmap corresponding to the probability that

the relative point is the center of the object. Besides, the second branch is used to regress the width and height of the object and the last branch is used to predict offsets of center points to localize objects more precisely.

3.5 An Analysis Considering Object Size

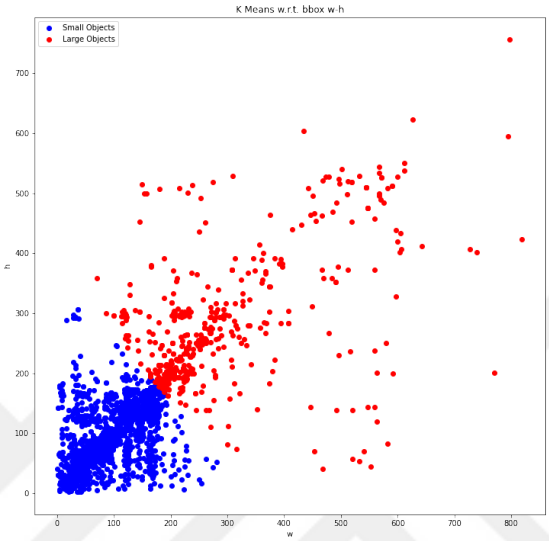
As previously indicated, our objective in object detection is to achieve enhanced precision and not to miss real small objects in the given frame as much as possible. For that purpose, we proposed a novel approach building upon the original CenterNet architecture. As delineated in the earlier sections, our model predominantly comprises of 2-HourGlass networks. By using prediction headers, both outputs of these HourGlass stacks can generate predictions for objects of varying sizes as obviously demonstrated in Figure 3.6b.

In contrast to this, we have introduced a distinction: the first stack should exclusively focus on small objects by filtering out large ones. The approach is rooted in the understanding that the later layers in the second stack may not be able to capture the features of small objects which are included in a few of pixels. Consequently, the second stack should focus on only large objects. Because the deeper layers can inspect large objects better. For this reason, we clustered bounding boxes with respect to their width and height by using K-Means [62] algorithm as small and large objects as in Figure 3.6a. As described, only the small objects are fed into the first stack (HourGlass) of the backbone to compute the loss function, whereas only the larger ones are utilized for the second stack's loss function. In other words, the output of the first stack is matched with small objects, and the output of the second stack is matched with larger objects. As seen in Figure 3.7, after the training, the first stack mostly produces predictions for the small objects, while the second stack mostly yields predictions for the larger ones. However, Table 3.5 shows that even though this approach sufficiently yields more structured results, it did not outperform other approaches as used in this thesis.

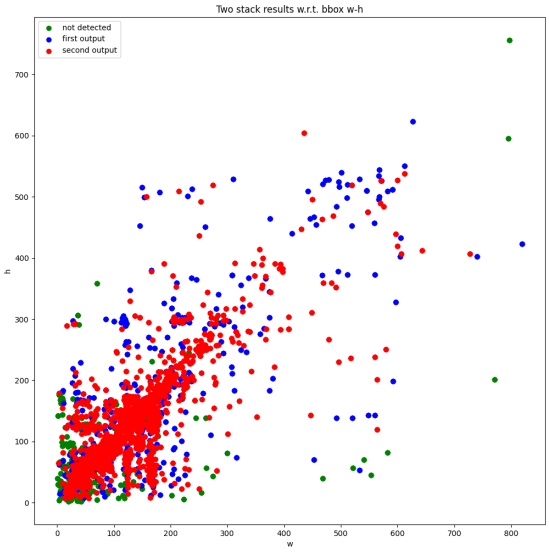
As elaborated in Table 3.5, the experiments concerning the separate cluster approaches encompass three distinct versions. The first version called as "CenterNet (2 HG) - Seperated Cluster v1" refers to a straightforward separated cluster approach that does not include the insertion of any additional blocks between two stacks or mathematical operation such as concatenation or summation. The only thing involved in the first version is assigning small objects to the output of the first stack, while the large objects are predicted at the output of the second stack. Besides, The subsequent version (CenterNet (2 HG) - Seperated Cluster v2) denotes the approach where we merge (sum) the input of first stack and its output by using 1x1 convolutional layer. The intention here is to retain features of small objects even within the deeper layers of the first stack. The final iteration is labeled as "CenterNet (2 HG) - Seperated Cluster v3" which corresponds to employing a 3x3 convolutional layer instead of the 1x1 convolutional layer as used in the second version.

3.6 Thermal Analysis

As an additional preliminary work, this thesis also examines the fusion of aligned images from visible and thermal domains within a single network to enhance the performance of object detection models. Our initial approach concatenates both RGB and thermal images, resulting in a single input with six channels. We compared the results of our network to the results of YOLOX and YOLOv3. As indicated in Table 3.2, we trained SkyDataNet in two different ways. In the first approach we use solely thermal images as the input and in our second approach, we used the concatenated images as the input. Our preliminary results denote that the model encountered challenges in extracting meaningful features from the thermal domain for SkyDataNet. Our initial deduction is that the model struggled to converge when started with the pre-trained weights obtained by training only on the visible image sets. In order to train deep architectures from scratch, we need a more diverse and scaled dataset, while running on GPU clusters for a long time. In this respect, we run a lot of experiments trained over 900 epochs, yet, none of our approaches was able to yield satisfactory predictions.



(a) The Bounding boxes of ground truths clustered by K-Means with respect to their width and height.



(b) The stacks where these boxes is predicted. The blue denotes the boxes generated by first stack output whereas the red means the boxes detected by second stack output. Lastly, the green refers the boxes which can not be detected by any stack.

Figure 3.6: The analysis of original two stacked network. The result demonstrates that any of these stacks does not focus on the objects with specified sizes.

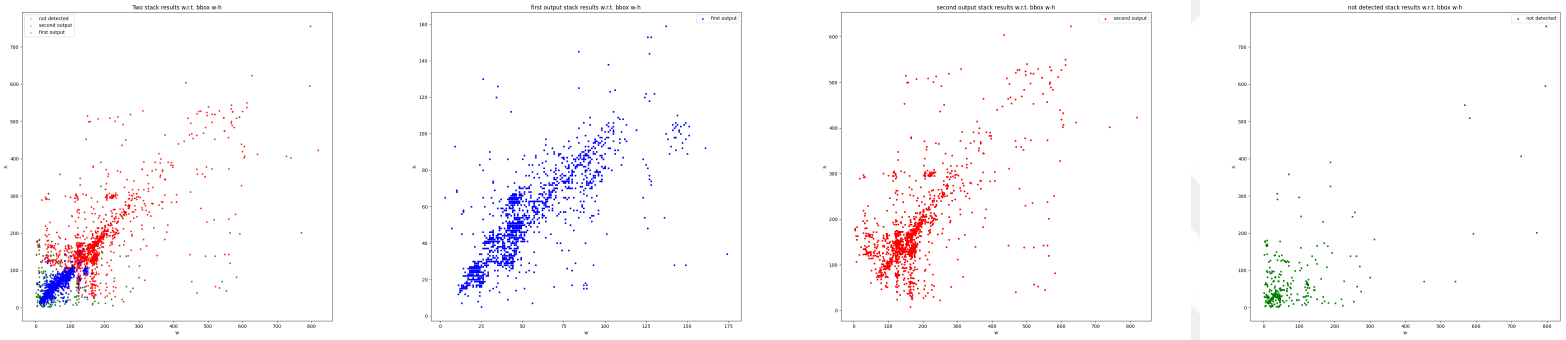


Figure 3.7: Another proposed approach to assign first stack for small objects while forwarding only large objects to second stack of HourGlass network. The first plot shows combination of other three plots. The second one is the ground truths detected by the first stack. The second one is detected by the second stack, and, the last one is the non-detected objects. The x axis is for the width of the object while the y axis is for the object's height. Small objects were mainly detected by the first stack. The second stack could not detect small object mostly. However, for the large objects, both of the stacks can detect them. The rows in Table 3.5 - named as "CenterNet (2 HG) - Seperated Cluster"- indicates quantitative results of the proposed approach in Figure 3.7. As seen Table 3.5, this approach did not outperform other approaches mentioned in this thesis.

3.7 Loss

The loss function of SkyDataNet contains three different terms. The first term is related to the offset loss (to reduce the distance between the ground truth and the predicted key-points). The second term is related to the heat-map prediction which is represented similar to the focal-loss, and the third term is related to the bounding box prediction.

The heatmap ground truth Y is defined as follows:

$$Y = R_q(p_0, p_1, ..., p_O) = \max\{G_0, G_1, ..., G_O\} \tag{3.1}$$

where R_q is the rendering function, q ϵ R^2 is pixel position at (x,y). p_i is the center point of the i^{th} object. Lastly, O is the number of existing objects in an image. By using this rendering function R_q, in a single heatmap image, we unify different peak points of Gaussian function obtained from each of the different objects.

$$G_i = \exp(-\frac{(p_i - q)^2}{2\sigma_i^2}) \tag{3.2}$$

where G_i is the 1D Gaussian equation for i^{th} object and σ_i is a measure of object size. It is defined as the radius with respect to the maximum of height and width (there are also other ways of defining that radius in the literature as in [63]). In Eq. 3.1, R_q aims to reduce the penalty of the points placed around the center point.

$$\sigma_i = K max\{w_i, h_i\} \tag{3.3}$$

where K is the scalar gain for the maximum of width and height (having a value between 0 and 1 in our experiments). w_i and h_i refer to width and height of the i^{th} object, respectively.

$$Y = R_q(p_0, p_1, ..., p_O) = \max\{G_0^{2D}, G_1^{2D}, ..., G_O^{2D}\} \tag{3.4}$$

where R_q is the rendering function, q ϵ R^2 is pixel position at (x,y). p_i is the center point of the i^{th} object. Lastly, O is the number of existing objects in an image.

$$G_i^{2D} = \exp(-(\frac{(p_{i_x} - q_x)^2}{2\sigma_{i_x}^2} + \frac{(p_{i_y} - q_y)^2}{2\sigma_{i_y}^2})) \tag{3.5}$$

where G_i^{2D} is 2D Gaussian function for the i^{th} object. It is defined with respect to varying aspect ratios.

$$\begin{aligned} \sigma_{i_x} &= K w_i \\ \sigma_{i_y} &= K h_i \end{aligned} \tag{3.6}$$

where σ_{i_x} is the Gaussian deviation for x axis while σ_{i_y} is used for y axis. The standard object detection algorithms such as [37, 64, 38], mostly use one-dimensional Gaussian which essentially consider the same parameter for both the x and y axis. However, especially for tiny objects that come with different aspect ratios, a two-dimensional Gaussian function might work better and that is what we use in this work. Therefore, in Eq. 3.4, 2D Gaussian distribution is defined in terms of the object's dimensions separately (based on width and height). Figure 3.3 shows how Equation 3.4 affects the representation of the center point on a bounding box. While the right side of Figure 3.3 refers to Equation 3.1, the left side represents the center point of 3.4. As shown in the figure, our approach yields a better representation of the aspect ratio by feeding different σ values of Gaussian loss for both axes. Additionally, we increased the gain (K) of the width and height of the bounding box over σ, in order to more precisely express the feature of the object in the periphery of a center point.

$$\mathcal{L}_h = \frac{1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & \text{if } Y_{xyc}{=}1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases} \tag{3.7}$$

where c is class, x and y are pixel index, σ_p is the standard deviation of the Gaussian kernel which is defined with respect to the dimensions of the object. $\hat{Y}_{xyc}$ is predictions indicating if the object exists. In addition, Y_{xyc} is the ground truth for fast focal loss to penalize heatmap output. It exploits Gaussian distribution to eliminate the penalty of negative points around real center key points. Furthermore, it is obtained from heatmap headers as in Figure 3.2. Finally, α is the weighting factor for easy and hard samples. β is the modulating factor for the points around the center point.

$$\mathcal{L}_s = (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2 \tag{3.8}$$

where w_i and h_i are real width and real height respectively, while $\hat{w}_i$ and $\hat{h}_i$ are predicted width and height obtained from size header as you can see in Figure 3.2. Considering regression loss for the width and height of the object, we have proposed a novel loss function by taking the square root of the predicted width and height as inspired from YOLOv3[31]. Our purpose is to diminish the rate of the penalty for large objects over the penalty for small objects. In other words, small bounding boxes are more penalizing than large bounding boxes.

$$L_{off} = \frac{1}{N}\sum_{p}((\hat{O}_{\tilde{p}_x} - (\frac{p}{r} - \tilde{p})_x)^2 + (\hat{O}_{\tilde{p}_y} - (\frac{p}{r} - \tilde{p})_y)^2) \tag{3.9}$$

where $\tilde{p}$ is the object's location indicated by heatmap indices, and p is the exact location, r is the down-ratio determined by preprocessing layers as in Figure 3.1. Finally, $\hat{O}$ is the prediction for offset outputs. We have experimented mean square error loss function in addition to the original CenterNet that utilized L_1 loss.

3.8 Metrics for Object Detection

3.8.1 Intersection over Union (IoU)

As illustrated in [65], Intersection over Union (IoU) is a metric used to evaluate the performance of object detection by comparing the ground truth bounding box to the predicted bounding box. Higher IoU implies that the predicted bounding box covers more pixels of the targeted object while excluding more background pixels, as much as possible. It is computed as follows:

$$\textbf{IoU} = \frac{\textbf{Area of Overlap}}{\textbf{Area of Union}} \tag{3.10}$$

3.8.2 Precision

As detailed in [66], precision is another metric that is used to measure the performance of the detection where a higher precision indicates that the predicted

bounding boxes (i.e. the predictions) are more precisely True, even if the model might fail to detect some of the objects in the image. It is defined as follows:

$$\textbf{Precision} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FP}} \tag{3.11}$$

where TP is the total number of True Positives and FP is the total number of False Positives.

A True Positive (TP) emerges when the prediction has a confidence score higher than the specified threshold, and sufficiently overlaps with any ground truth of the same type of objects, satisfying a higher IoU criterion than IoU threshold. A False Positive (FP) occurs when the prediction has a confidence score higher than the specified threshold but it does not overlap with any of the ground truth of the objects. Similarly, a False Negative arises when the prediction has a confidence score lower than the specified threshold, which means that the model neglects to detect the object which should have been classified as positive.

3.8.3 Recall

Recall measures how effectively the model is capable of detecting the objects in the image although it may generate false detections as elaborated in [66]. Higher recall indicates that the model does not miss the existing objects in the image even if it outputs false positives. It is defined as follows [66]:

$$\textbf{Recall} = \frac{\textbf{TP}}{\textbf{TP} + \textbf{FN}} \tag{3.12}$$

3.8.4 Precision-Recall Curve

Precision-Recall Curve is obtained by calculating the precision and recall at various confidence threshold levels according to the work [67]. By using PR Curve, we can evaluate the performance of a given model at different confidence levels. Furthermore, Average Precision (AP) implies that the model is able to sharply

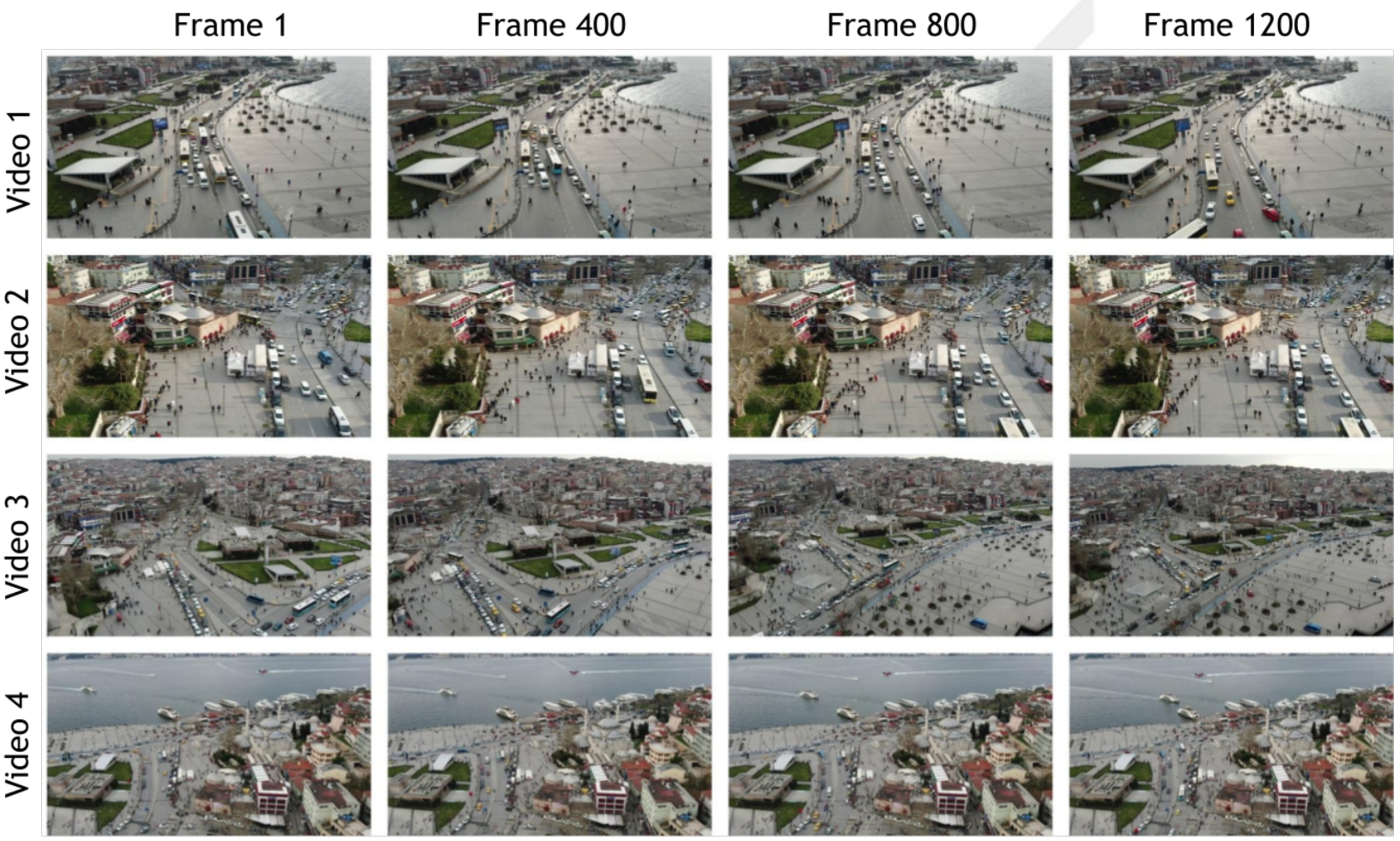


Figure 3.8: Sample frames from SkyData. As seen in the frames, The videos are captured from varying altitudes.

