Tiny Object Detection in Remote Sensing Images: End-to-End Super-Resolution and Object Detection with Deep Learning

by

Jakaria Rabbi

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

In this thesis, we study the problem of detecting small objects on low-resolution (LR) satellite imagery. Small-object detection is a challenging problem, especially from LR images. To tackle the challenge, we propose a method to generate super-resolution images from low-resolution images and simultaneously detect objects from the super-resolution images. A generative adversarial network (GAN)-based model called enhanced super-resolution GAN (ESRGAN) shows remarkable image enhancement performance, but reconstructed images miss high-frequency edge information. Therefore, object detection performance degrades for the small objects on recovered noisy and low-resolution remote sensing images. Inspired by the success of edge enhanced GAN (EEGAN) and ESRGAN, we apply a new edge-enhanced super-resolution GAN (EESRGAN) to improve the image quality of remote sensing images and used different detector networks in an end-to-end manner where detector loss is backpropagated into the EESRGAN to improve the detection performance. We propose an architecture with three components: ESRGAN, Edge Enhancement Network (EEN), and Detection network. We use residual-in-residual dense blocks (RRDB) for both the GAN and EEN, and for the detector network, we use the faster region-based convolutional network (FRCNN) (two-stage detector) and single-shot multi-box detector (SSD) (one stage detector). Extensive experiments on a public (car overhead with context) dataset and another self-assembled (oil and gas storage tank) satellite dataset show superior performance of our method compared to the standalone state-of-the-art object detectors. While working with the detection problem, we create a GUI tool to label, train, and detect objects from remote sensing images that cover a large area. This GUI makes it easier to create small image tiles from the large satellite images, training the state-of-the-art object detection models, running the detection, and finally obtaining the output geolocation for the detected objects.

Preface

The work presented in this thesis is advised by Professor Nilanjan Ray and Professor Matthias Schubert. Their suggestions were indispensable in formulating the methods and experimental setups. The author was the main contributor who implemented the methods and performed experiments.

Chapter 4 of this thesis is the basis of a journal paper published as: Rabbi, J.; Ray, N.; Schubert, M.; Chowdhury, S.; Chao, D. *Small-Object Detection in Remote Sensing Images with End-to-End Edge-Enhanced GAN and Object Detector Network.* Remote Sens. 2020, 12, 1432.

"If I have seen further it is by standing on the shoulders of Giants."

– Isaac Newton

Acknowledgements

I am very grateful to Professor Nilanjan Ray and Professor Matthias Schubert for their excellent supervision and guidance throughout the thesis. The work could not be done without their insightful guidance. They taught me how to find a problem and do research. I would also like to thank Professor Hong Zhang and all members of the Robotics-vision group for the suggestions and discussions in group meetings during my master's program.

Finally, I would like to thank my parents and wife for their love and support. Their love and support help me to accomplish this task.

Contents

List of Tables

List of Figures

List of Acronyms

The following acronyms are used in this thesis:

SRCNN	Single image Super-Resolution Convolutional Neural Network
VDSR	Very Deep Convolutional Networks
GAN	Generative Adversarial Network
SRGAN	Super-Resolution Generative Adversarial Network
ESRGAN	Enhanced Super-Resolution Generative Adversarial Network
EEGAN	Edge-Enhanced Generative Adversarial Network
EESRGAN	Edge-Enhanced Super-Resolution Generative Adversarial Network
RRDB	Residual-in-Residual Dense Blocks
EEN	Edge-Enhancement Network
SSD	Single-Shot MultiBox Detector
YOLO	You Only Look Once
CNN	Convolutional Neural Network
R-CNN	Region-based Convolutional Neural Network
FRCNN	Faster Region-based Convolutional Neural Network
VGG	Visual Geometry Group
BN	Batch Normalization
MSCOCO	Microsoft Common Objects in Context
OGST	Oil and Gas Storage Tank
COWC	Car Overhead With Context
GSD	Ground Sampling Distance
G	Generator
D	Discriminator
ISR	Intermediate Super-Resolution
SR	Super-Resolution
HR	High-Resolution
LR	Low-Resoluton
GT	Ground Truth
FPN	Feature Pyramid Network
RPN	Region Proposal Network
AER	Alberta Energy Regulator
AGS	Alberta Geological Survey
AP	Average Precision
IoU	Intersection over Union

TP	True Positive
FP	False Positive
FN	False Negative

Chapter 1

Introduction

1.1 Introduction

Object detection on remote sensing imagery has numerous prospects in various fields, such as environmental regulation, surveillance, military [11], [80], national security, traffic, forestry [16], oil and gas activity monitoring. There are many methods for detecting and locating objects from images, which are captured using satellites or drones. However, detection performance is not satisfactory for noisy and low-resolution (LR) images, especially when the objects are small [54]. Even on high-resolution (HR) images, the detection performance for small objects is lower than that for large objects [64].

Current state-of-the-art detectors have excellent accuracy on benchmark datasets, such as ImageNet [62] and Microsoft common objects in context (MSCOCO) [46]. These datasets consist of everyday natural images with distinguishable features and comparatively large objects.

On the other hand, there are various objects in satellite images like vehicles, small houses, small oil and gas storage tanks etc., only covering a small area [54]. The state-of-the-art detectors [45], [49], [59], [60] show a significant performance gap between LR images and their HR counterparts due to a lack of input features for small objects [25]. In addition to the general object detectors, researchers have proposed specialized methods, algorithms, and network architectures to detect particular types of objects from satellite images such as vehicles [70], [77], buildings [66], and storage tanks [53]. These methods are object-specific and use fixed resolution for feature extraction and detection.