Figure 3.9: The segmentation masks on each class instance on sample frames. These are colored by utilizing the annotated polygons.

differentiate the positive and negative outputs as stated in [68]. Likewise, it proves the average precision of the model outputs at various confidence. In other words, It is mathematically the area under the PR curve, and is computed as follows:

$$\textbf{Average Precision (AP)} = \int_{r=0}^{1} P(r)dr \tag{3.13}$$

where r is variable for Recall, P(r) is the precision value when Recall=r. The standard AP is defined when IoU threshold is constant. However, for the metrics introduced below, AP is calculated by considering different IoU thresholds. On the other hand, Average Precision can be computed in a more simple format as in [7]. By using n confidence levels, it can be calculated in discrete format as follows:

$$\textbf{Average Precision (AP)} = \sum_{k=0}^{k=n-1} P(k) * [R(k) - R(k+1)] \tag{3.14}$$

where n is the number of confidence thresholds. $P(k)$ represents the precision value when the confidence threshold is set to the k^{th} value of confidence scores, while $R(k)$ denotes the recall value under the same condition. Additionally, the Average Precision (AP) can be computed for different IoU thresholds, such as $\mathbf{AP}_{IoU=0.5}$, which corresponds to the **AP** when the IoU threshold is 0.5. Below is a detailed explanation of $\mathbf{AP}_{IoU=0.5}$:

$$\mathbf{AP}_{IoU=0.5} = \sum_{k=0}^{k=n-1} P(k)_{IoU=0.5} * [R(k)_{IoU=0.5} - R(k+1)_{IoU=0.5}] \tag{3.15}$$

For general **AP**, we consider various IoU thresholds to evaluate the model performance in different IoU settings. This metric helps us to understand how well the model outputs overlap with ground-truth of the object in addition to the confidence score. We can define as follows:

For the general **AP**, we assess the model's performance by considering various IoU thresholds, allowing us to evaluate its performance across different IoU settings. Besides, this metric enables us to comprehend the degree of the overlap between the model's prediction and the ground truth of the object, in addition to the well-generated confidence score. The definition can be stated as follows:

$$\mathbf{AP} = \frac{1}{N} \sum_{t=0}^{t=N-1} AP_{IoU=t*s+0.5} \tag{3.16}$$

where N is the number of IoU thresholds whereas s is the step size of the threshold range. In our experiments, we set N to 10 and s is 0.05 as it is computed in standard COCO evaluation metrics. As stated by COCO evaluation formats, AP_{small} is AP for small objects whose area is smaller than 322 in terms of pixels. AP_{medium} is for medium objects smaller than 962 pixels and greater than 322. At the end, AP_{large} is for large objects greater than 962 pixels.

3.8.5 Recall-IoU Curve

Recall-IoU Curve is calculated by using a fixed number of detections in the images at various IoU thresholds. It measures recall values in different IoU thresholds, coinciding two times of area under the Precision IoU curve between IoU=0.5 and IoU=1. In addition to solely recall metric, average recall evaluates recall values in different IoU thresholds.

The Recall-IoU Curve is computed by utilizing a fixed number of detections in the images at different IoU thresholds as explained in the work [69]. It evaluates the recall values across these IoU thresholds, coinciding two times of the area under the Precision-IoU curve between IoU=0.5 and IoU=1. Therefore, [69] proposed that the Average Recall metric assesses recall values at various IoU thresholds beyond just a single IoU threshold.

$$\textbf{Average Recall (AR)} = 2 \int_{o=0.5}^{1} R(o)do \tag{3.17}$$

where R(o) is recall value when IoU threshold is o and o is the variable for IoU threshold. Similar to AP for various-sized objects, AR_{small} is AP for small objects whose area is smaller than 322 in terms of pixels. AR_{medium} is for medium objects smaller than 962 pixels and greater than 322. At the end, AR_{large} is for large objects greater than 962 pixels.

3.9 Datasets for Object Detection

We use VisDrone [70, 1], NWPU VHR-10 [71], and DOTA [72] datasets in our experiments. In general, these datasets contain aerial images captured by either satellites or unmanned aerial vehicles. Therefore, objects appear much smaller compared to the other benchmark datasets like MS COCO [7] and Pascal VOC [17]. The details of each dataset is provided in the following subsections.

3.9.1 SkyData

Currently, SkyData, available at[1], consists of 9 sequences of frames and the total number of frames in each sequence varies between 200 and 1650. There are 9 classes in total in SkyData. They are: people, bicycle, motor, pickup, car, van, truck, bus and boat. The dataset is collected via an on-board UAV camera at a public area with dense traffic of pedestrians and vehicles from different angles and at different altitudes. The number of scenes is 7 and each scene contains multiple sequences. Fig. 3.8 shows sample frames from four different sequences taken at difference scenes and Fig. 3.9 shows random sample frames with segmentation annotations from 3 scenes. The SkyData provides both thermal and RGB pairs obtained from a single platform. To align these pairs, we followed several registration methods as detailed in ablation studies. After that, we conducted several experiments for these pairs on different works. As a result, preliminary results for fused object detection algorithms are reported in Table 3.2.

We splitted the SkyData into two parts as being training and test sets. There are 8085 frames in the training set and the remaining 2275 frames in the test set out of the 10360 total frames. Test set consists of 4 fully annotated video sequences including diverse scenes in terms of camera height from the ground. Thus, We made validation more reliable to compare precision of models in different scales.

[1] `www.skydatachallenge.com`

3.9.2 VisDrone

VisDrone dataset was created by AISKYEYE team from Tianjin University, China. It contains 400 video clips along with 10,209 static images, captured by drone cameras. The images in the dataset contain various environments with annotated objects in sparse and dense scenes. The creators also launched a benchmark challenge called VisDrone Challenge, which is held every year. There are four main categories in the challenge, namely, object detection, single-object tracking, multi-object tracking, and crowd counting. One minor drawback of the dataset is that it does not contain any pixel-wise annotations, which means it is not possible to combine object detection and segmentation together without an additional labeling. At the end, VisDrone-DET 2019 provides a challenging dataset for object detection that consists of 6471 images for training and 548 images for validation. Dataset brings some challenges for detection since it involves many images with dense scenes. There exist 10 different categories for the object detection task.

3.9.3 NWPU VHR-10

NWPU VHR-10[73] is a dataset of very high-resolution remote sensing images created by Northwestern Polytechnical University. It consists of a total of 800 VHR remote sensing images in which 715 of them are RGB and the rest of the 75 are infrared. 600 out of these 800 are positive and the remaining ones are negative, that is, they don't contain any ground truth objects. Ten different object categories exist as ground truth in these images, which can be listed as airplane, ship, storage tank, baseball diamond, tennis court, basketball court, grand truck field, harbor, bridge, and vehicle. Dataset brings several challenges. First, the size of the dataset is relatively low compared to other popular datasets like ImageNet [74] and MS COCO. Second, the number of instances for each class is not evenly distributed. Finally, most of the images involve many objects that are close together. Therefore, it can be defined as dense datasaets.

3.9.4 DOTA

DOTA[72] is a large-scale aerial imagery dataset for object detection. The dataset consists of 2806 images and 15 object categories. Each image has objects with wide variety of sizes and orientations. We include DOTA in our experiments as we believe that it is suitable for UAV-based aerial applications due to including the small and large objects from UAV angle. On the other hand, DOTA provides the orientation of bounding boxes to represent the objects in a better way. Consequently, it introduces another challenge as oriented object detection algorithms.

3.10 Experimental Results

We ran our proposals and baseline methods such as CenterNet on the above-mentioned datasets. We used two different platforms to conduct the experiments in the results table. The first platform is mainly based on the original CenterNet repository [2]. Besides, the second platform utilizes the MMCV [75] and MMDetection [76] in order to fairly compare different algorithms on a single installation setup. Because the second platform helps us to develop our proposals in an easier and more feasible way in addition to ability to run the published works. In this respect, please note that if the results are listed in a single table, their relative experiments are conducted on the same platform. This is the reason why we have multiple results belonging to CenterNet (2HG) on same datasets.

Table 3.1 shows the detection results original between CenterNet [38] and our proposed architectures with Unifying Self Attention Module (USAM) and equations mentioned above. According to the Table 3.1, our preliminary results indicate that SkyDataNet shows better performance on average when average recall is concerned while CenterNet shows better performance when the average precision is concerned. At the end, results show that our proposed architecture SkyDataNet

[2] `https://github.com/xingyizhou/CenterNet`

with modified loss function surpasses the original CenterNet architecture on SkyData as well as VHR and DOTA datasets. Besides, Table 3.1 shows CenterNet architecture which uses ConvNext[48], Swin[45], SwinUNet (This is our proposed approaches used in our another publication[77]) as backbone of overall architecture. On the other hand, CentripetalNet[64] did not manage to yield comparative results on SkyData in our preliminary experiments, even though it outperforms on large scale datasets as inlike COCO.

Table 3.3 and 3.6 compare benefits of USAM blocks over original Centernet, while Table 3.4 and 3.7 are summarizing the comparison of promoted loss functions on DOTA and VHR, respectively. Besides, Table 3.5 unfortunately shows the results of experiments related to section 3.5. As a result, splitting groundtruths for different stack with respect to their size did not manage to outperform original CenterNet architecture.

We conducted several experiments for ablation studies of the object detection chapter in this thesis as seen in Table 3.8. We compared CenterNet architecture to our SkyData architecture utilizing recently published state-of-the-art backbones such as Swin, Swin-Unet, and ConvNext instead of two stacked HourGlass backbones. Here are details of those different experiments listed in Table 3.8. CenterNet(2HG) means the original CenterNet results using two stacked HourGlass as the backbone. ConvNext refers to utilizing ConvNext as the backbone instead of 2 stacked HourGlass.

Swin-UNet denotes the usage of Swin-UNet architecture as a backbone in Table 3.8. Then, Swin-UNet-AdamW-LinearLR-v1 utilizes AdamW as an optimizer and applies a linear learning rate scheduler while using Swin-UNet architecture as the backbone. Additionally, Swin-UNet-AdamW-LinearLR-v2 uses pre-trained weights that are obtained from another training on COCO. For pre-trained version, we trained 20 epochs on COCO, and then, started to train on SkyData. Furthermore, Swin-UNet-AdamW-LinearLR-v3 trains only the prediction header of architectures when the weights of Swin-Unet backbone are frozen.

Swin refers to the usage of Swin [45] as the backbone of general architecture in Table 3.8. First, Swin-AdamW-LinearLR-v1 uses Swin as a backbone instead of HourGlass without pre-trained weights for 120 epochs while other Swin-based trials are conducted for at least 600 epochs. Consequently, Swin-AdamW-LinearLR-v7 is the long-trained version of the first trials, more clearly v7 also utilizes Swin without pre-trained weights for 600 epochs. After trying multiple learning rates as a preliminary experiment, v2, v3, v4, v5 and v6 for Swin-based trials are trained using the learning rate as $5e{-}3$ despite the v1 and v7 apply $1e{-}3$ as the learning rate for the optimizer. Besides, Swin-AdamW-LinearLR-v2 utilizes pre-trained weights of Swin backbone in contrast to the first version. Additionally, Swin-AdamW-LinearLR-v3 trains the network when the HourGlass backbone has frozen pre-trained weights for the first 100 training epochs. In the other epochs, all of the weights including the weights of pretrained backbone are also updated. Similarly, Swin-AdamW-LinearLR-v4 updates all weights starting from the first epoch of training in the architecture while using pre-trained weights. Moreover, Swin-AdamW-LinearLR-v5 decreases the rate of $\mathcal{L}_s$ to strength effect of $\mathcal{L}_h$ to the architecture while freezing weights of the backbone at only the first 100 epochs whereas the later epochs update all the weights. Finally, Swin-AdamW-LinearLR-v6 does not freeze the weights while increasing the effect of $\mathcal{L}_h$ compared to v5.

Furthermore, there are also visual results indicating the capability of the models: Figure 3.10 shows the detection ground truths for a selected image from SkyDataV1 dataset. Figure 3.11 shows comparative results obtained using SkyDataNet on SkyDataV1. Figure 3.12 shows the results obtained by using the original Centernet with 2HG backbone (where 2HG refers to a backbone using two Hour-Glass stacks in a consecutive way as used in CenterNet algorithm in [38]). Figure 3.13 shows the results obtained by SkyDataNet with 2D Gaussian heat map. While Figure 3.14 corresponds to the results obtained by using SkyDataNet (1HG) with $\mathcal{L}_s$ and 2D-Gaussian heatmap, Figure 3.15 refers the detection results of original CenterNet (1HG)

Table 3.1: The detection results over SkyDataV1. This table includes different versions of our proposed architecture 'SkydataNet' and the SOTA works. The parenthesis refers the name of backbone utilized by the method. 2HG means 2 stacked HourGlass while 1HG is single HourGlass network.

	AP 0.50:0.95 all max=100	AP 0.50 all max=1000	AP 0.75 all max=1000	AP 0.50:0.95 small max=1000	AP 0.50:0.95 medium max=1000	AP 0.50:0.95 large max=1000	AR 0.50:0.95 all max=100	AR 0.50:0.95 all max=300	AR 0.50:0.95 all max=1000	AR 0.50:0.95 small max=1000	AR 0.50:0.95 medium max=1000	AR 0.50:0.95 large max=1000
SkyDataNet + 2D-Gaussian + L_s	0,44	0,72	0,47	0,32	0,72	**0,67**	0,54	0,56	0,56	0,45	0,78	**0,87**
SkyDataNet + 2D-Gaussian	0,45	0,72	0,49	0,34	**0,75**	0,64	**0,54**	**0,56**	**0,57**	0,45	**0,81**	0,85
CenterNet (2HG)	**0,46**	**0,75**	**0,50**	**0,36**	0,71	0,63	0,54	0,55	0,56	**0,46**	0,76	0,72
CentripetalNet (2HG)	0,18	0,24	0,21	0,08	0,39	0,50	0,20	0,20	0,20	0,09	0,42	0,59
CentripetalNet (1HG)	0,14	0,19	0,16	0,07	0,34	0,46	0,16	0,16	0,16	0,08	0,37	0,55
CenterNet (1HG)	0,43	0,70	0,46	0,34	0,64	0,63	0,53	0,55	0,55	0,45	0,74	0,68
CenterNet (1HG) + 2D-Gaussian + L_s	0,42	0,71	0,45	0,34	0,65	0,59	0,50	0,52	0,52	0,43	0,72	0,64
CenterNet (Swin)	0,21	0,47	0,17	0,21	0,34	0,19	0,33	0,35	0,35	0,34	0,50	0,28
CenterNet (ConvNext)	0,01	0,01	0,00	0,00	0,01	0,08	0,01	0,01	0,01	0,00	0,31	0,11
CenterNet (SwinUnet)	0,07	0,22	0,03	0,12	0,08	0,05	0,06	0,17	0,25	0,24	0,24	0,01

Table 3.2: The detection results over SkyDataV1. The results compares performances of the methods in different domains. 'RGB+Thermal' denotes that the input of architecure is concatenation of RGB and Thermal domain as six channel whereas '+Thermal' refers to three channeled input in thermal spectrum.

	AP 0.50:0.95 all max=100	AP 0.50 all max=1000	AP 0.75 all max=1000	AP 0.50:0.95 small max=1000	AP 0.50:0.95 medium max=1000	AP 0.50:0.95 large max=1000	AR 0.50:0.95 all max=100	AR 0.50:0.95 all max=300	AR 0.50:0.95 all max=1000	AR 0.50:0.95 small max=1000	AR 0.50:0.95 medium max=1000	AR 0.50:0.95 large max=1000
SkyDataNet + 2D-Gaussian + L_s	0,440	0,720	0,470	0,320	**0,720**	**0,670**	**0,540**	**0,560**	**0,560**	0,450	**0,780**	**0,870**
CenterNet (2HG)	**0,460**	**0,750**	**0,500**	**0,360**	0,710	0,630	**0,540**	0,550	**0,560**	**0,460**	0,760	0,720
SkyDataNet+RGB+Thermal	0,069	0,133	0,068	0,264	0,076	0,008	0,328	0,328	0,328	0,323	0,338	0,181
SkyDataNet+Thermal	0,039	0,075	0,036	0,202	0,580	0,000	0,238	0,238	0,238	0,220	0,293	0,013
YOLOX-Low Res+RGB+Thermal	0,249	0,414	0,246	0,135	0,455	0,552	0,309	0,314	0,315	0,182	0,537	0,597
YOLOX-High Res+RGB+Thermal	0,273	0,488	0,295	0,22	0,473	0,42	0,371	0,383	0,388	0,298	0,563	0,574
YOLOv3+RGB+Thermal	0,239	0,445	0,222	0,109	0,475	0,519	0,308	0,311	0,311	0,184	0,549	0,613

Table 3.3: The detection results over DOTA to compare SkyDataNet and CenterNet

	AP 0.50:0.95 all max=100	AP 0.50 all max=1000	AP 0.75 all max=1000	AP 0.50:0.95 small max=1000	AP 0.50:0.95 medium max=1000	AP 0.50:0.95 large max=1000	AR 0.50:0.95 all max=100	AR 0.50:0.95 all max=300	AR 0.50:0.95 all max=1000	AR 0.50:0.95 small max=1000	AR 0.50:0.95 medium max=1000	AR 0.50:0.95 large max=1000
CenterNet (2HG)	0.308	0.517	0.315	0.192	0.321	0.330	0.447	0.447	0.447	0.307	0.484	0.498
SkyDataNet (2HG) + 2D-Gaussian + AdamW + $\mathcal{L}_s$ + $\mathcal{L}_{off}$	0.334	0.550	0.348	0.229	0.350	0.359	0.495	0.495	0.495	0.349	0.538	0.520

Table 3.4: The detection results over DOTA to analyze performances of the proposed loss terms on CenterNet

	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	AP_s	AP_m	AP_l	$AR_{max=1}$	$AR_{max=10}$	$AR_{max=100}$	AR_s	AR_m	AR_l
CenterNet (2 HG)	0,346	0,565	0,366	0,231	0,395	0,378	0,128	0,354	0,482	0,332	0,550	0,510
CenterNet (1 HG)	0,342	0,575	0,358	0,214	0,376	0,372	0,128	0,352	0,474	0,328	0,533	0,498
CenterNet with 2D-Gaussian (2HG)	**0,383**	**0,612**	**0,413**	**0,245**	**0,419**	**0,423**	**0,142**	**0,380**	**0,510**	**0,354**	**0,574**	**0,557**
CenterNet with 2D-Gaussian (1 HG)	0,349	0,593	0,362	0,214	0,384	0,373	0,131	0,357	0,480	0,327	0,536	0,513
CenterNet with L_s (2 HG)	0,365	0,621	0,378	0,223	0,401	0,399	0,133	0,359	0,484	0,332	0,547	0,521
CenterNet with L_s + L_{off} (2 HG)	0,381	0,634	0,402	0,244	0,410	0,430	0,139	0,370	0,496	0,341	0,555	0,548
CenterNet with 2D-Gaussian + L_s (2 HG)	0,371	0,623	0,381	0,234	0,405	0,418	0,135	0,363	0,491	0,342	0,550	0,539
CenterNet with 2D-Gaussian + L_s (1 HG)	0,331	0,593	0,323	0,205	0,359	0,355	0,122	0,330	0,451	0,314	0,500	0,479
CenterNet with 2D-Gaussian + L_s + L_{off} (2 HG)	0,367	0,620	0,385	0,235	0,402	0,409	0,133	0,361	0,487	0,344	0,547	0,532
CenterNet with 2D-Gaussian + L_s + L_{off} (1 HG)	0,323	0,590	0,317	0,206	0,358	0,350	0,121	0,327	0,442	0,307	0,493	0,477

where AP_s, AP_m and AP_l refers to the detection score for small, medium, and large objects, respectively. HG means HourGlass stack.

Table 3.5: The detection results on DOTA to compare CenterNet and proposed approaches detailed in Section 3.5

	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	AP_s	AP_m	AP_l	$AR_{max=1}$	$AR_{max=10}$	$AR_{max=100}$	AR_s	AR_m	AR_l
CenterNet (2 HG)	0,346	0,565	0,366	0,231	0,395	0,378	0,128	0,354	0,482	0,332	0,550	0,510
CenterNet with 2D-Gaussian (2HG)	**0,383**	**0,612**	**0,413**	**0,245**	**0,419**	**0,423**	**0,142**	**0,380**	**0,510**	**0,354**	**0,574**	**0,557**
CenterNet (2 HG) - Seperated Cluster v1	0,339	0,555	0,351	0,203	0,349	0,440	0,130	0,358	0,481	0,314	0,530	0,560
CenterNet (2 HG) - Seperated Cluster v2	0,305	0,553	0,290	0,174	0,309	0,429	0,122	0,330	0,441	0,299	0,469	0,539
CenterNet (2 HG) - Seperated Cluster v3	0,314	0,501	0,307	0,181	0,319	0,421	0,123	0,331	0,444	0,296	0,466	0,537

where AP_s, AP_m and AP_l refers to the detection score for small, medium, and large objects, respectively. HG means HourGlass stack.

Table 3.6: The detection results over VHR. This table includes different versions of our proposed architecture 'SkydataNet' and the other works.

	AP 0.50:0.95 all max=100	AP 0.50 all max=1000	AP 0.75 all max=1000	AP 0.50:0.95 small max=1000	AP 0.50:0.95 medium max=1000	AP 0.50:0.95 large max=1000	AR 0.50:0.95 all max=100	AR 0.50:0.95 all max=300	AR 0.50:0.95 all max=1000	AR 0.50:0.95 small max=1000	AR 0.50:0.95 medium max=1000	AR 0.50:0.95 large max=1000
CenterNet (2HG)	0.711	0.948	0.826	0.587	0.712	0.663	0.762	0.762	0.762	0.621	0.756	0.761
SkyDataNet (2HG) + 2D-Gaussian	0.703	0.939	0.831	0.570	0.711	0.694	0.771	0.771	0.771	**0.639**	0.768	0.798
SkyDataNet (2HG) + 2D-Gaussian + AdamW	0,729	0,953	0,848	**0,598**	0,731	**0,703**	0,784	0,784	0,784	0,639	0,781	0,793
SkyDataNet (2HG) + 2D-Gaussian +$\mathcal{L}_s$ + AdamW	**0.730**	0.946	**0.850**	0.583	**0.742**	0.702	**0.792**	**0.792**	**0.792**	0.628	**0.790**	**0.812**
SkyDataNet (2HG) + 2D-Gaussian + $\mathcal{L}_s$ + $\mathcal{L}_{off}$	0.714	**0.949**	0.820	0.590	0.725	0.647	0.779	0.779	0.779	0.635	0.778	0.791

Table 3.7: The detection results over VHR to analyze performances of the proposed loss terms on CenterNet

	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	AP_s	AP_m	AP_l	$AR_{max=1}$	$AR_{max=10}$	$AR_{max=100}$	AR_s	AR_m	AR_l
CenterNet (2 HG)	0,616	0,878	0,736	0,550	0,625	0,421	0,258	0,615	0,697	0,607	0,702	0,578
CenterNet (1 HG)	0,550	0,851	0,616	0,475	0,564	0,322	0,237	0,560	0,644	0,549	0,645	0,443
CenterNet with 2D-Gaussian (2 HG)	0,641	0,896	0,778	0,530	0,663	0,414	0,256	0,627	0,712	0,584	0,727	0,528
CenterNet with L_s (2 HG)	0,623	0,921	0,758	0,564	0,639	0,437	0,245	0,602	0,684	0,628	0,694	0,510
CenterNet with L_s + L_{off} (2 HG)	**0,680**	**0,948**	**0,805**	**0,603**	**0,688**	**0,557**	**0,270**	**0,647**	**0,734**	**0,646**	**0,736**	**0,624**
CenterNet with 2D-Gaussian + L_s + L_{off} (2 HG)	0,607	0,911	0,729	0,539	0,627	0,422	0,245	0,593	0,671	0,599	0,682	0,484
CenterNet with 2D-Gaussian + L_s + L_{off} (1 HG)	0,506	0,860	0,550	0,454	0,523	0,251	0,216	0,514	0,590	0,560	0,593	0,342
CenterNet with 2D-Gaussian + L_s (1 HG)	0,506	0,864	0,547	0,455	0,523	0,263	0,217	0,517	0,594	0,572	0,593	0,355

where AP_s, AP_m and AP_l refers to the detection score for small, medium, and large objects, respectively.

Figure 3.10: The ground truth of an example frame from SkyData

Figure 3.11: The detection results obtained from SkyDataNet

Figure 3.12: The detection samples from CenterNet (2HG)

Figure 3.13: The detection results obtained from SkyDataNet + 2D-Gaussian

Figure 3.14: The detection results obtained from SkyDataNet (1HG) + 2D-Gaussian + L_s

Figure 3.15: The detection results obtained from SkyDataNet (1HG) with 2D Gaussian heat map.

Table 3.8: Additional Evaluated Results on VHR-10 to compare CenterNet with two stacked HourGlass and CenterNet using other SOTA backbones in different parameters.AdamW means the usage of AdamW[4] as an optimizer for the training. LinearLR indicates the usage of linear learning rate scheduler for each iteration.

	$AP_{0.50:0.95}$	$AP_{0.50}$	$AP_{0.75}$	AP_s	AP_m	AP_l	$AR_{max=1}$	$AR_{max=10}$	$AR_{max=100}$	AR_s	AR_m	AR_l
CenterNet(2HG)	**0,616**	**0,878**	**0,736**	**0,550**	**0,625**	0,421	0,258	**0,615**	**0,697**	**0,607**	**0,702**	**0,578**
ConvNext	0,483	0,809	0,529	0,217	0,485	**0,674**	**0,540**	0,540	0,540	0,222	0,544	0,716
Swin-UNet	0,117	0,310	0,070	0,213	0,112	0,054	0,086	0,226	0,295	0,361	0,273	0,147
Swin-UNet-AdamW-LinearLR-v1	0,103	0,272	0,054	0,188	0,100	0,043	0,068	0,193	0,247	0,316	0,229	0,094
Swin-UNet-AdamW-LinearLR-v2	0,341	0,609	0,341	0,422	0,346	0,190	0,166	0,420	0,504	0,558	0,481	0,407
Swin-UNet-AdamW-LinearLR-v3	0,276	0,538	0,239	0,308	0,280	0,157	0,143	0,374	0,464	0,517	0,431	0,359
Swin-AdamW-LinearLR-v1	0,281	0,543	0,257	0,348	0,285	0,143	0,144	0,377	0,469	0,529	0,440	0,336
Swin-AdamW-LinearLR-v2	0,372	0,636	0,407	0,372	0,388	0,165	0,175	0,439	0,542	0,537	0,525	0,407
Swin-AdamW-LinearLR-v3	0,325	0,597	0,337	0,431	0,342	0,110	0,134	0,389	0,490	0,535	0,491	0,313
Swin-AdamW-LinearLR-v4	0,186	0,418	0,139	0,236	0,193	0,057	0,107	0,291	0,379	0,429	0,344	0,235
Swin-AdamW-LinearLR-v5	0,326	0,604	0,321	0,487	0,347	0,119	0,135	0,392	0,494	0,569	0,494	0,300
Swin-AdamW-LinearLR-v6	0,446	0,724	0,507	0,497	0,457	0,208	0,19	0,495	0,590	0,633	0,584	0,397
Swin-AdamW-LinearLR-v7	0,272	0,534	0,237	0,334	0,267	0,123	0,13	0,368	0,449	0,496	0,413	0,322

where AP_s, AP_m and AP_l refers to the detection score for small, medium, and large objects, respectively.

Chapter 4

InfraGAN: A GAN-based Domain Transfer Approach for Generating Synthetic Infrared Images

Utilizing both visible and infrared (IR) images in various deep learning-based computer vision tasks has been a recent trend. Consequently, datasets having both visible and IR image pairs are desired in many applications. However, while large image datasets taken at the visible spectrum can be found in many domains, large IR-based datasets are not easily available in many domains. The lack of IR counterparts of the available visible image datasets limits existing deep algorithms to perform on IR images effectively. In this thesis, to overcome with that challenge, we introduce a generative adversarial network (GAN) based solution and generate the IR equivalent of a given visible image by training our deep network to learn the relation between visible and IR modalities. In our proposed GAN architecture (InfraGAN), we introduce using structural similarity as an additional loss function. Furthermore, in our discriminator, we do not only consider the entire image being fake or real but also each pixel being fake or real. We evaluate our comparative results on three different datasets and

report the state-of-the-art results over five metrics when compared to Pix2Pix and ThermalGAN architectures from the literature. We report up to +16% better performance in Structural Similarity Index Measure (SSIM) over Pix2Pix and +8% better performance over ThermalGAN for VEDAI dataset. Further gains on different metrics and different datasets are also reported in our experiments section.

GANs basically consist of a generator and a discriminator network where each of those networks tries to win over the other network in an adversarial manner [78]. The goal of the discriminator is to determine whether the input image is real or fake by discriminating the output of the generator from the real ones, and the goal of the generator is to produce fake images imitating the real ones so that it can fool the discriminator.

Our network uses two distinct U-Net-based architectures. The first one is used in our generator, inspired from [18], (see Figure 4.4) and the second one is used in our discriminator (see Figure 4.5) which is similar to the U-Net architecture in [79]. In both figures, we use block structures in encoder and decoder sides of the networks and each of those blocks are color-coded based on their type. In contrast to the generator network, discriminator's upsampling layer is composed of bi-linear interpolation with scale rate 2 as shown in Figure 4.5. Furthermore, its down sampling layer is made up of average pooling with stride 2 to downscale the input into 1/2 resolution. Moreover, Our discriminator utilizes attention blocks as described in Figure 4.6 to increase the spatial relationship between all pixels. More clearly, while the standard convolutional operation takes into account the pixels overlapped by the convolutional kernel windows, the attention blocks interpret relations between all points of the frame. In the figures, s is used for the stride value, f for the number of used filters, p for the padding value, 1x1, 2x2, 3x3 and 4x4 are the kernel sizes.