To improve detection accuracy on remote sensing images, researchers have used deep convolutional neural network (CNN)-based super-resolution (SR) techniques to generate artificial images and then detect objects [25], [64]. Deep CNN-based SR techniques such as single image super-resolution convolutional networks (SRCNN) [13] and accurate image super-resolution using very deep convolutional networks (VDSR) [30] showed excellent results on generating realistic HR imagery from LR input data. Generative Adversarial Network (GAN)-based [19] methods such as super-resolution GAN (SRGAN) [36] and enhanced super-resolution GAN (ESRGAN) [73] showed remarkable performance in enhancing LR images with and without noise. These models have two subnetworks: a generator and a discriminator as depicted in the figure 1.1. Both subnetworks consist of deep CNNs. Datasets containing HR and LR image pairs are used for training and testing the models. The generator generates HR images from LR input images, and the discriminator predicts whether generated image is a real HR image or an upscaled LR image. After sufficient training, the generator generates HR images that are similar to the ground truth HR images, and the discriminator cannot correctly discriminate between real and fake images anymore.

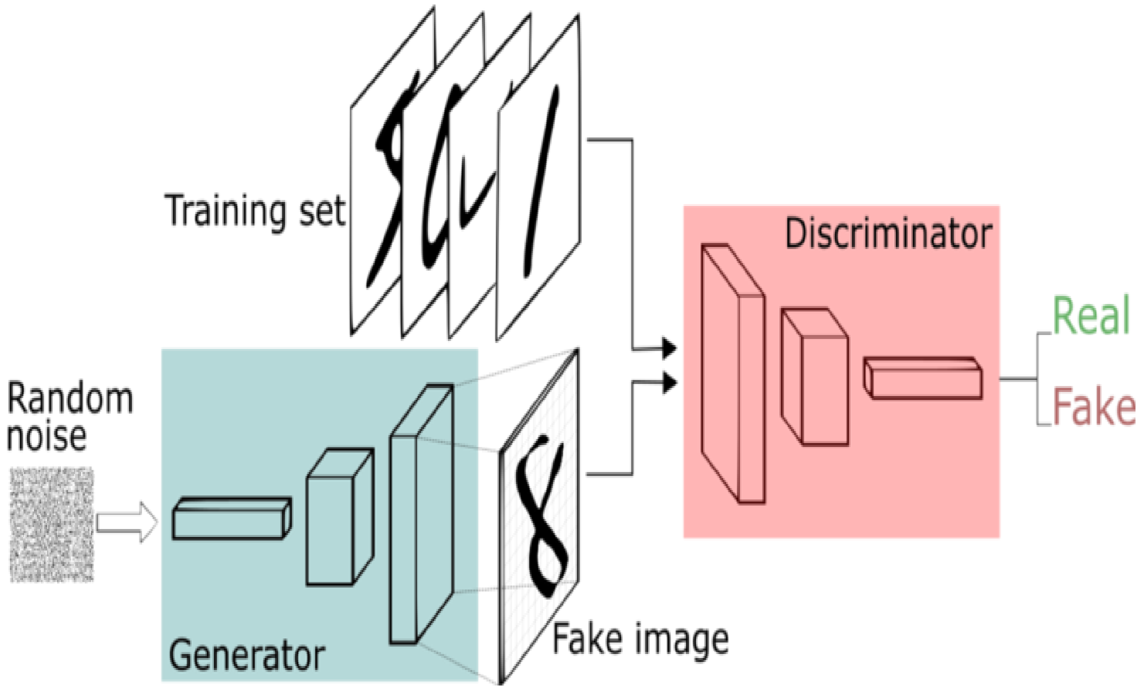


Figure 1.1: A diagram of GAN. It has a generator and a discriminator. The generator generates a fake image from random noise and the discriminator tries to differentiate between the fake image from the generator and the real image from the training set [19].

Although the resulting images look realistic, the compensated high-frequency details such as image edges may cause inconsistency with the HR ground truth images [27]. Some works showed that this issue negatively impacts land cover classification results [26], [71]. Edge information is an important feature for object detection [79], and therefore, this information needs to be preserved in the enhanced images for acceptable detection accuracy.

In order to obtain clear and distinguishable edge information, researchers proposed several methods using separate deep CNN edge extractors [51], [75]. The results of these methods are sufficient for natural images, but the performance degrades on LR and noisy remote sensing images [27]. A recent method [27] used the GAN-based edge-enhancement network (EEGAN) to generate a visually pleasing result with sufficient edge information. EEGAN employs two subnetworks for the generator. One network generates intermediate HR images, and the other network generates sharp and noise-free edges from the intermediate images. The method uses a Laplacian operator [29] to extract edge information and in addition, it uses a mask branch to obtain noise-free edges. This approach preserves sufficient edge information, but sometimes the final output images are blurry compared to a current state-of-the-art GAN-based SR method [73] due to the noises introduced in the enhanced edges that might hurt object detection performance.

Another important issue with small-object detection is the huge cost of HR imagery for large areas. Many organizations are using very high-resolution satellite imagery to fulfill their purposes. When it comes to continuous monitoring of a large area for regulation or traffic purposes, it is costly to buy HR imagery frequently. Publicly available satellite imagery such as Landsat-8 [35] (30 m/pixel) and Sentinel-2 [63] (10 m/pixel) are not suitable for detecting small objects due to the high ground sampling distance (GSD). Detection of small objects (e.g., oil and gas storage tanks and buildings) is possible from commercial satellite imagery such as 1.5-m GSD SPOT-6 imagery but the detection accuracy is low compared to HR imagery, e.g., 30-cm GSD DigitalGlobe imagery in Bing map.

We also need a proper labeling tool to annotate satellite imagery. There are are many labeling tools to label satellite imagery. ArcGIS [4], labelbox [33], and ENVI [14] software have some tools to annotate and detect objects from satellite imagery, and also, it can train and test models using the annotated data. They are resource extensive or very costly. There are also open-source labeling tools such as Labelimg [72] and computer vision annotation tool (CVAT) [12]. But they lack the facility to load large satellite imagery, create small tiles for annotation and train/test options. Therefore, a light-weight annotation tool with the train/test facility for large satellite imagery can be a good addition to the open-source research community.