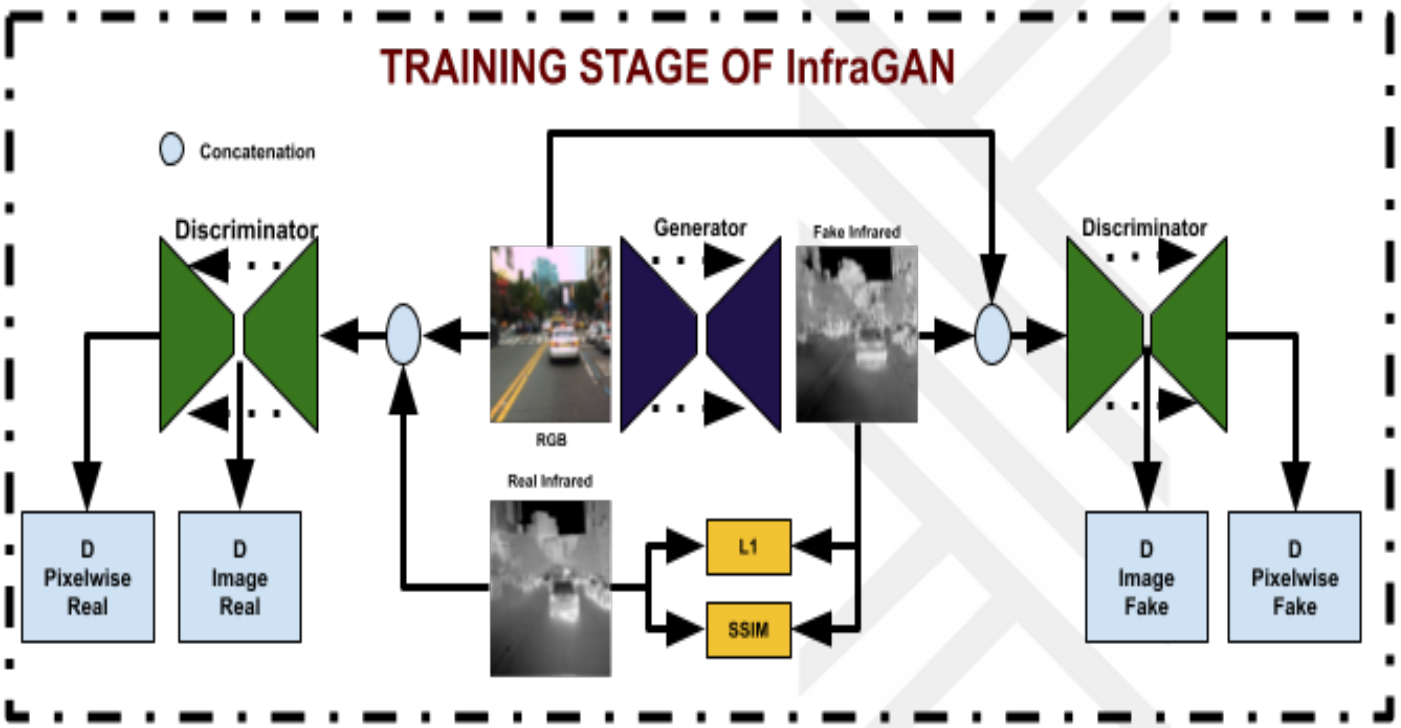


Figure 4.1: Overview of our proposed InfraGAN architecture for training stage. The details of our generator are given in Figure 4.4 and the details of our discriminator are given in Figure 4.5.

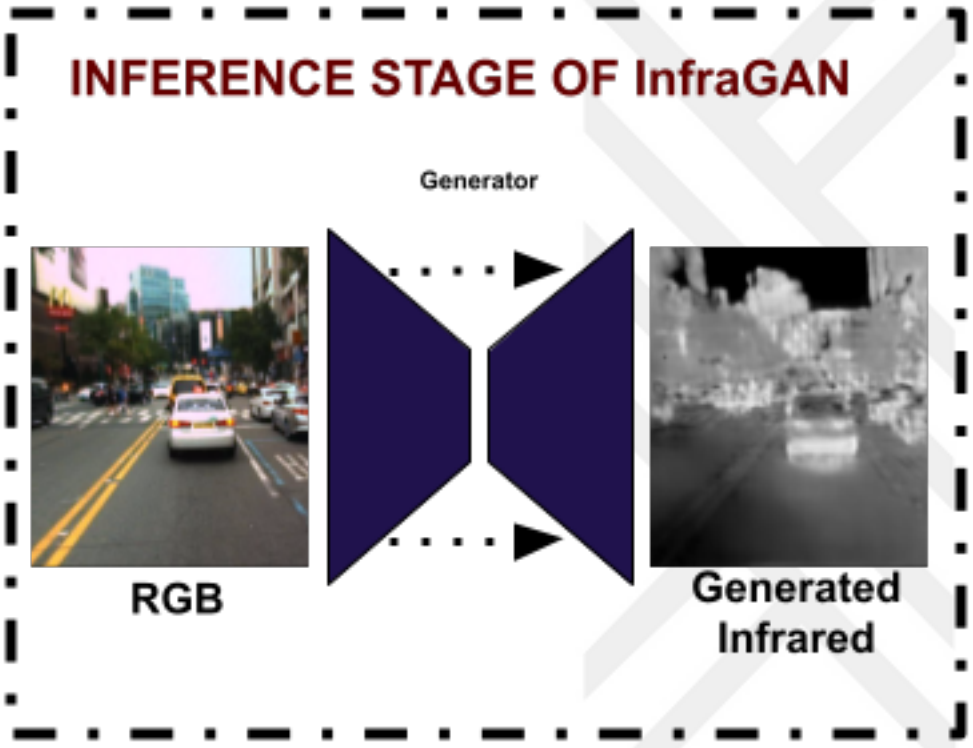


Figure 4.2: Inference stage of InfraGAN architecture. For this stage, the images from the visible domain are fed into the generator detailed in Figure 4.4.

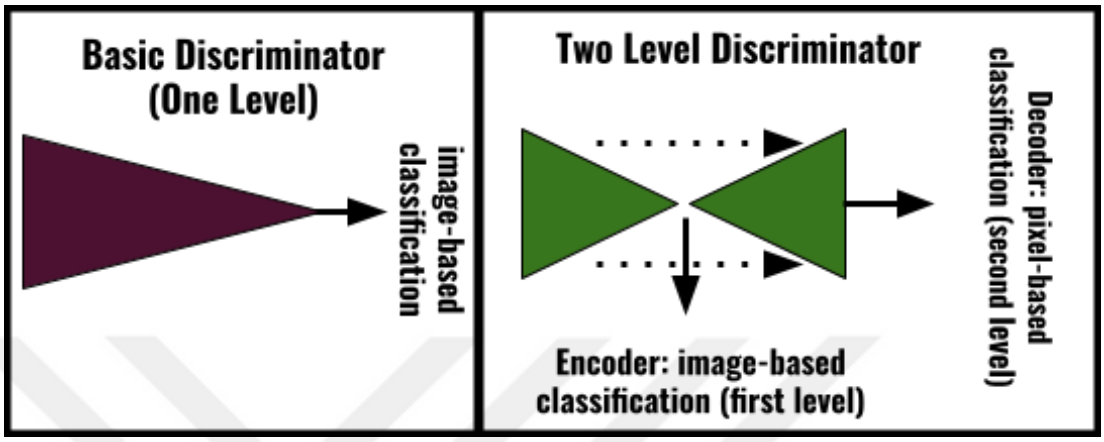


Figure 4.3: Comparison of two discriminator types: one-level (on the left) and two-level discriminators (on the right). One-level discriminator only classifies the entire image while the two-level one classifies the image and its pixels as being fake or real at the same time.

4.1 Generator Architecture

Our generator downscales images by using 2D convolutional layers with stride 2 and uses deconvolutional layers without skip connections (instead of using interpolation). These layers manage to increase the quality of IR images in the generator block thanks to learned parameters. On the other hand, the input image is normalized between 1 and -1 in our network and therefore, the generator uses a tangent hyperbolic activation function at the end to generate pixel values within the same input range.

4.2 Discriminator Architecture

In general, the discriminator architectures (based on the output format) can be classified under two main types: basic discriminators that classify only the entire input image being fake or real, and pixel-based discriminators that can classify the entire (input) image being fake or real as well as its individual pixels being

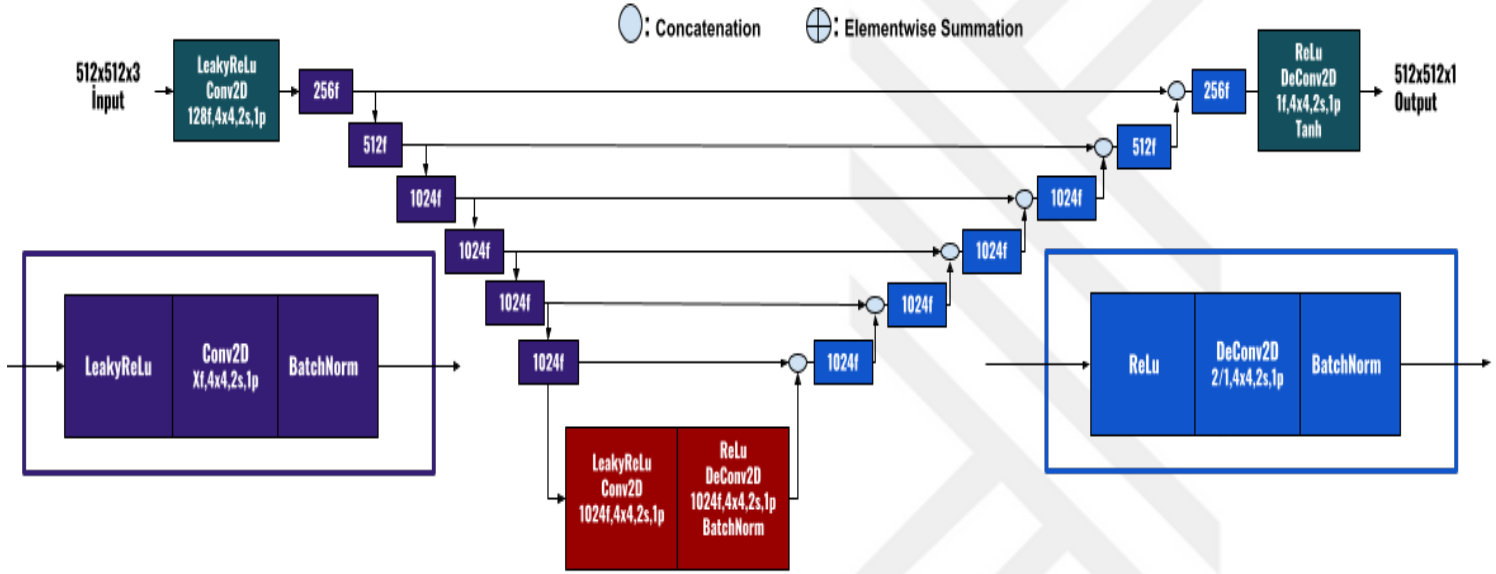


Figure 4.4: Our U-Net based generator architecture. In the figure, Xf is used to represent the total number of filters in each block where X is the total number of filters.

fake or real. See Figure 4.3 for illustrative comparisons of those two types. The basic discriminator architectures typically employ sequential convolutional layers without incorporating skip connections or upsampling layers. This design choice allows for a gradual reduction in dimensionality as the input passes through each layer. At the end, the discriminator outputs a single probability that indicates whether the given image as a whole is real or not rather than assessing individual pixels separately.

Recent literature in [79] reports that utilizing a U-Net architecture in discriminator (by classifying each pixel being fake or real) would improve the performance of overall GAN architecture. Inspired from that work, we use an architecture allowing us to classify each pixel separately in our discriminator to generate IR images. In our discriminator, we used bilinear interpolation for upsampling stages and max-pooling layer for downsampling stages to limit the number of learnable parameters, since, using upsampling and max-pooling functions reduces the required memory consumption when compared to the convolutional and deconvolutional layers (deconv with scale rate 2 and conv2d with stride 2). 1 neuron FCN is used as a normalization layer to limit the output values being between 0 and 1. Furthermore, a single attention block is used (see Figure 4.5 for the details of the attention block).

4.3 Residual Blocks

A residual block is a collection of multiple layers with skip connection(s). In our discriminator, we utilize two different residual block types. The first residual block type is used to downscale its input (green boxes in Figure 4.5), while the second type is used to upscale its input (the red boxes in Figure 4.5). Both residual blocks include, 2D convolutional layers with 3x3, 1x1 kernel size, and same padding. The red block upscales its input via interpolation before applying convolutional layers. The green block down-scales its input via average pooling after applying convolutional layers. At the end, both blocks perform channel-wise summation on two parallel branches and pass the output to next block. Next we

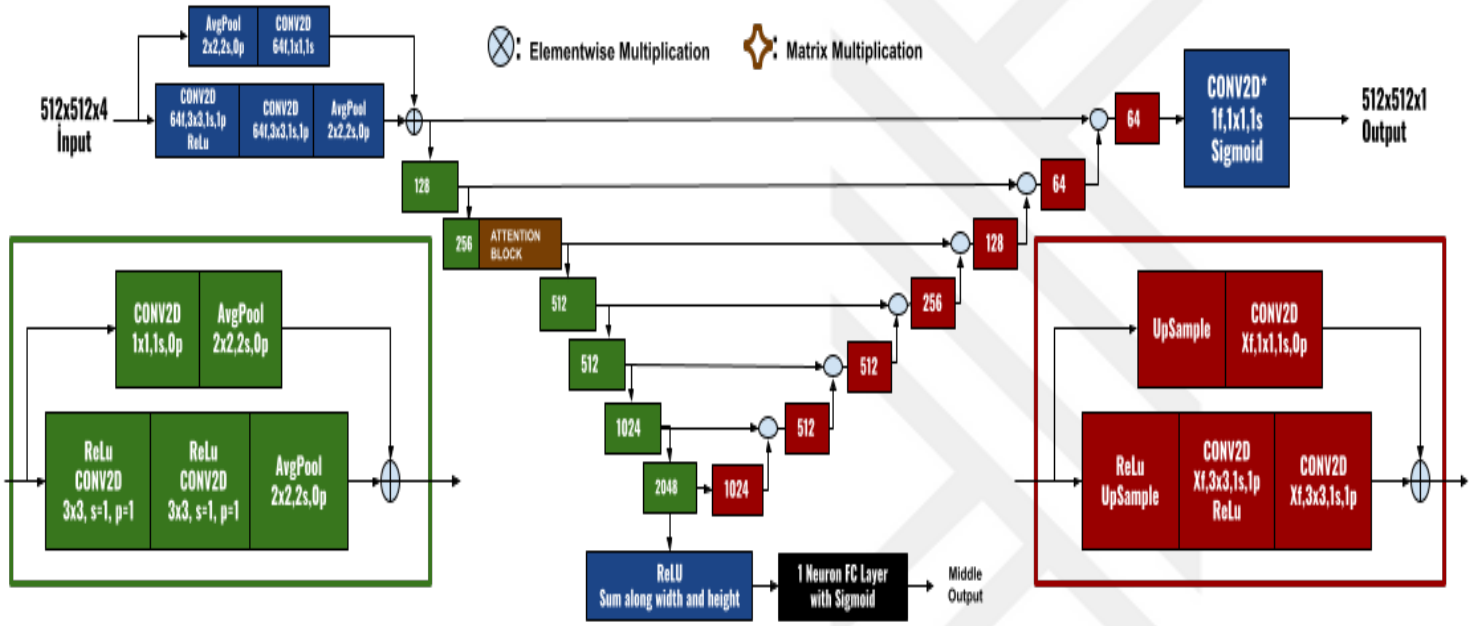


Figure 4.5: Our U-Net based discriminator architecture. It shows the discriminator architecture where the symbol $\oplus$ means element-wise summation and the symbol O means concatenation. See Figure 4.6 for the details of the attention block.

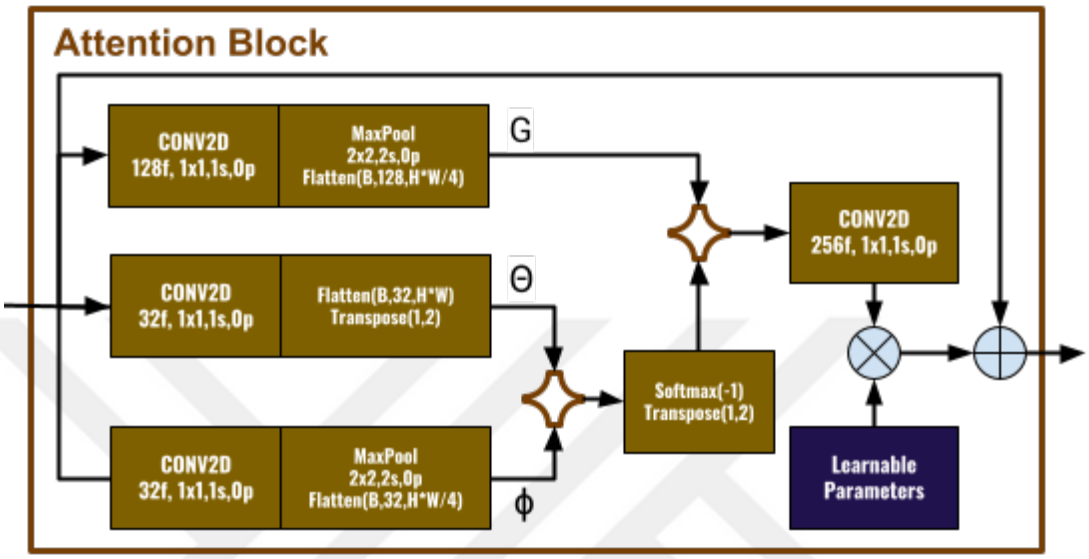


Figure 4.6: The attention block of the architecture as used in the discriminator. For further details, see Figure 4.5. ✧ refers matrix multiplication.

describe the used loss functions.