1.2 Thesis Statement

We have identified two main problems to detect small-objects from satellite imagery. First, the accuracy of small-object detection is lower compared to large objects, even in HR imagery due to sensor noise, atmospheric effects, and geometric distortion. Secondly, we need to have access to HR imagery, which is very costly for a vast region with frequent updates. Therefore, we need a solution to increase the accuracy of the detection of smaller objects from LR imagery, and the solution would resolve both the problems that we have identified. To the best of our knowledge, no work employed both SR network with edge-enhancement and object detector network in an end-to-end manner, i.e, using joint optimization to detect small remote sensing objects.

In this thesis, we propose an end-to-end architecture where object detection and super-resolution is performed simultaneously. Figure 1.2 shows the significance of our method. State-of-the-art detectors miss objects when trained on the LR images; in comparison, our method can detect those objects. The detection performance improves when we use SR images for the detection of objects from two different datasets. Average precision (AP) versus different intersection over union (IoU) values (for both LR and SR) are plotted to visualize overall performance on test datasets. From figure 1.2, we observe that for both the datasets, our proposed end-to-end method yields significantly better IoU values for the same AP. Our proposed architecture consists of two parts: EESRGAN network and a detector network. Our approach is inspired by EEGAN and ESRGAN networks and showed a remarkable improvement over EEGAN to generate visually pleasing SR satellite images with enough edge information.

We employed a generator subnetwork, a discriminator subnetwork, and an edge-enhancement subnetwork [27] for the SR network. For the generator and edge-enhancement network, we used residual-in-residual dense blocks (RRDB) [73]. These blocks contain multi-level residual networks with dense connections that showed good performance on image enhancement.

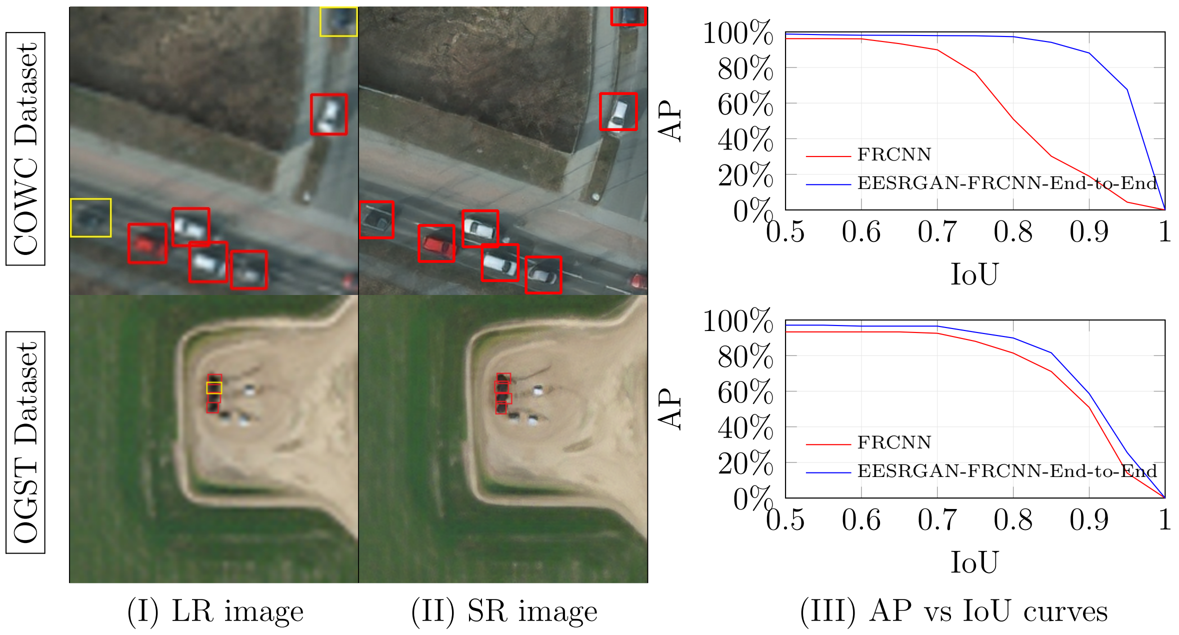


Figure 1.2: Detection on LR (low-resolution) images (60cm/pixel) is shown in (I); in (II), we show the detection on generated SR (super-resolution) images (15 cm/pixel). The first row of this figure represents the COWC (car overhead with context) dataset [52], and the second row represents the OGST (oil and gas storage tank) dataset [57]. AP (average precision) values versus different IoU (intersection over union) values for the LR test set and generated SR images from the LR images are shown in (III) for both the datasets. We use FRCNN (faster region-based CNN) detector on LR images for detection. Then instead of using LR images directly, we use our proposed end-to-end EESRGAN (edge-enhanced SRGAN) and FRCNN architecture (EESRGAN-FRCNN) to generate SR images and simultaneously detect objects from the SR images. The red bounding boxes represent true positives, and yellow bounding boxes represent false negatives. IoU=0.75 is used for detection.

We used a relativistic discriminator [28] instead of a normal discriminator.

Besides GAN loss and discriminator loss, we employed Charbonnier loss [8] for the edge-enhancement network. Finally, we used different detectors [49], [60] to detect small objects from the SR images. The detectors acted like the discriminator as we backpropagated the detection loss into the SR network and, therefore, it improved the quality of the SR images.

We have created a software to label, train, and detect objects from remote sensing images that cover a large area. The interface of our software makes it easier to create small image tiles from the large satellite images, training the state-of-the-art object detection models, running the detection, and finally obtaining the output geolocation for the detected objects.

We created the oil and gas storage tank (OGST) dataset [57] from satellite imagery (Bing map), which has 30 cm and 1.2 m GSD. The dataset contains labeled oil and gas storage tanks from the Canadian province of Alberta, and we detected the tanks on SR images. Detection and counting of the tanks are essential for the Alberta Energy Regulator (AER) [1] to ensure safe, efficient, orderly, and environmentally responsible development of energy resources. Therefore, there is a potential use of our method for detecting small objects from LR satellite imagery. The OGST dataset is available on Mendeley [57].