4.4 Discriminator Loss:

Our discriminator uses an encoder-decoder architecture and contains two loss terms as given in Eq. 4.1. One of those terms focuses on the entire image being fake or real (see Eq. 4.2), and the other one focuses on each pixel being fake or real (see Eq. 4.3).

$$Loss_{D^U} = Loss_{D^U_{enc}} + Loss_{D^U_{dec}} \tag{4.1}$$

where $Loss_{D^U_{enc}}$ is the loss obtained at the end of the encoder for image based loss, and $Loss_{D^U_{dec}}$ is the loss at the end of the decoder architecture for pixel based loss.

Their definitions are given below:

$$\begin{aligned} Loss_{D^U_{enc}} = &- \mathbb{E}_{x,y}[\log D^U_{enc}(X,Y)] \\ &- \mathbb{E}_x[\log(1 - D^U_{enc}(X, G(X)))] \end{aligned} \tag{4.2}$$

$$\begin{aligned} Loss_{D^U_{dec}} = &- \mathbb{E}_{x,y}[\sum_{i,j} \log[D^U_{dec}(X,Y)]_{i,j}] \\ &- \mathbb{E}_x[\sum_{i,j} \log(1 - [D^U_{dec}(X, G(X))]_{i,j})] \end{aligned} \tag{4.3}$$

where X is the input image in visible domain, Y the ground truth IR image, $E(.)$ refers to the expected value, $G(X)$ the generator's IR output, $D(X,Y)$ and $D(X,G(Y))$ are the binary outputs of the discriminator, (i,j) refers to a pixel's coordinate in both images (where both input and output has the same dimensions). The discriminator's decoder output yields probability of each pixel being real or fake separately.

4.5 Generator Loss:

Our generator uses three different terms in its loss ($Loss_G$). The first term is the standard conditional GAN loss ($Loss_{cGAN}$). Additionally, we use two more terms: structural similarity index (SSIM) based loss ($Loss_{SSIM}$) and L1 loss ($Loss_{L1}$). The reason for us to add the additional $Loss_{SSIM}$ is to ensure that the generated output will look similar to the given input image structurally.

$$Loss_G = Loss_{cGAN} + \lambda_1 * Loss_{L1} + \lambda_2 * Loss_{SSIM} \tag{4.4}$$

where λ_1 and λ_2 are the hyperparameters for loss function of generator network, and each used term is defined below. For a single image, the $Loss_{L1}$ is defined as:

$$Loss_{L1} = \frac{1}{N} \sum_{i,j} |G(X)_{i,j} - Y_{i,j}| \tag{4.5}$$

where $G(X)_{i,j}$ is the pixel value of the generated image at
(i, j) and N is the total number of pixels in the image. $Y_{i,j}$ is the pixel value of ground truth (real) IR image at (i, j). L_1 is the absolute value of the difference between two pixels. This loss enhances similarity of generated images to its real counterparts in terms of pixel values.

$$\begin{aligned} Loss_{cGAN} = &- \mathbb{E}_x[\sum_{i,j} \log([D^U_{dec}(X, G(X))]_{i,j})] \\ &- \mathbb{E}_x[\log(D^U_{enc}(X, G(X)))] \end{aligned} \tag{4.6}$$

$Loss_{cGAN}$ enhances the generator to learn complex features of the real images while trying to beat or fool the discriminator. As other GAN works justified, $Loss_{cGAN}$ is more robust to produce significant and meaningful features of the objects in the image such as edges and structural patterns. Similarly, our works also proves that $Loss_{cGAN}$ yield more impressive results in Table 4.2, 4.3, 4.4 and Figure 4.10.

$$Loss_{SSIM} = \frac{1}{m} \sum_{i=0}^{m-1} (1 - SSIM(G(X_i), Y_i)) \tag{4.7}$$

where m is the batch-size. The Structural Similarity Index (SSIM) is a metric measuring the similarity between two images: Y and $G(X)$. The SSIM ranges from 0 to 1, and a higher SSIM value means better similarity between images considering their structural components. Because of that, $Loss_{SSIM}$ is defined 1-SSIM to maximize the value. Besides, SSIM is defined as in Equation 4.8.

By using Eq. 4.6, the objective function of conditional GAN loss (cGAN) forces $D^U(X, G(X))$ to maximize in order to deceive the discriminator. That means: the higher realness score we obtained from $D^U(X, G(X))$, the more realistic image the generator can produce. The generator's loss is computed as if the generated images are classified as real by the discriminator.

4.6 Experiments

In this section, we first describe the used metrics and datasets and then present our experimental results. All our experiments were running on Intel(R) Core(TM) i9-10920X CPU @ 3.50GHz with 2 Titan RTX (24GB) GPUs. We mainly utilized PyTorch frameworks in floating point 32 precision while utilizing other main stream deep learning frameworks.

4.6.1 Datasets for Infrared Image Generation

We utilized three different datasets containing both IR and visible image pairs: VEDAI [5], FLIR[1] and KAIST [6] datasets.

4.6.1.1 VEDAI

VEDAI [5] is a dataset for vehicle detection in aerial imagery while including nine different vehicles as a separated class. In addition to these, it also has various backgrounds such as desert, road, urban and rural areas. It contains 1268 image pairs (visible and IR images). We use 1068 pairs of those for training and use the remaining 200 pairs for testing. As mentioned before, VEDAI consist of nine different classes of vehicles; car, camping-car, van, tractor,truck, plane, boat, pick-up and other category; namely. Even though it published its images in two different sizes, (1024x1024 and 512x512 pixels), we mainly utilized bigger one in our experiments.

4.6.1.2 FLIR

FLIR is another dataset containing visible-thermal image pairs published for autonomous vehicles researchers and Advanced Driver Assistance Systems (ADAS).

[1]FLIR dataset, https://www.flir.com/oem/adas/adas-dataset-form/

They also provide different conditions such as frame in a sunny day and night mode. It also helps researchers to utilize aligned thermal-rgb couples for autonomous driving. Similar to KAIST[6], it is collected from a vehicle carrying imaging equipment on itself. In our experiments, we found FLIR dataset in a grayscale domain platform by using open sources. Besides, we use 12795 image pairs for training and 839 pairs for testing in FLIR.

4.6.1.3 KAIST

KAIST [6] is another multispectral dataset that provides 11 video sequences in different circumstances such as in night mode and in sunny mode. The dataset captured and collected by imaging hardware involving a RGB camera, a thermal camera and a beam splitter. It introduces registered 95 thousand color-thermal couples in 640x480 resolution captured from a vehicle with imaging equipment. They manually annotated images in both domain by dividing following three classes: person, person, cyclist. KAIST annotations have only bounding box information instead of object's segmentation. In our experiments, we use 12538 image pairs for training and 2252 image pairs for testing from the KAIST dataset.

While in both VEDAI and KAIST datasets, the visible image comes in RGB format, in FLIR dataset the visible image comes in grayscale (single channel) format. In all of our networks, we use Adam optimization with learning rate of generator being 2.10^{-4} and learning rate of discriminator being 2.10^{-6}. Batch-size is set to 8. All networks are trained from scratch with random initialization. We set $\lambda_1 = 100$ and $\lambda_2 = 100$ in our experiments.

4.6.2 Metrics for Image Generation

4.6.2.1 Structural Similarity Index Measure (SSIM)

The Structural Similarity Index (SSIM) is used as a metric to assess the structural similarity between two images: Y and $G(X)$.

$$\begin{aligned} SSIM(Y', Y) &= \frac{2\mu_{y'}\mu_y + C_1}{\mu_{y'}^2 + \mu_y^2 + C_1} \cdot \frac{2\sigma_{y'}\sigma_y + C_2}{\sigma_{y'}^2 + \sigma_y^2 + C_2} \cdot \frac{\sigma_{y'y} + C_3}{\sigma_{y'}\sigma_y + C_3} \\ &= \frac{2\mu_{y'}\mu_y + C_1}{\mu_{y'}^2 + \mu_y^2 + C_1} \cdot \frac{\sigma_{y'y} + C_3}{\sigma_{y'}^2 + \sigma_y^2 + C_2} \end{aligned} \tag{4.8}$$

where C_1, C_2 and C_3 are the parameters to ensure the stability of divisions [80]. We set C_3=$C_2/2$ while $C_1 = (0.01 * L)^2$ and $C_2 = (0.03 * L)^2$ where L is the range of the image pixels. As described above, Y' is the generated output (IR) image $G(X)$ while Y is the ground truth (IR image). $\sigma_{y'}$ and σ_y are modified standard deviations for Y' and Y; $\mu_{y'}$ and μ_y are the modified mean values of their respective images. Finally, $\sigma_{y'y}$ refers to modified co-variance. Further details on computing SSIM can be found in [80].

4.6.2.2 Mean Structural Similarity Index Measure (MSSIM)

We evaluate the performance of each network over five different metrics including Structural Similarity Index Measure (SSIM, see Eq. 4.8), Mean Structural Similarity Index Measure (MSSIM), Learned Perceptual Image Patch Similarity (LPIPS), L1 (pixel by pixel, see Eq. 4.5) norm and Peak Signal-to-Noise Ratio (PSNR). Those metrics are briefly explained below.

$$MSSIM(Y', Y) = \frac{1}{K} \sum_{y_i', y_i}^{K} SSIM(y_i', y_i) \tag{4.9}$$

MSSIM is the mean SSIM value over the downscaled versions of the images [80] where y_i' and y_i are downscaled versions of generated (Y') and real (Y) infrared

images. K is the number of down-scaling factors (we used $2^1, 2^2, 2^3, 2^4, 2^5$ values for K=5 in this work).

4.6.2.3 Learned Perceptual Image Patch Similarity (LPIPS)

Learned Perceptual Image Patch Similarity (LPIPS) [81] measures the Euclidian distance between the feature vectors of both images. To compute the metric, the comparative features are obtained from a CNN-based backbone pretrained on ImageNet [2] (in our experiments we used AlexNet [20]). The work in [81] reported that LPIPS manages to favorably evaluate closeness rate between target patch and reference patch in terms of human judgment compared to other traditional metrics such as SSIM and PSNR by using deep networks and their deep features.

$$LPIPS(Y', Y) = \sum_{l}^{L} \frac{1}{H_l W_l} \sum_{h_l, w_l} [f_l(Y')_{h_l, w_l} - f_l(Y)_{h_l, w_l}]^2 * \omega_l \tag{4.10}$$

where $f_l(Y'), f_l(Y) \in \mathbb{R}^{H_l x W_l x C_l}$ denote the normalized features derived from the l^{th} layer of the pre-trained CNN backbone. C_l is the channel size for l^{th} layer. H, W are the height and width respectively. $\frac{1}{H_l W_l} \sum_{h_l, w_l}$ corresponds to the spatial average function as defined in [81]. h_l, w_l show pixel's coordinate for l^{th} layer. Moreover, $\omega_l \in \mathbb{R}^{C_l x 1}$ refers to the trained weights of LPIPS and its main purpose is reducing C_l value to one, before computing spatial average. The multiplication operator (*) is used as linear matrix multiplication. Finally, L is the number of used layers for evaluation.

4.6.2.4 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) represents how far the generated image deviates from the real image in terms of signal-noise ratio. It also assesses the quality of generation of IR images when compared to real images, which means that approximation to human perception of image generation quality in other words. At

the end, PSNR is mathematically defined as follows:

$$PSNR(Y', Y) = 10 \log \frac{MAXVALUE}{MSE(Y', Y),}$$

where MSE is defined as: (4.11)

$$MSE(Y', Y) = \frac{1}{HW} \sum_{i=0}^{H} \sum_{j=0}^{W} (Y'(i,j) - Y(i,j))^2$$

MAXVALUE is set to 1 (normalized maximum), as the ground truth and the pixel values in the generated image are normalized between -1 and 1. PSNR indicates the quality of the reconstruction of the IR images from the visible images [82]. H, W are the height and width of the image respectively.

4.7 Results

In this section, we compare our InfraGAN's performance to Pix2Pix [19] using residual blocks in its generator, to UNet, and to the ThermalGAN implementation[2] from [18]. In ThermalGAN, there is an extra input which is called as temperature vector T, (see [18] for the definition of vector T), which helps network to learn thermal color spectrum better. However, to have fair comparison in our algorithms (since none of the other algorithms use additional input vector), and since we do not have ground truth T vectors in our used datasets, we used the version that does not consider the input vector T for ThermalGAN in our experiments. We also study the performance of InfraGAN without using the SSIM loss term and present the results of that architecture as well in our results.

Table 4.2 compare overall performance of those four different architectures (ThermalGAN, Pix2Pix, UNet and InfraGAN) in five different metrics over VEDAI dataset whereas Table 4.3 and Table 4.4 asses the improvements on KAIST[6] and FLIR, respectively. Furthermore, we also include different versions of InfraGAN in the table (see Table 4.1 for the difference between different InfraGAN versions). As shown in Table 4.2 and Table 4.3, the VEDAI and KAIST

[2] `https://github.com/vlkniaz/ThermalGAN` (last accessed on December 2022)

Table 4.1: Network definitions based on their configurations, where $\mathcal{L}$ refers to the used loss function and A.F. means activation function. Encoder and Decoder is listed for the generator. This table also compares run times of different networks on KAIST dataset.

Algorithms	$\mathcal{L}_{cGAN}$	$\mathcal{L}_{L1}$	$\mathcal{L}_{MSE}$	$\mathcal{L}_{SSIM}$	A.F. in Encoder	A.F. in Decoder	A.F. in Discriminator	Sec/Epoch	FPS
ThermalGAN [18]	✓	✓			LeakyReLU	ReLU	ReLU	444 sec	328.06
UNet		✓		✓	LeakyReLU	ReLU	ReLU	936 sec	341.36
Pix2Pix [19]	✓	✓			ReLU	ReLU	ReLU	1104 sec	264.95
INFRAGANv1	✓		✓		LeakyReLU	ReLU	ReLU	2148 sec	340.37
INFRAGANv2	✓	✓			LeakyReLU	ReLU	ReLU	2172 sec	342.12
INFRAGANv3	✓	✓			LeakyReLU	ReLU	LeakyReLU	2220 sec	340.76
INFRAGANv4	✓	✓			LeakyReLU	LeakyReLU	ReLU	2244 sec	341.56
INFRAGANv5	✓	✓		✓	LeakyReLU	ReLU	ReLU	2256 sec	342.29
INFRAGANv6	✓	✓			LeakyReLU	LeakyReLU	LeakyReLU	2252 sec	341.56

results for each algorithm are higher on average than the results obtained for the FLIR dataset in Table 4.4. We think that it is mainly because of the format difference of the datasets since the FLIR dataset is the only dataset we used where the visible image was present in grayscale, while the other two datasets provide the visible image in the RGB format. Consequently, we believe that all the networks could not use the color information in FLIR experiments yielding lower results on average when compared to the other two datasets. Overall, in all five metrics, our InfraGAN architecture yielded the best results on both KAIST and VEDAI datasets when compared to the other GAN-based networks. While simple U-Net architecture (that is trained without using a GAN loss) yielded slightly higher results on SSIM, it yielded lower results on the other metrics on average. The table also demonstrates that using also the SSIM loss in our network helped gaining higher performance (compare InfraGANv2 results to InfraGANv5 results).

Table 4.5, Table 4.6 and Table 4.7 show individual class performances of each algorithm on the KAIST dataset namely Cyclist, People, and Person. The tables are arranged in the order corresponding to the mentioned classes. The validation in these tables are conducted by using the images that include corresponding objects while discarding other images. These tables indicate that InfraGAN outperformed all other algorithms across all metrics in all classes seperately. Moreover, the utilization of SSIM loss led to improvements in all results, with the single exception of the PSNR value in the people classes.

Table 4.8, 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 and 4.15 express the performance of each algorithm for specific classes on the VEDAI dataset. The test dataset in VEDAI has eight classes: car, truck, pickup, tractor, camping car, boat, van and other. Each column in the table shows the five metric in the format of: SSIM / MSSIM / LPIPS / L1 / PSNR for each class.

According to the data presented in Table 4.8, our approach has proven superior performance, outperforming Pix2Pix by up to 14% in terms of SSIM metric and 22% in terms of MSSIM metric. Furthermore, our method prevailed all other mentioned works across all evaluated metrics in the images that include Car object in KAIST dataset.

Similar to Table 4.8, InfraGAN has also surpassed ThermalGAN and Pix2Pix across the all metrics on the each subset of KAIST dataset as seen in Table 4.9, 4.10, 4.11, 4.12, 4.13, 4.14 and 4.15. These tables justify that InfraGAN and its version with SSIM loss is able to generate more similar images to real ones from infrared domain, as evidenced by numerous mathematical metrics, namely; SSIM, MSSIM, LPIPS, L1 and PSNR.