In addition to the OGST dataset, we applied our method on the publicly available car overhead with context (COWC) [52] dataset to compare the performance of detection for varying use-cases. During training, we used HR and LR image pairs but only required LR images for testing. Our method outperformed standalone state-of-the-art detectors for both datasets.

1.3 Thesis Contribution

Our work in this thesis has the following contributions.

- We show a super-resolution network with object detector networks improve object detection performance on low-resolution images compare to standalone object detectors, and performance is further improved by end-to-end training. We also create an oil and gas tank storage dataset to show the improvement.
- We provide empirical evidence that edge-enhancement helps to get improvement for object detection results.
- We create a tool to create small tiles of images from large satellite imagery, create labels, and train/test the images with different object detection models and also generate detection results with geolocation.

We make our tool and the end-to-end network available to the community through GitHub.

1.4 Organization of the Thesis

This thesis is organized as follows.

- **Chapter 1. Introduction**

  The problem related to object detection on low-resolution remote sensing images and related solutions are discussed in this chapter. We also discuss our labeling and detection tool here. We provide the thesis statement and contribution in solving the problem related to the detection of objects from satellite imagery.

- **Chapter 2. Related Works**

  We reviewed works related to our research on object detection, image super-resolution, simultaneous object detection with super-resolution, and similar labeling tools for object detection.

- **Chapter 3. A Tool for Labeling and Detection**

  We give a short description of our GUI tool and include the steps to create labels, and train/test the images with different object detection models.

- **Chapter 4. An End-to-End Deep Learning Architecture for Small-Object Detection in Remote Sensing Images**

  We present our end-to-end small-object detection framework in this chapter. We depict our method, internal network components, and derive our final loss functions here. We provide the details of the training procedure, dataset description, experimental results, and show the effectiveness of our end-to-end method.

- **Chapter 5. Conclusion and Future Work**

  We conclude our thesis in this section. We discuss the results of our method and suggest some ideas to improve our method as future work.

Chapter 2

Related Works

Our work consists of an end-to-end edge enhanced image super-resolution network with an object detector network and an image annotation tool with GUI. In this section, we discuss some research related to our method.

2.1 Image Super-Resolution

Many methods were proposed on SR using deep CNNs. Dong et al. proposed super-resolution CNN (SRCNN) [13] to enhance low-resolution image in an end-to-end training outperforming previous super-resolution techniques. The deep CNNs for super-resolution evolved rapidly later on, and researchers introduced residual blocks [36], densely connected networks [67], and residual dense block [82] for improving super-resolution results. Ledig et al. introduced a framework called SRGAN [36] that was capable of generating photo-realistic natural images for 4× upscaling factors. Authors proposed a perceptual loss function, which was a combination of an adversarial loss and a content loss. The adversarial loss helped a generator to generate a SR image similar to the original photo-realistic image and helped a discriminator network to differentiate between the SR images and original photo-realistic images. The authors also used a content loss of perceptual similarity. Figure 2.1 shows the architecture of the SRGAN. It contains a generator and a discriminator network, and the network diagrams of them are also shown.

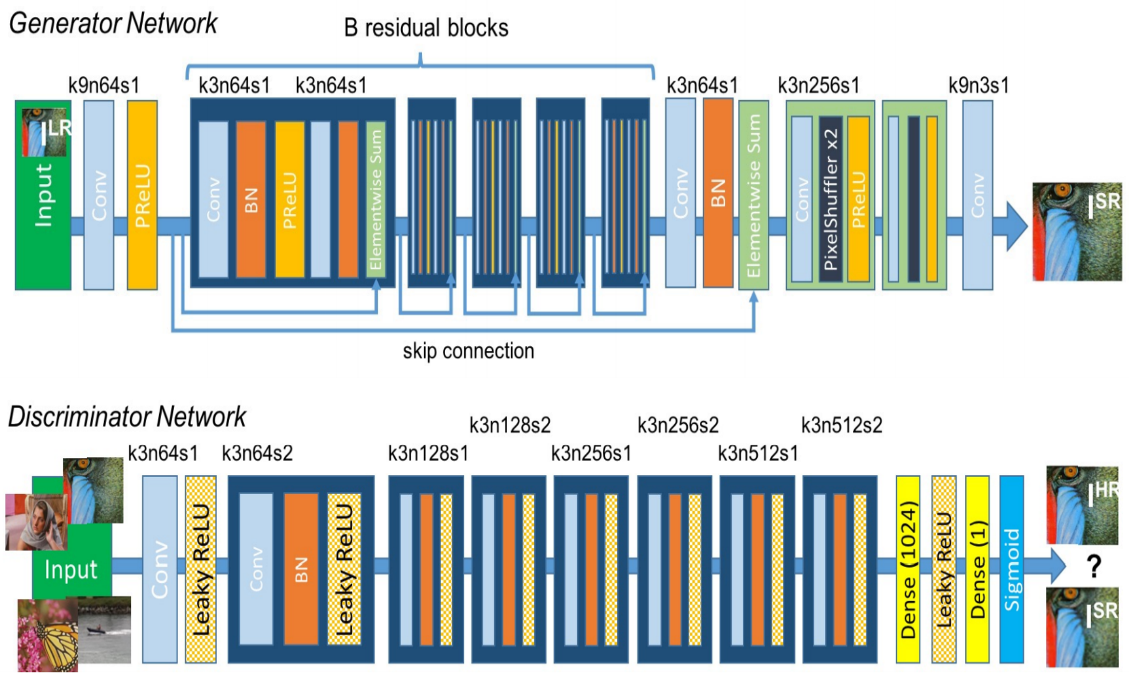


Figure 2.1: Architecture of SRGAN: network diagram of generator and discriminator with corresponding kernel size (k), number of feature maps (n) and stride (s) shown for each convolutional layer [36].