Table 4.1 summarizes our ablation study on computation time and on the use of activation function. The term FPS means frame per second and computed as the time required by algorithm to test a given image (where image resolution is 512x512). Sec/Epoch means the training time (in seconds) per epoch on KAIST dataset. Since we used different versions of our InfraGAN in the table where we changed the hyperparameters including the used activation function and the used loss functions, we named each used version seperately. In total, we listed six different versions of InfraGAN in the table. Additionally, Table 4.16 shows comparative results of using L1 vs. using Mean Square Error (MSE) in InfraGANv2 on two different datasets. The result proves that MSE yields slightly better performance in spite of its heavy computational cost.

Figure 4.7, Figure 4.8 and Figure 4.9 compare qualitative results obtained from three different algorithms, namely Pix2Pix, ThermalGAN and InfraGAN (ours) for selected (sample) two pairs from each dataset. In the figures, the rows of Figure 4.7 show the results of each compared algorithms in this thesis for the image pairs sampled from FLIR. Moreover, the rows of Figure 4.8 show results for the sample image pairs from KAIST. Lastly, the rows of Figure 4.9 show the results for the sample image pairs from VEDAI datasets, respectively. Some of our results yield checkerboard artifact and we think one potential reason might be due to the use of trainable deconvolutional layers that are used instead of upscaling layers.

As seen in Figure 4.7, our algorithms is capable to produce the images that closely resemble real infrared pairs. Notably, when examining the edge of the windows and the shape of person crossing the croad, they saliently preserve realistic patterns and shape such as structural edges, including vertices of the objects

in comparison with the other works.

As depicted in Figure 4.8, our works manages to generate more natural-looking objects in the images from KAIST dataset. Particularly, when assessing the flagstone and lane on the road, our algorithms remarkably enhance their visibility in contrast to others. Additionally, the buildings in the generated images from our algorithms exhibit more regular shape along with their well-designed fold. Turning our attention to the vehicles, we are able to produce less deformed cars so that our algorithms might make object detection algorithms more robust.

As illustrated in Figure 4.9, our InfraGAN exceeds the other works in terms of visual results, significantly when dealing with small objects and the tonal characteristics of particular patches in the images from the VEDAI dataset. However, it is should be particularly underlined that since infrared images in VEDAI are obtained from a range closer to the visible domain, the distinction between the all approaches may not be visually obvious enough.

Figure 4.10 qualitatively compares the results of UNet to GAN-based networks on three sample images from KAIST. As it can be seen in the figure, UNet results are more blurry when compared to the results of GAN-based networks.

After the time, this work [83] completed, the researchers published other GAN based methods such as [3] for domain translation. This thesis included results of [3] to compare to our solution. Thus, Figure 4.11 shows comparative results between InfraGAN and UGATIT on VEDAI dataset. Table 4.2 also involves results of UGATIT. As a consequence, these results justified that the recent SOTA works without pixel based loss functions such as L1, MSE and SSIM like UGATIT is not able to outperfrom the previos works even though it leverages the recent developments such as transformers. On the other hand, instead of domain translation problem, the recent GAN based solutions focuses on different problems such as synthetic image generation and text-to-image generation.

As a general comment for future works of GAN approaches for domain translation, by the time the future methods empowers the training of GAN architectures,

Table 4.2: Comparison of different algorithms on VEDAI dataset [5]. Best result is shown in bold for each metric over all classes for each dataset.

VEDAI dataset [5]	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]	0.72	0.66	0.066	0.175	19.84
ThermalGAN [18]	0.80	0.83	0.019	0.068	27.44
UGATIT [3]	0.18	0.20	0.087	0.215	17.60
U-Net	**0.89**	0.87	0.013	0.066	27.96
InfraGANv1	0.86	0.87	0.013	0.058	28.74
InfraGANv2	0.79	0.84	0.030	0.065	27.64
InfraGANv3	0.88	**0.90**	**0.011**	**0.055**	29.24
InfraGANv4	0.88	**0.90**	**0.011**	**0.055**	**29.26**
InfraGANv5	0.86	0.88	0.013	0.056	28.98
InfraGANv6	0.88	**0.90**	**0.011**	0.056	29.21

the unsupervised approaches can replace the pixel based approaches that are detailed in this thesis. Because, as seen in the visual results, especially in night mode, the pixel based methods may failed to produce features of objects that can be seen apparently in night mode. Yet, in many tasks such as text-to-image generation utilizing GANs, unsupervised methods are able to predict features that are not included in input. In this respects, in the future, these methods may be capable of producing features of objects in nightmode while the its rgb pair does not reflect anything about itself.

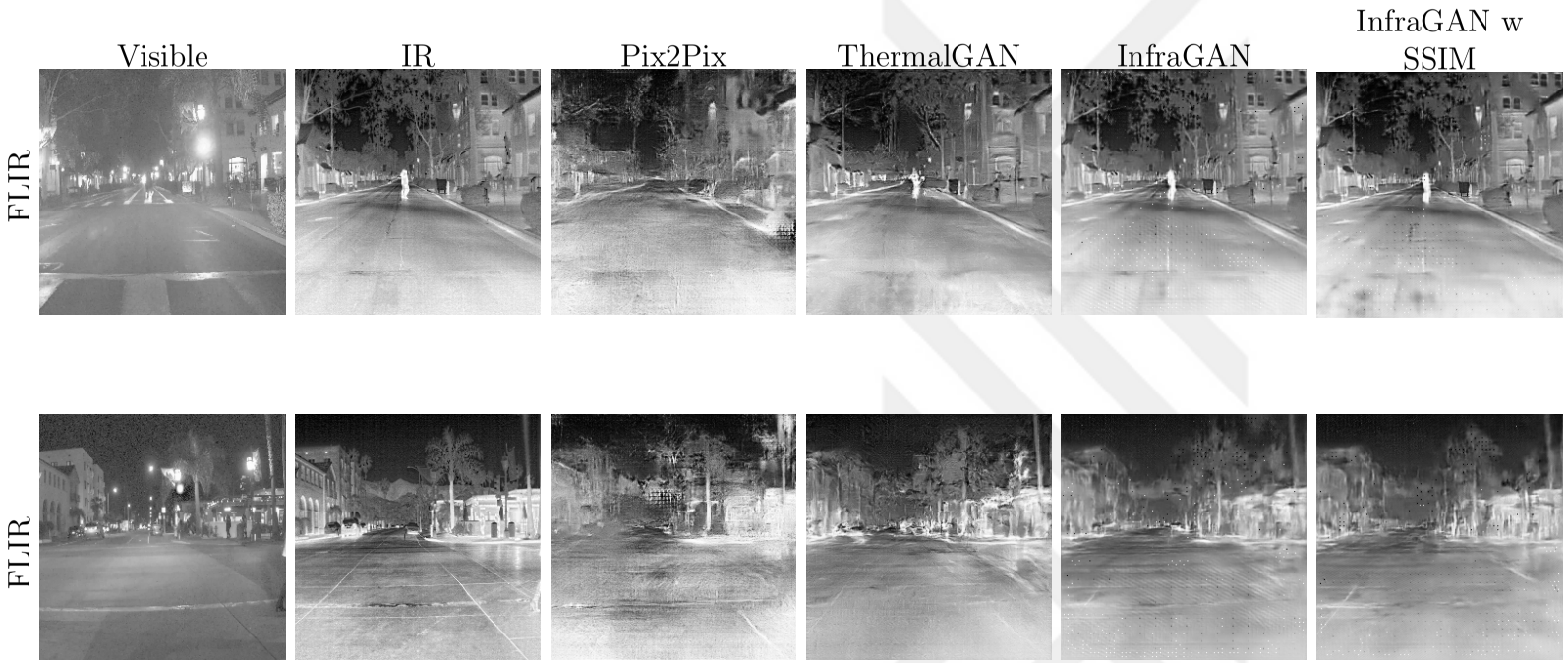


Figure 4.7: Comparative results of multiple algorithms on sample image pairs from FLIR dataset. The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively.

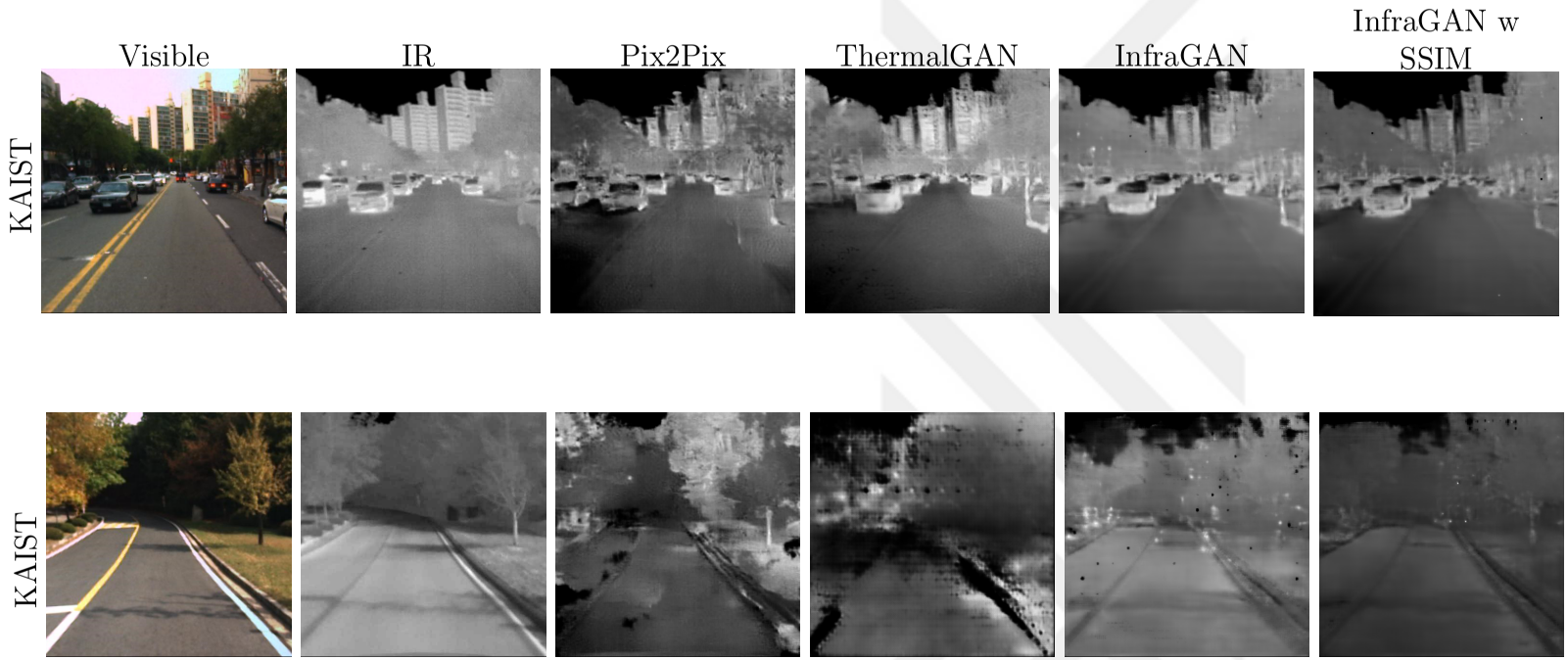


Figure 4.8: Comparative results of multiple algorithms on sample image pairs from KAIST dataet. The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively.

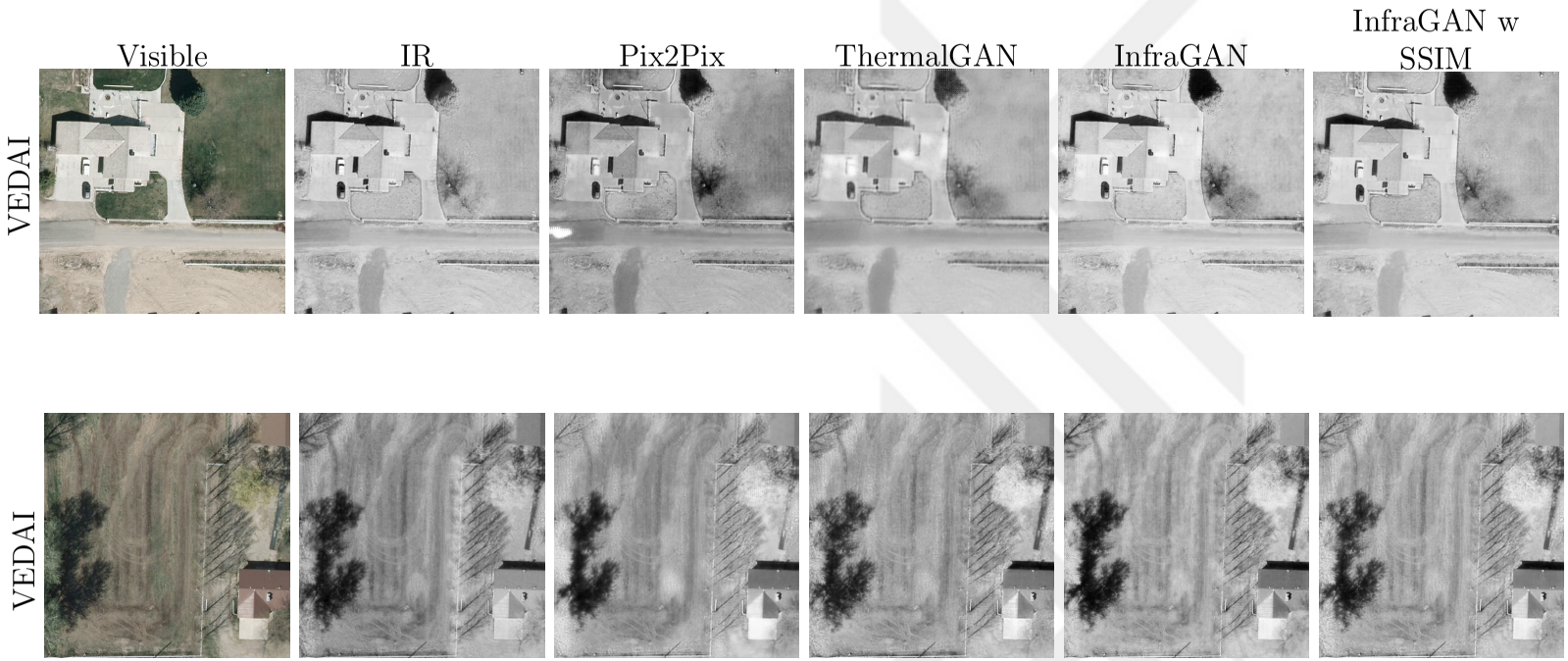


Figure 4.9: Comparative results of multiple algorithms on sample image pairs from VEDAI dataset. The first two columns are the real images (visible and IR). The 3rd through 6th column images are the results from Pix2Pix, ThermalGAN, InfraGAN without SSIM loss and InfraGAN with SSIM loss, respectively.

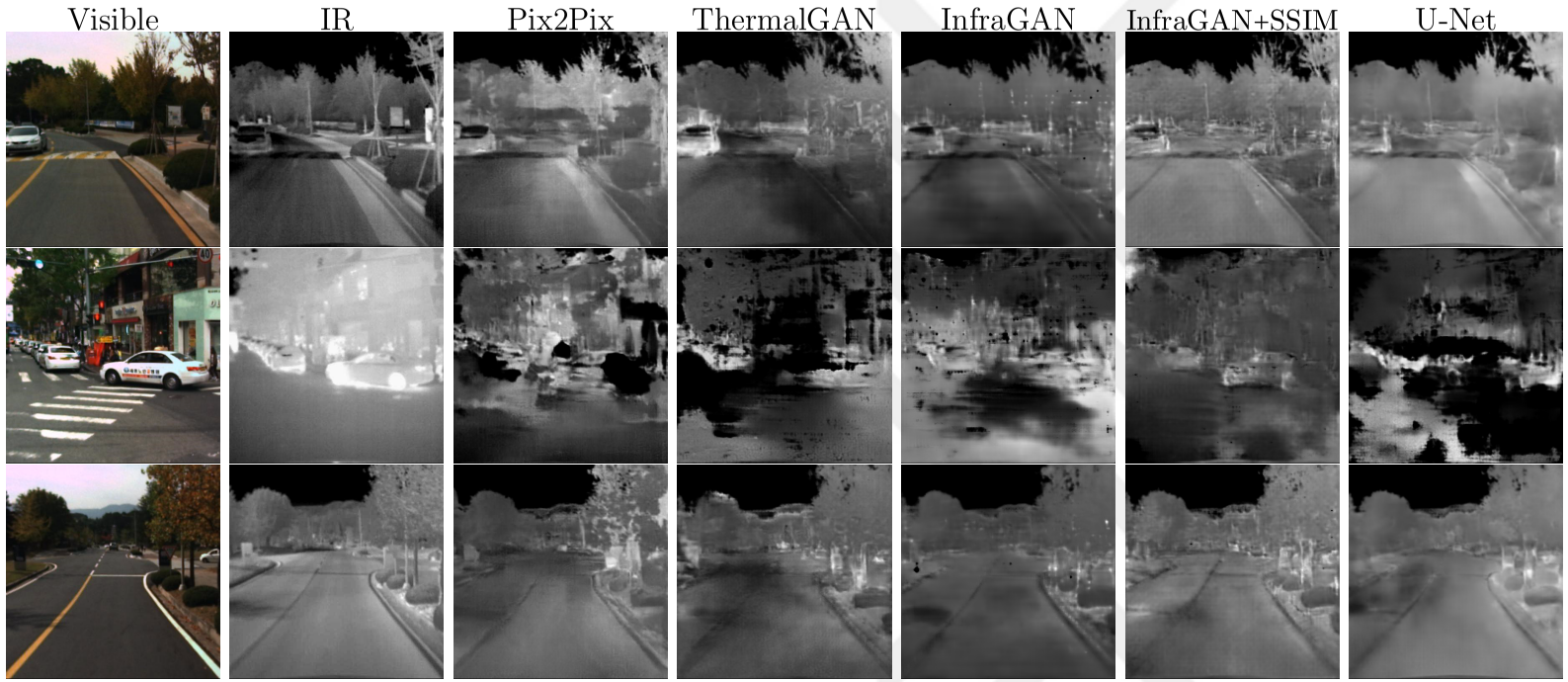


Figure 4.10: Comparative results between simple U-Net generator and other GANs to show the difference of using GAN loss. These qualitative results show that using GAN architecture can help improving the generated image quality. In the figure, the first two columns show the original images, from third column to sixth column shows GAN-based networks' results and the last column shows U-Net results on three samples.

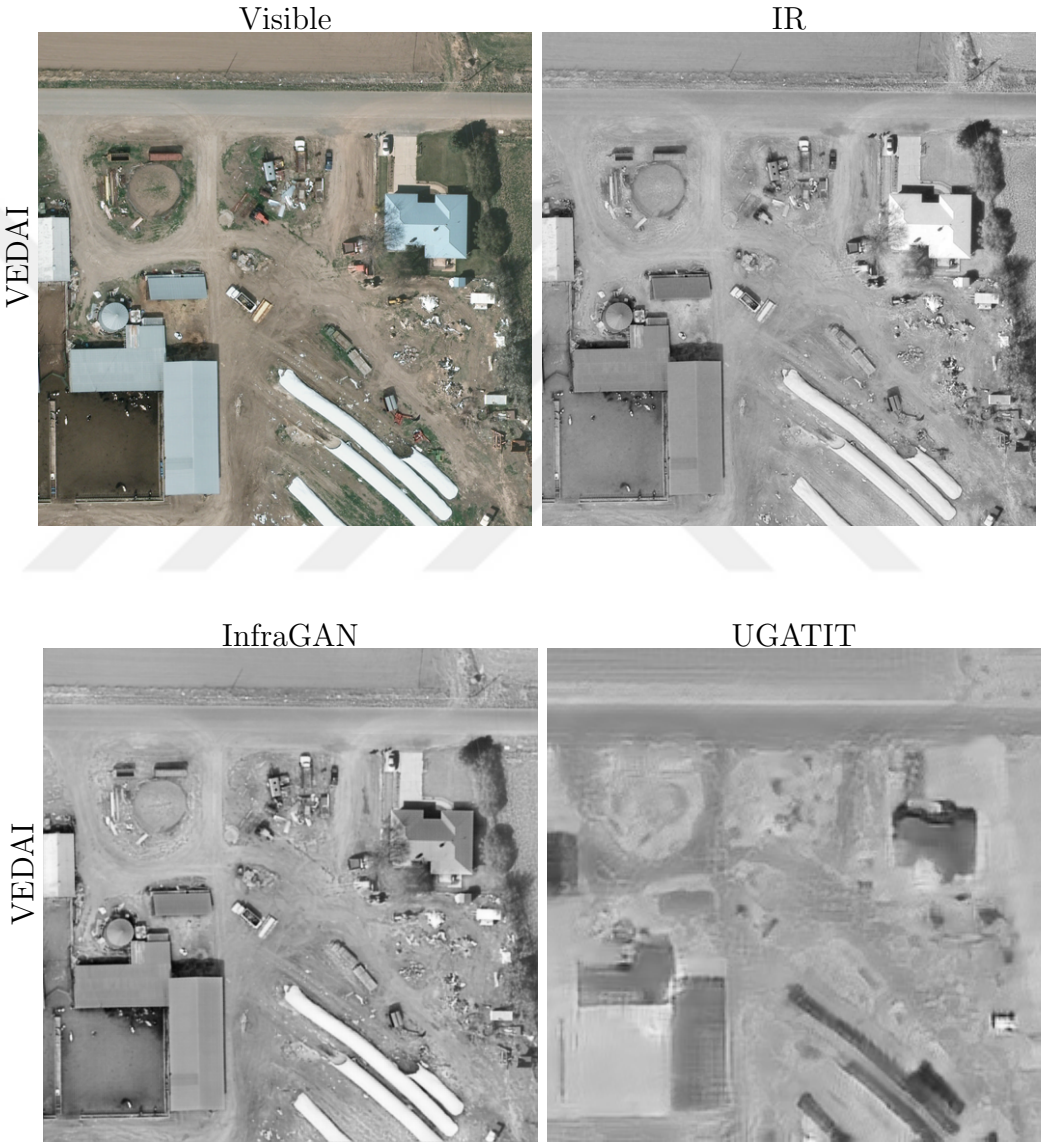


Figure 4.11: Comparative results between InfraGAN and U-GAT-IT[3]. As seen in the lower right corner, U-GAT-IT[3] does not produce the objects and edges in the image in high resolution and qualitiy.

Table 4.3: Comparison of different algorithms on KAIST dataset [6]. Best result is shown in bold for each metric over all classes for each dataset.

KAIST dataset [6]	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]	0.69	0.55	0.196	0.137	21.25
ThermalGAN [18]	0.66	0.51	0.242	0.165	19.74
U-Net	**0.78**	0.64	0.172	0.137	21.75
InfraGANv2	0.76	0.67	0.167	0.123	22.81
InfraGANv3	0.76	**0.68**	**0.157**	0.123	22.89
InfraGANv5	0.77	**0.68**	0.165	0.122	22.87
InfraGANv6	0.76	0.67	0.159	**0.121**	**22.97**

Table 4.4: Comparison of different algorithms on FLIR dataset. Best result is shown in bold for each metric over all classes for each dataset.

FLIR dataset	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]	0.13	0.38	0.203	0.228	16.51
ThermalGAN [18]	0.17	0.43	0.192	0.209	17.30
U-Net	**0.31**	**0.54**	0.188	0.169	**19.29**
InfraGANv1	0.24	**0.54**	0.164	0.171	19.23
InfraGANv2	0.25	0.53	0.173	0.171	19.10
InfraGANv3	0.26	**0.54**	0.159	0.169	19.24
InfraGANv4	0.26	**0.54**	**0.158**	0.170	19.21
InfraGANv5	0.26	**0.54**	0.175	**0.168**	19.28

Table 4.5: Comparison of four different algorithms on the KAIST dataset for Cyclist class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

KAIST dataset [6]	Cyclist				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix	0.70	0.54	0.196	0.155	20.63
ThermalGAN	0.66	0.51	0.243	0.176	19.33
INFRAGAN w/o SSIM	0.78	0.69	0.154	0.125	23.06
INFRAGAN	**0.78**	**0.70**	**0.147**	**0.118**	**23.29**

Table 4.6: Comparison of four different algorithms on the KAIST dataset for People class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

KAIST dataset [5]	People				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix	0.69	0.55	0.196	0.144	20.96
ThermalGAN	0.65	0.52	0.237	0.174	19.43
INFRAGAN w/o SSIM	0.76	0.68	0.163	0.131	**22.44**
INFRAGAN	**0.76**	**0.68**	**0.161**	**0.130**	22.30

Table 4.7: Comparison of four different algorithms on the KAIST dataset for Person class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

KAIST dataset [5]	Person				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix	0.68	0.54	0.199	0.143	21.00
ThermalGAN	0.65	0.52	0.235	0.168	19.66
INFRAGAN w/o SSIM	0.75	0.66	0.173	0.131	22.23
INFRAGAN	**0.75**	**0.66**	**0.170**	**0.128**	**22.40**

Table 4.8: Comparison of four different algorithms on the VEDAI dataset for Car class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Car				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.72	0.67	0.065	0.161	20.39
ThermalGAN [18]:	0.81	0.85	0.019	0.064	27.98
InfraGAN w/o SSIM:	0.79	0.85	0.029	0.061	28.11
InfraGAN:	**0.86**	**0.89**	**0.013**	**0.053**	**29.43**

Table 4.9: Comparison of four different algorithms on the VEDAI dataset for Truck class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Truck				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.74	0.70	0.056	0.173	20.09
ThermalGAN [18]:	0.82	0.86	0.017	0.065	27.56
InfraGAN w/o SSIM:	0.81	0.87	0.026	0.061	27.90
InfraGAN:	**0.87**	**0.90**	**0.013**	**0.057**	**28.72**

Table 4.10: Comparison of four different algorithms on the VEDAI dataset for Pickup class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Pickup				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.73	0.69	0.061	0.169	20.12
ThermalGAN [18]:	0.82	0.84	0.019	0.067	27.51
InfraGAN w/o SSIM:	0.80	0.85	0.029	0.066	27.51
InfraGAN:	**0.86**	**0.89**	**0.014**	**0.057**	**28.87**

Table 4.11: Comparison of four different algorithms on the VEDAI dataset for Tractor class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Tractor				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.71	0.67	0.065	0.162	20.23
ThermalGAN [18]:	0.80	0.83	0.020	0.70	27.21
InfraGAN w/o SSIM:	0.78	0.84	0.029	0.064	27.51
InfraGAN:	**0.85**	**0.88**	**0.014**	**0.056**	**28.97**

Table 4.12: Comparison of four different algorithms on the VEDAI dataset for Camping Car class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Camping Car				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.72	0.66	0.060	0.166	20.10
ThermalGAN [18]:	0.80	0.84	0.018	0.065	27.81
InfraGAN w/o SSIM:	0.79	0.85	0.029	0.064	27.77
InfraGAN:	**0.86**	**0.89**	**0.013**	**0.054**	**29.49**

Table 4.13: Comparison of four different algorithms on the VEDAI dataset for Boat class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Boat				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.75	0.79	0.050	0.109	23.45
ThermalGAN [18]:	0.82	0.86	0.019	0.063	27.69
InfraGAN w/o SSIM:	0.80	0.86	0.030	0.067	26.64
InfraGAN:	**0.86**	**0.86**	**0.018**	**0.063**	**27.53**

Table 4.14: Comparison of four different algorithms on the VEDAI dataset for Van class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Van				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.68	0.60	0.069	0.167	19.64
ThermalGAN [18]:	0.77	0.78	0.030	0.104	23.76
InfraGAN w/o SSIM:	0.76	0.81	0.039	0.094	24.27
InfraGAN:	**0.82**	**0.83**	**0.024**	**0.081**	**25.21**

Table 4.15: Comparison of four different algorithms on the VEDAI dataset for 'Other' class. Each column shows five different metrics in the order of: SSIM, MSSIM, LPIPS, L1 and PSNR. Best result is shown in bold for each metric.

VEDAI dataset [5]	Other				
Algorithms	SSIM	MSSIM	LPIPS	L1	PSNR
Pix2Pix [19]:	0.73	0.65	0.075	0.266	17.05
ThermalGAN [18]:	0.84	0.85	0.015	0.070	27.88
InfraGAN w/o SSIM:	0.82	0.86	0.022	0.055	29.08
InfraGAN:	**0.88**	**0.90**	**0.011**	**0.052**	**29.85**

Table 4.16: Comparison of using L1 (InfraGANv2) and using Mean Squared Error, MSE, (InfraGANv1) for pixel by pixel loss on VEDAI and FLIR.

VEDAI [5]	SSIM	MSSIM	LPIPS	L1	PSNR
L1	0.79	0.84	0.030	0.065	27.64
MSE	**0.86**	**0.87**	**0.013**	**0.058**	**28.74**
FLIR	SSIM	MSSIM	LPIPS	L1	PSNR
L1	**0.25**	0.53	0.173	**0.171**	19.10
MSE	0.24	**0.54**	**0.164**	**0.171**	**19.23**

Chapter 5

Conclusions

In the thesis, we first introduce an object detection algorithm to detect objects in aerial images. Object detection is an important problem in computer vision. For instance, by detecting objects in UAV-based videos, we can help detecting any suspicious activity and traffic accidents in a more automated manner. This thesis examined and proposed a novel object detection architecture with enhanced loss terms and attention based modules for detecting objects in aerial images taken by UAVs. Our preliminary results confirm that SkyDataNet can outperform other baseline works detailed in the thesis in terms of average recall, despite the others showed similar, but slightly better, performance with respect to AP. In addition to these, when examining Figure 3.10 and Figure 3.11, despite there is no ground truth labels for the objects especially in upper region of the frame, SkyDataNet is able to detect some of the objects. When zooming in those relative regions, there is a confusion between person and other background objects. For future work, to fairly compare the methods, we can ignore those regions for both training and validation stages. As the use of IR sensors increases, the demand on using algorithms that can run on IR images also increases. The IR images complement RGB images from the thermal spectrum and help algorithms to detect many objects easier. However, the lack of big and public datasets containing IR images, limits what can be obtained from the existing deep object detection algorithms (such as SyNet [39] or YOLO [30]). In this work, we tackle that problem where algorithms

can learn the relation between the visible and IR spectrum so that we can artificially generate the IR equivalent of any given visible image. For that purpose, as being the first step, we introduce InfraGAN architecture and study the performance of different InfraGAN versions in this thesis. While the literature has some preliminary works reporting qualitative results (see for example [56]), our work is one of the early works that presents a comprehensive analysis using three different bench-marking data sets and five different metrics. Our experimental results (see Table 4.2, 4.3 and 4.4) show that our proposed algorithm yields the best overall performance on all five metrics for both VEDAI and KAIST datasets, when compared to the other baseline approaches. Additionally, Figure 4.7, Figure 4.8 and Figure 4.9 demonstrate that our generated IR images look similar to the ground truth IR images. That makes our approach of using GAN-based networks utilizing U-Nets with two level discriminator a good alternative solution to obtain the IR equivalent of a given visible image. Since we build additional layers as decoder on top of the existing encoder stage, our implementation takes slightly longer time during the training (see Table4.1) when compared to ThermalGAN and Pix2Pix architectures. Figure 4.10 compares U-Net architecture (as used in our implementation, the U-Net architecture is shown in Figure 4.4) to GAN-based networks. As shown in the figure, the U-Net architecture (which is trained by using both SSIM and L1 losses) yields more blurry outputs than GAN based networks, even if it yields slightly higher SSIM results in Table 4.2, 4.3 and 4.4. Even though more recent literature heavily considers unsupervised or semi-supervised solutions which focuses on a different set of problems than ours, unsupervised algorithms are not good to use in every domain as straightforward as utilizing the supervised algorithms. At this point, Figure 4.11 points out that the recent SOTA unsupervised GAN models does not satisfy domain transfer in our used datasets. In other words, when we use models and datasets at this scale, today's supervised domain transfer algorithms, such as InfraGAN, should be the first approach to be considered. Within the scope of this thesis, three journal articles have been published [83, 55, 77].

Bibliography

[1] D. Du, P. Zhu, L. Wen, X. Bian, H. Ling, Q. Hu, T. Peng, J. Zheng, X. Wang, Y. Zhang, L. Bo, H. Shi, R. Zhu, A. Kumar, A. Li, A. Zinollayev, A. Askergaliyev, A. Schumann, B. Mao, and Z. Liu, "Visdrone-det2019: The vision meets drone object detection in image challenge results," 10 2019.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[3] J. Kim, M. Kim, H. Kang, and K. H. Lee, "U-gat-it: Unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation," in *International Conference on Learning Representations*, 2020.

[4] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," in *International Conference on Learning Representations*, 2019.

[5] S. Razakarivony and F. Jurie, "Vehicle detection in aerial imagery : A small target detection benchmark," *Journal of Visual Communication and Image Representation*, vol. 34, 03 2015.

[6] S. Hwang, J. Park, N. Kim, Y. Choi, and I. S. Kweon, "Multispectral pedestrian detection: Benchmark dataset and baselines," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[7] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[8] R. Girshick, "Fast r-cnn," in *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, (USA), p. 1440–1448, IEEE Computer Society, 2015.