The generator generates fake images, and the discriminator tries to differentiate between the real and fake images. The generator has residual blocks [82] with skip connection and parametric ReLU [21] as the activation function. Residual blocks have 3×3 kernels [32], 64 feature maps, and stride of 1 for the convolutional layers. The generator also has upsampling layers to upsample the low-resolution images. The discriminator has convolutional blocks with leaky ReLu [74] and a fully connected layer with a final sigmoid activation. In the discriminator, 3×3 kernels are used with different numbers of feature maps and strides. He et al. [22] and Lim et al. [43] used deep CNNs without the batch normalization (BN) layer and observed significant performance improvement and stable training with a deeper network. These works were done on everyday natural images.

Liebel et al. [42] proposed deep CNN-based super-resolution network for multi-spectral remote sensing imagery. Authors got superior results compared to the conventional interpolation methods. A method was proposed by Yuan et al. [78] that can create high-resolution hyperspectral images from low-resolution images through transfer learning. They used natural images to train their model and used transfer learning to infer results for hyperspectral images. A method based on the transferred generative adversarial network for satellite image super-resolution was proposed in [50]. Authors removed the batch normalization layers from the generator to reduce the memory consumption and the computational time. The model used transfer learning for the insufficiency of training data in many application related to remote sensing imagery.

Jiang et al. [27] proposed a new super-resolution architecture for satellite imagery that was based on GAN. They introduced an edge-enhanced network to acquire smooth edge details in the final SR image and used Laplacian operator [29] to extract edge information from satellite images. Liu et al. [47] proposed a multi-task deep neural network for remote sensing image super-resolution and colorization simultaneously. Authors incorporated natural images with satellite imagery for colorization. They proposed a multi-scale deep encoder-decoder symmetrical network for the image super-resolution. Another super-resolution architecture named local-global combined networks based on deep CNN's was proposed by Lei et al. [37]. The architecture learned multilevel representations of satellite imagery, including both local details and global environmental priors.

2.2 Object Detection

Deep learning-based object detectors can be categorized into two subgroups, region-based CNN (R-CNN) models that employ two-stage detection and uniform models using single stage detection [69]. Two-stage detectors comprise of R-CNN [18], Fast R-CNN [17], Faster R-CNN [60] and the most used single stage detectors are SSD [49], You only look once (YOLO) [59] and RetinaNet [45]. In the first stage of a two-stage detector, regions of interest are determined by selective search or a region proposal network. Then, in the second stage, the selected regions are checked for particular types of objects and minimal bounding boxes for the detected objects are predicted. In figure 2.2, a high level architecture of faster R-CNN [60] is shown with two stages.

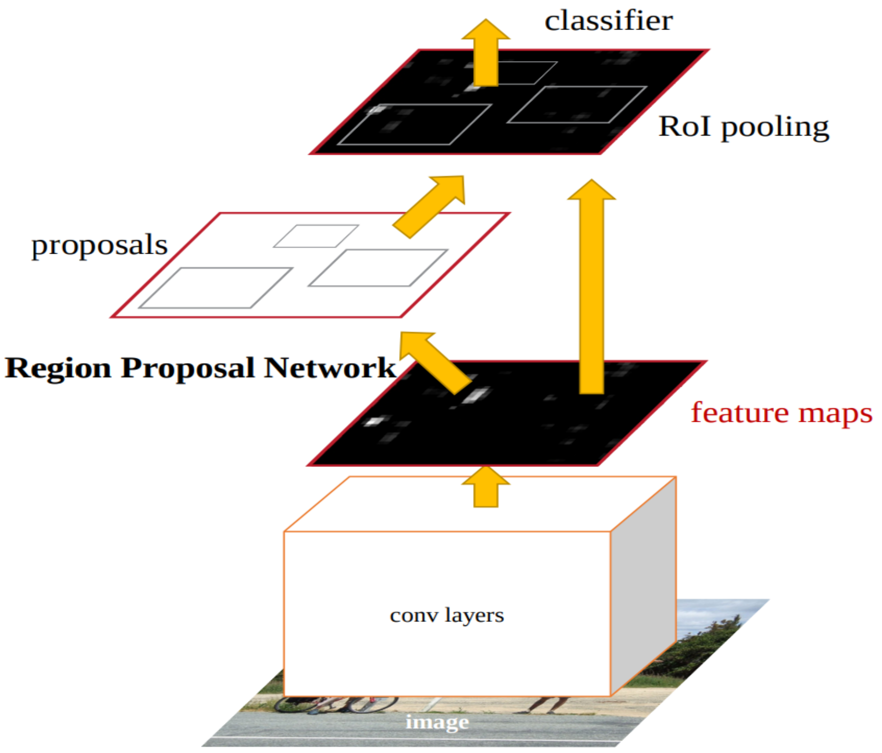


Figure 2.2: Faster R-CNN: A two-stage detector with region proposal and classification network [60].

In contrast, single-stage detectors omit the region proposal network and run detection on a dense sampling of all possible locations. Therefore, single-stage detectors are faster but, usually less accurate. RetinaNet [45] uses a focal loss function to deal with the data imbalance problem caused by many background objects and often showed similar performance as the two-stage approaches.

Many deep CNN-based object detectors were proposed on remote sensing imagery to detect and count small objects, such as vehicles [3], [39], [70]. Tayara et al. [70] introduced a convolutional regression neural network to detect vehicles from satellite imagery. Furthermore, a deep CNN-based detector was proposed [39] to detect multi oriented vehicles from remote sensing imagery. A method combining a deep CNN for feature extraction and a support vector machine (SVM) for object classification was proposed [3]. Ren et al. [61] modified the faster R-CNN detector to detect small objects in remote sensing images. They changed the region proposal network and incorporated context information into the detector. Another modified faster R-CNN detector was proposed by Tang et al. [68]. They used a hyper region proposal network to improve recall and used a cascade boosted classifier to verify candidate regions. This classifier can reduce false detection by mining hard negative examples.

An SSD-based end-to-end airplane detector with transfer learning was proposed, where, the authors used a limited number of airplane images for training [9]. They also proposed a method to solve the input size restrictions by dividing a large image into smaller tiles. Then they detected objects on smaller tiles and finally, mapped each image tile to the original image. They showed that their method performed better than the SSD model. In [58], the authors showed that finding a suitable parameter setting helped to boost the object detection performance of convolutional neural networks on remote sensing imagery. They used YOLO [59] as object detector to optimize the parameters and infer the results.

In [16], the authors detected conifer seedlings along recovering seismic lines from drone imagery. They used a dataset from different seasons and used faster R-CNN to infer the detection accuracy. There is another work [40] related to plant detection, where authors detected palm trees from satellite imagery using sliding window techniques and an optimized convolutional neural network.

Some works produced excellent results in detecting small objects. Lin et al. [44] proposed feature pyramid networks, which is a top-down architecture with lateral connections. The architecture could build high-level semantic feature maps at all scales. These feature maps boosted the object detection performance, especially for small object detection, when used as a feature extractor for faster R-CNN. Inspired by the receptive fields in human visual systems, Liu et al. [48] proposed a receptive field block (RFB) module that used the relationship between the size and eccentricity of receptive fields to enhance the feature discrimination and robustness. Hence, the module increased the detection performance of objects with various sizes when used as the replacement of the top convolutional layers of SSD.

A one-stage detector called single-shot refinement neural network (RefineDet) [81] was proposed to increase the detection accuracy and also enhance the inference speed. The detector worked well for small object detection. RefineDet used two modules in its architecture: an anchor refinement module to remove negative anchors and an object detection module that took refined anchors as the input. The refinement helped to detect small objects more efficiently than previous methods. In [41], feature fusion SSD (FSSD) was proposed where features from different layers with different scales were concatenated together, and then some downsampling blocks were used to generate new feature pyramids. Finally, the features were fed to multibox detector for prediction. The feature fusion in FSSD increased the detection performance for both large and small objects. Zhu et al. [85] trained single-shot object detectors from scratch and obtained state-of-the-art performance on various benchmark datasets. They removed the first downsampling layer of SSD and introduced root block (with modified convolutional filters) to exploit more local information from an image. Therefore, the detector was able to extract powerful features for small object detection.

All of the aforementioned works were proposed for natural images. A method related to small object detection on remote sensing imagery was proposed by Yang et al. [76]. They used modified faster R-CNN to detect both large and small objects. They proposed rotation dense feature pyramid networks (R-DFPN), and the use of this network helped to improve the detection performance of small objects.

There is an excellent review paper by Zhao et al. [83], where the authors showed a thorough review of object detectors and also showed the advantages and disadvantages of different object detectors. The effect of object size was also discussed in the paper. Another survey paper about object detection in remote sensing images by Li et al. [38] showed review and comparison of different methods.

2.3 Simultaneous Super-resolution with Object Detection

The positive effects of SR on object detection tasks was discussed in [64] where the authors used remote sensing datasets for their experiments. Simultaneous CNN-based image enhancement with object detection using single-shot multibox detector (SSD) [49] was done in [5]. Haris et al. [20] proposed a deep CNN-based generator to generate a HR image from a LR image and then used a multi-task network as a discriminator and also for localization and classification of objects. These works were done on natural images, and LR and HR image pairs were required. In another work [25], a method using simultaneous super-resolution with object detection on satellite imagery was proposed. The SR network in this approach was inspired by the cycle-consistent adversarial network [84]. A modified faster R-CNN architecture was used to detect vehicles from enhanced images produced by the SR network.

2.4 Labeling Tools

There are many labeling tools to label ground and satellite imagery. ArcGIS [4] software has a tool to annotate and detect objects from satellite imagery, and also it can train/test models using the annotated data. Labelbox [33] is another software that provides similar types of services like the ArcGis tool and services are provided online. ENVI [14] also provides similar types of services as the two previously mentioned software. ArcGis and ENVI are heavy in terms of resource allocation and all the three software charge users for the services. There are also open-source labeling tools such as LabelImg [72] and computer vision annotation tool (CVAT) [33]. They are easy to use and light-weight, and users can annotate small to medium size images.

Chapter 3

A Tool for Labeling and Detection

3.1 Introduction

We have created a GUI tool called Label-Detect [24]. It is a graphical image annotation tool and a user can also train and test large satellite images. Users can create small tiles of images from a large remote sensing image, annotate it, create training and testing data, select model, and train-test the model. The final detection results can be generated as geographical coordinate files, which is very helpful in plotting the results on many remote sensing software.

The annotations (labeling) part of this application is based on LabelImg [72] GitHub repository. It is written in Python and uses Qt for its graphical interface. Tensorflow object detection API [23] is used for training and testing the models.

Annotations are saved as XML files in PASCAL VOC [15] format, the format used by ImageNet [62]. Users can use many deep learning models, such as Faster RCNN [60] with Resnet [44] or SSD [49] with VGG [65]. We show an example of the options of our tool and detection of objects on large satellite imagery in figure 3.1 and 3.2. Labeling, training, and testing procedures are described and depicted in the next sections.