[9] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[10] R. Valiente, M. Zaman, Y. P. Fallah, and S. Ozer, *CONNECTED AND AUTONOMOUS VEHICLES IN THE DEEP LEARNING ERA: A CASE STUDY ON COMPUTER-GUIDED STEERING*, ch. CHAPTER 2.10, pp. 365–384. WORLD SCIENTIFIC, 2020.

[11] D. Gözen and S. Ozer, "Visual Object Tracking in Drone Images with Deep Reinforcement Learning," in *25th International Conference on Pattern Recognition (ICPR2020)*, 2020.

[12] N. Barzilay, T. B. Shalev, and R. Giryes, "MISS GAN: a Multi-Illustrator style generative adversarial network for image to illustration translation," *Pattern Recognition Letters*, vol. 151, pp. 140–147, 2021.

[13] L. Ran, Y. Hong, S. Zhang, Y. Yang, and Y. Zhang, "Improving visible-thermal reid with structural common space embedding and part models," *Pattern Recognition Letters*, vol. 142, pp. 25–31, 2021.

[14] J. Miura, M. Demura, K. Nishi, and S. Oishi, "Thermal comfort measurement using thermal-depth images for robotic monitoring," *Pattern Recognition Letters*, vol. 137, pp. 108–113, 2020.

[15] C. Bisogni, L. Cascone, J.-L. Dugelay, and C. Pero, "Adversarial attacks through architectures and spectra in face recognition," *Pattern Recognition Letters*, vol. 147, pp. 55–62, 2021.

[16] P. Wang, F. Su, Z. Zhao, Y. Zhao, L. Yang, and Y. Li, "Deep hard modality alignment for visible thermal person re-identification," *Pattern Recognition Letters*, vol. 133, pp. 195–201, 2020.

[17] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[18] V. V. Kniaz, V. A. Knyaz, J. Hladuvka, W. G. Kropatsch, and V. Mizginov, "Thermalgan: Multimodal color-to-thermal image translation for person re-identification in multispectral dataset," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[19] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CVPR*, 2017.

[20] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, pp. 1097–1105, Curran Associates, Inc., 2012.

[21] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[23] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[24] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger, "Convolutional networks with dense connectivity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[25] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[26] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the 36th International Conference on Machine Learning* (K. Chaudhuri and R. Salakhutdinov, eds.), vol. 97 of *Proceedings of Machine Learning Research*, pp. 6105–6114, PMLR, 09–15 Jun 2019.

[27] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei, "Deformable convolutional networks," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 764–773, 2017.

[28] X. Zhu, H. Hu, S. Lin, and J. Dai, "Deformable convnets v2: More deformable, better results," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9308–9316, 2019.

[29] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, pp. 21–37, Springer, 2016.

[30] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[31] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[32] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.

[33] O. Sahin and S. Ozer, "Yolodrone: Improved yolo architecture for object detection in drone images," in *2021 44th International Conference on Telecommunications and Signal Processing (TSP)*, pp. 361–365, 2021.

[34] O. Sahin and S. Ozer, "Yolodrone+: Improved yolo architecture for object detection in uav images," in *2022 30th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, 2022.

[35] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "Yolox: Exceeding yolo series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[36] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

[37] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 734–750, 2018.

[38] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," in *arXiv preprint arXiv:1904.07850*, 2019.

[39] B. M. Albaba and S. Ozer, "Synet: An ensemble network for object detection in uav images," in *25th International Conference on Pattern Recognition (ICPR2020)*, 2020.

[40] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2980–2988, 2017.

[41] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems* (I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.

[42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations*, 2021.

[43] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, pp. 213–229, Springer, 2020.

[44] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable detr: Deformable transformers for end-to-end object detection," *arXiv preprint arXiv:2010.04159*, 2020.

[45] Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, and B. Guo, "Swin transformer v2: Scaling up capacity and resolution," in *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[46] F. Li, H. Zhang, S. Liu, J. Guo, L. M. Ni, and L. Zhang, "Dn-detr: Accelerate detr training by introducing query denoising," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 13619–13627, 2022.

[47] H. Zhang, F. Li, S. Liu, L. Zhang, H. Su, J. Zhu, L. M. Ni, and H.-Y. Shum, "Dino: Detr with improved denoising anchor boxes for end-to-end object detection," 2022.

[48] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[49] M. Xu, Z. Zhang, H. Hu, J. Wang, L. Wang, F. Wei, X. Bai, and Z. Liu, "End-to-end semi-supervised object detection with soft teacher," *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[50] J. Zhang, X. Lin, W. Zhang, K. Wang, X. Tan, J. Han, E. Ding, J. Wang, and G. Li, "Semi-detr: Semi-supervised object detection with detection transformers," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 23809–23818, June 2023.

[51] Y. Jiang, Y. Liu, Q. Peng, F. Jie, and D. Ming, "Infrared image generation method based on visible light remote sensing image," in *2019 IEEE 3rd Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 63–68, 2019.

[52] L. Zhang, A. Gonzalez-Garcia, J. van de Weijer, M. Danelljan, and F. S. Khan, "Synthetic data generation for end-to-end thermal infrared tracking," *IEEE Transactions on Image Processing*, vol. 28, no. 4, pp. 1837–1850, 2018.

[53] L. Li, P. Li, M. Yang, and S. Gao, "Multi-branch semantic gan for infrared image generation from optical image," in *Intelligence Science and Big Data Engineering. Visual Data Engineering* (Z. Cui, J. Pan, S. Zhang, L. Xiao, and J. Yang, eds.), (Cham), pp. 484–494, Springer International Publishing, 2019.

[54] H. Xu, P. Liang, W. Yu, J. Jiang, and J. Ma, "Learning a generative model for fusing infrared and visible images via conditional generative adversarial network with dual discriminators.," in *IJCAI*, pp. 3954–3960, 2019.

[55] S. Özer, M. Ege, and M. A. Özkanoglu, "Siamesefuse: A computationally efficient and a not-so-deep network to fuse visible and infrared images," *Pattern Recognition*, vol. 129, p. 108712, 2022.

[56] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[57] A. Newell, K. Yang, and J. Deng, "Stacked hourglass networks for human pose estimation," in *Computer Vision – ECCV 2016* (B. Leibe, J. Matas, N. Sebe, and M. Welling, eds.), (Cham), pp. 483–499, Springer International Publishing, 2016.

[58] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015* (N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, eds.), (Cham), pp. 234–241, Springer International Publishing, 2015.

[59] X. Dai, Y. Chen, B. Xiao, D. Chen, M. Liu, L. Yuan, and L. Zhang, "Dynamic head: Unifying object detection heads with attentions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7373–7382, June 2021.

[60] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, (Cambridge, MA, USA), p. 3123–3131, MIT Press, 2015.

[61] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic relu," in *16th European Conference Computer Vision (ECCV 2020)*, August 2020.

[62] X. Jin and J. Han, *K-Means Clustering*, pp. 563–564. Boston, MA: Springer US, 2010.

[63] S. Ozer, "Similarity domains machine for scale-invariant and sparse shape modeling," *IEEE Transactions on Image Processing*, vol. 28, no. 2, pp. 534–545, 2019.

[64] Z. Dong, G. Li, Y. Liao, F. Wang, P. Ren, and C. Qian, "Centripetalnet: Pursuing high-quality keypoint pairs for object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10519–10528, 2020.

[65] R. Padilla, S. L. Netto, and E. A. B. da Silva, "A survey on performance metrics for object-detection algorithms," in *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pp. 237–242, 2020.

[66] B. Carterette, *Precision and Recall*, pp. 2126–2127. Boston, MA: Springer US, 2009.

[67] K. Boyd, K. H. Eng, and C. D. Page, "Area under the precision-recall curve: Point estimates and confidence intervals," in *Machine Learning and Knowledge Discovery in Databases* (H. Blockeel, K. Kersting, S. Nijssen, and F. Železný, eds.), (Berlin, Heidelberg), pp. 451–466, Springer Berlin Heidelberg, 2013.

[68] E. Zhang and Y. Zhang, *Average Precision*, pp. 192–193. Boston, MA: Springer US, 2009.

[69] J. Hosang, R. Benenson, P. Dollár, and B. Schiele, "What makes for effective detection proposals?," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 814–830, 2016.

[70] P. Zhu, L. Wen, X. Bian, H. Ling, and Q. Hu, "Vision meets drones: A challenge," *arXiv preprint arXiv:1804.07437*, 2018.

[71] H. Su, S. Wei, M. Yan, C. Wang, J. Shi, and X. Zhang, "Object detection and instance segmentation in remote sensing imagery based on precise mask r-cnn," in *IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 1454–1457, IEEE, 2019.

[72] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang, "Dota: A large-scale dataset for object detection in aerial images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[73] H. Su, S. Wei, M. Yan, C. Wang, J. Shi, and X. Zhang, "Object detection and instance segmentation in remote sensing imagery based on precise mask r-cnn," in *IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium*, pp. 1454–1457, 2019.

[74] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[75] M. Contributors, "MMCV: OpenMMLab computer vision foundation." `https://github.com/open-mmlab/mmcv`, 2018.

[76] K. Chen, J. Wang, J. Pang, Y. Cao, Y. Xiong, X. Li, S. Sun, W. Feng, Z. Liu, J. Xu, Z. Zhang, D. Cheng, C. Zhu, T. Cheng, Q. Zhao, B. Li, X. Lu, R. Zhu, Y. Wu, J. Dai, J. Wang, J. Shi, W. Ouyang, C. C. Loy, and D. Lin, "MMDetection: Open mmlab detection toolbox and benchmark," *arXiv preprint arXiv:1906.07155*, 2019.

[77] S. Ozer, H. E. Ilhan, M. A. Ozkanoglu, and H. A. Cirpan, "Offloading deep learning powered vision tasks from uav to 5g edge server with denoising," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 6, pp. 8035–8048, 2023.

[78] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[79] E. Schonfeld, B. Schiele, and A. Khoreva, "A u-net based discriminator for generative adversarial networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8207–8216, 2020.

[80] Zhou Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," *IEEE Transactions on Image Processing*, vol. 13, no. 4, pp. 600–612, 2004.

[81] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang, "The unreasonable effectiveness of deep features as a perceptual metric," in *CVPR*, 2018.

[82] A. Horé and D. Ziou, "Image quality metrics: PSNR vs. SSIM," in *2010 20th International Conference on Pattern Recognition*, pp. 2366–2369, 2010.

[83] M. A. Özkanoğlu and S. Ozer, "Infragan: A gan architecture to transfer visible images to infrared domain," *Pattern Recognition Letters*, vol. 155, pp. 69–76, 2022.

[84] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[85] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International Conference on Computer Vision*, pp. 2564–2571, 2011.

Appendix A

Additional Tasks: Image Registration

Object detection using thermal and RGB based images may require a proper image registration as a preprocessing step. As a part of this thesis, we also tacked with image registration process between the thermal and RGB images. To register thermal and RGB video frames, we used SkyData where the thermal and RGB images are obtained by using Zenmuse H20T camera. We followed three different methods on the dataset and those methods are described below:

A.1 Method I

We calculated camera intrinsic parameters and transformation matrices between both frames by using conventional chessboard methods. We insolated the chessboard in order for the thermal camera to perceive edges and corners of chessboard, since, thermal camera may not be able to distinguish black and white squares under the cloudy weather in contrast to RGB standard camera. After getting transformation matrix, all annotated labels in RGB are projected into their thermal counterpart. As seen in Figure A.1 and Figure A.2, edges of chessboard are

Figure A.1: The chessboard in RGB domain captured by Zenmuse H20-T. The colored points are matching points captured by conventional edge detection algorithm.

found by using conventional edge detection method where the sizes of chessboard are 125mm squares - 5x3 vertices, 6x4 squares in A1 format.. Although both of images are obviously overlapped for bigger objects, the annotations that are mapped into another images represented the objects poorly. Upon detailed examination, we found that smaller objects were more affected and the labels did not match. Especially, focusing on the moving ones, there were more inconsistency between both images. In addtition to these, it is observed that thermal camera of Zenmuse H20-T has more distortion effect as seen in Figure A.2. Because, when you look at the corners and sides of the chessboard in thermal image, it is seen that there are more bends in the chessboard compared to RGB frame in Figure A.1. As a consequence, this degrades the quality of annotations of small objects that are mapped from RGB to thermal domain. Figure A.3 proves that both images are nearly overlapped when the frames are not taken from a distant angle.

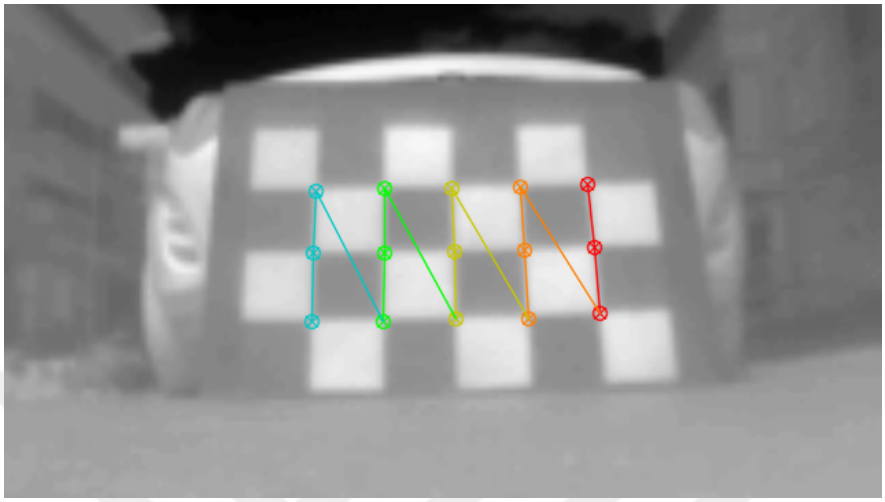


Figure A.2: The chessboard in infrared domain captured by Zenmuse H20-T. The colored points are matching points captured by conventional edge detection algorithm.

A.2 Method II

We manually selected numerous matching points by zooming in both images like corners of stones, traffic lines and any other points that are visible in both domain. Then, by using pixel locations of these points, we calculated homoghraphy matrix. We did not manage to fully overlap both images because of several problems such as camera distortion and human effect. We observed that each cameras have different distortion ratio and does not reflect exactly same points in the objects. Because of that, it is impossible to select same points sufficiently on both images.

A.3 Method III

We directly used conventional computer vision algorithms to match vertices and edges in both images such as SIFT [84], ORB[85]. However, because of domain difference, they are not convenient to find similar points on both images. They are justified and used in similar images on same domain in contrast to our case. Besides, they are preferred in which angle of the cameras are changed and when the cameras have different attributes such as different lens.

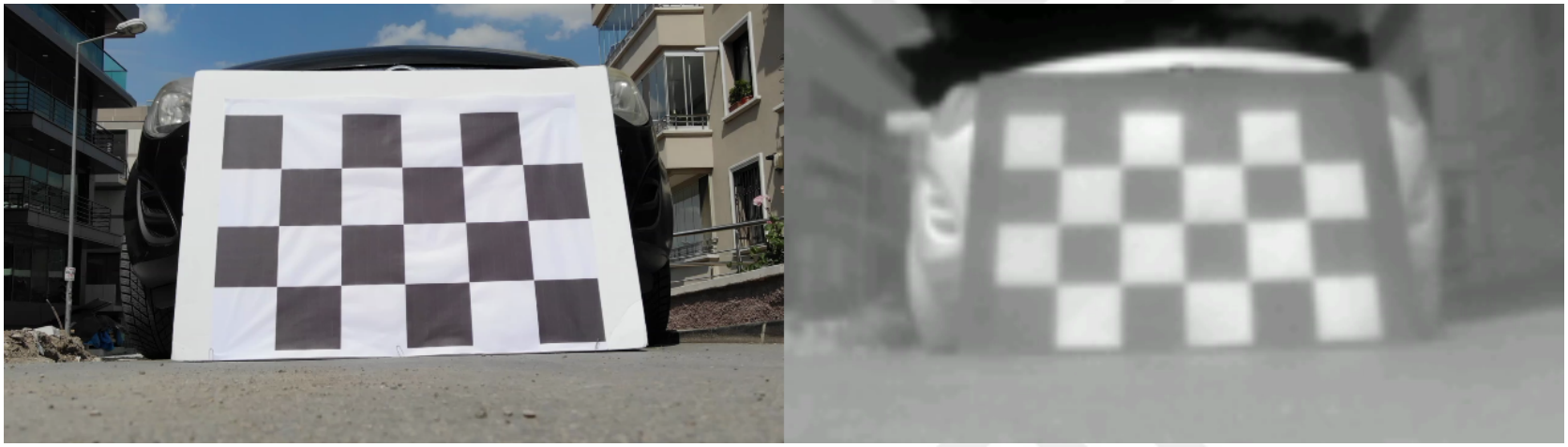


Figure A.3: The registered version of both images. The left RGB image is mapped into to original Infrared image.