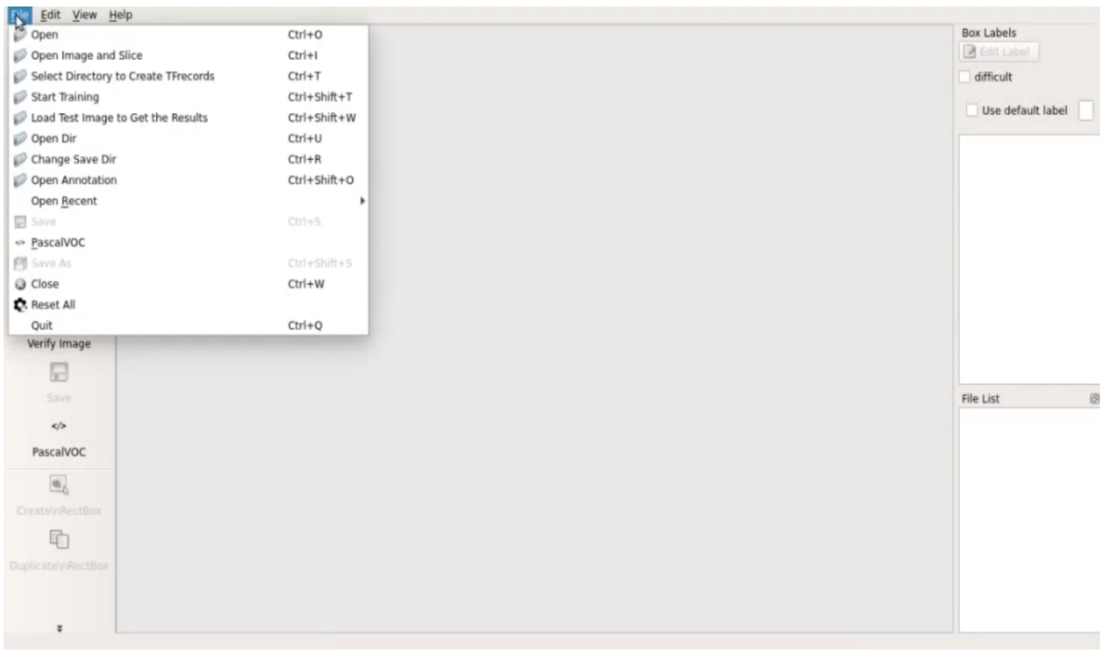


Figure 3.1: Our labeling tool - Label-Detect.

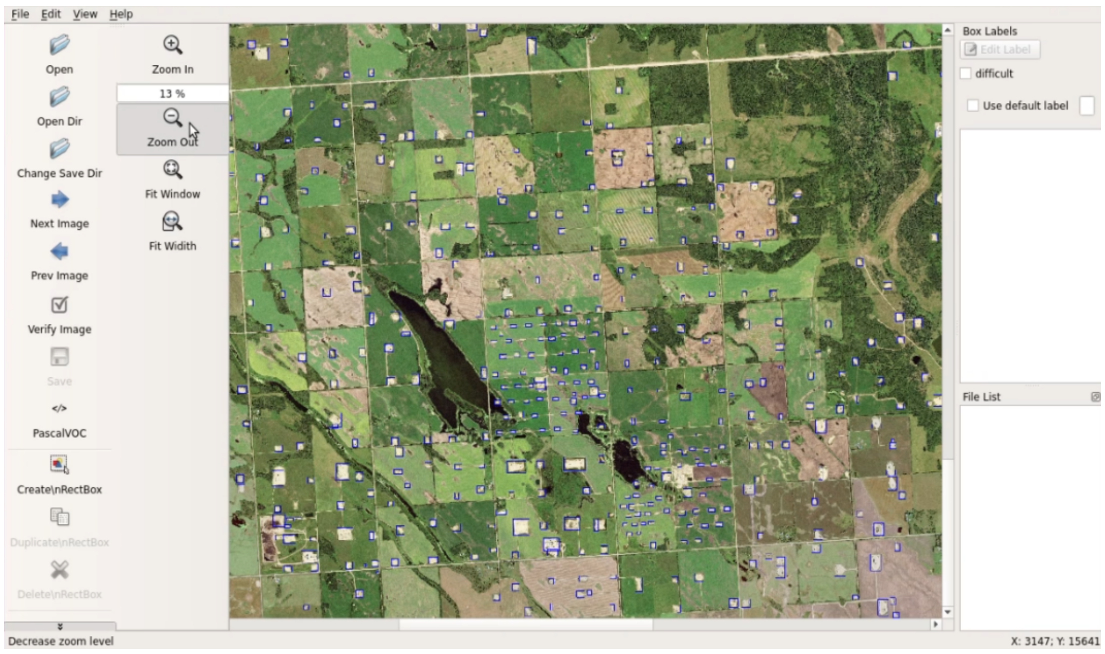


Figure 3.2: Detection of objects on a large satellite image.

3.2 How to Use the Tool

3.2.1 Annotation

Steps (PascalVOC Style Annotation)

- Build and launch using the instructions in the GitHub repository [24].
- Select 'Change default saved annotation folder' in Menu/File to save the annotation files from user specified directory.
- Select 'Open Dir,' and users can see the images from the selected directory.
- Select 'Create RectBox' to create annotation.
- Click and release the left mouse to select a region to annotate the 'rect box.'

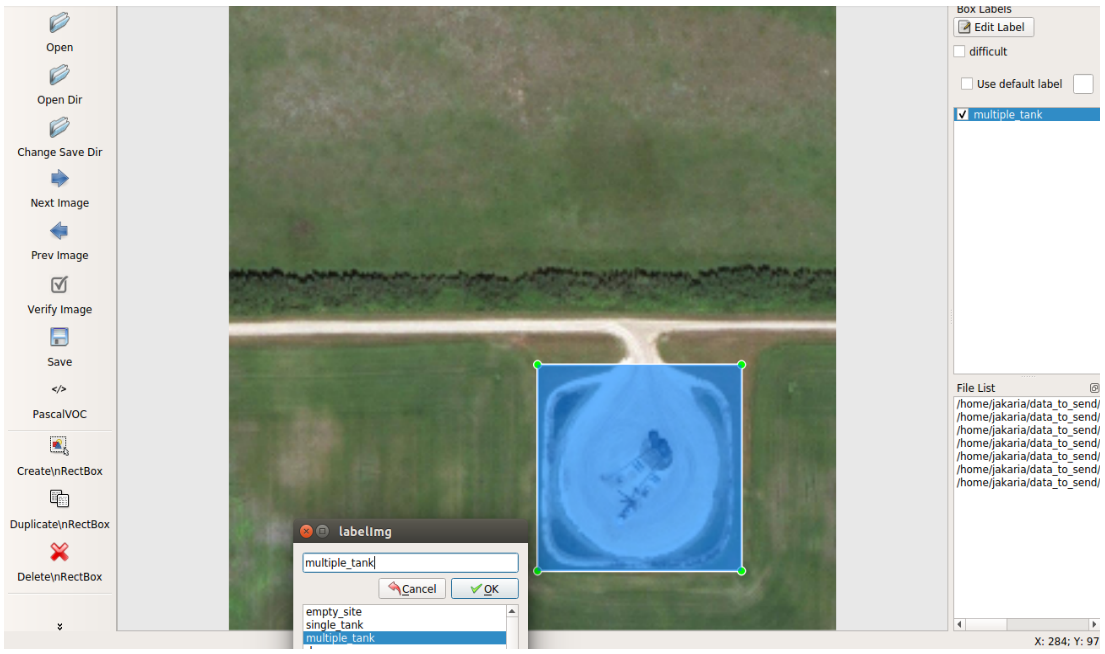


Figure 3.3: An Example of labeling of an object is shown here.

Users can use right mouse button to drag the ‘rect box’ to copy or move it. An example screenshot of labeling is shown in figure 3.3.

3.2.2 Training

Steps

- Select ‘File →Open Image and Slice’ [Ctrl+i]
- Select the desired satellite Image and then enter the tile/slice height and width. The default value is 512 pixels.
- Then select ‘Start Slicing’.
- After slicing the big image into small tiles, users can see a new directory on the image’s directory, and within it, users can see image tiles/slices.
- Annotate the images and save the .xml files according to the ‘Annotation’ section discussed above.
- Select ‘File →Select Directory to Create TFrecords’ [Ctrl+t] and select the directory that contains all the .xml files.
- Then TFRecords files for training and testing will be created under TFrecords folder within the directory selected in the previous step.
- Select ‘Start Training’ [Ctrl+Shift+t]
- Select the TFRecord file for training, which is ‘train.record’.
- Select ‘detection.pbtxt’ and a ‘.config’ file from ‘Label-Detect/Training_config’ directory. If users want to use Faster R-CNN ResNet-101 then they can select the corresponding file otherwise one can select the ‘.config’ file for SSD MobileNet. Apart from these two models, other models available in the object detection API also can be used.

- Download the Faster R-CNN Resnet-101 model, extract it and select the ‘model.ckpt.index’ file for the model file. Users can also use SSD MobileNet or other models.

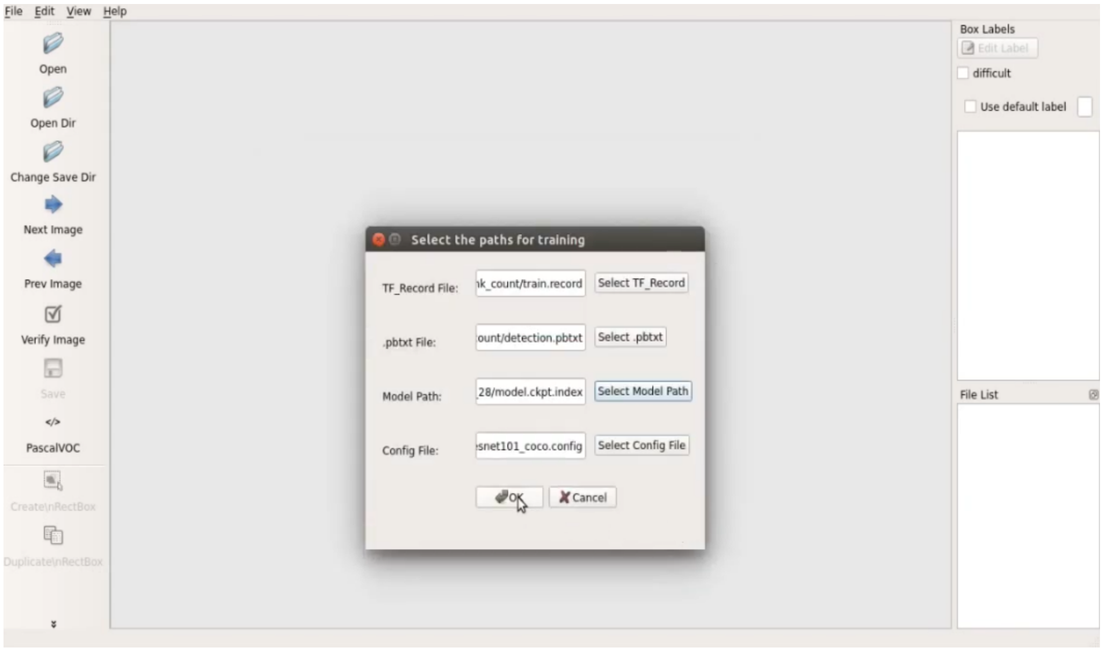
- If users want to use other models they can download from the tensorflow object detection API repository and the corresponding .config files from the same repository.

- Then one can start training, and after the completion of the training users can get ‘frozen_inference_graph.pb’ file, and this file can be used for testing images.

- By default, the training is for three labels which are in ‘detection.pbtxt’ file. If users want to create their own labels, they can edit the ‘detection.pbtxt’ file, give an item id starting from value 1 and give it a name. The format must be the same as the given ‘detectin.pbtxt’ file.

- In the config files that are provided in the repository in ‘Label-Detect/Training_config’, users can find ‘num_steps: 20000’. This is the number of training steps. Users can change the steps to their desired values.

We train different models using our tool. In figure 3.4, we show a screenshot of selecting different files required for training.



TF_Record File: k_count/train.record

.pbtxt File: ount/detection.pbtxt

Model Path: 28/model.ckpt.index

Config File: snet101_coco.config

Figure 3.4: Selection of required files before training.

3.2.3 Testing

Steps

- Select 'File →Load Test Image to Get the Results'
- For testing large images, the images must be sliced into small overlapping tiles for detection. Therefore, users need to enter the height and width of the slices. The default size is 512 for height and width.
- Then select the 'frozen_inference_graph.pb' file that is created in the training phase.
- Then after some processing time (slicing, detection of objects, conversion of local coordinates of the detected bounding box to the global coordinate and application of non-max suppression to the overlapped detection), the final labeled image with bounding boxes can be seen within the tool. The files with geolocation of the bounding boxes are also generated.

In figure 3.5, we show some example detections with labels.

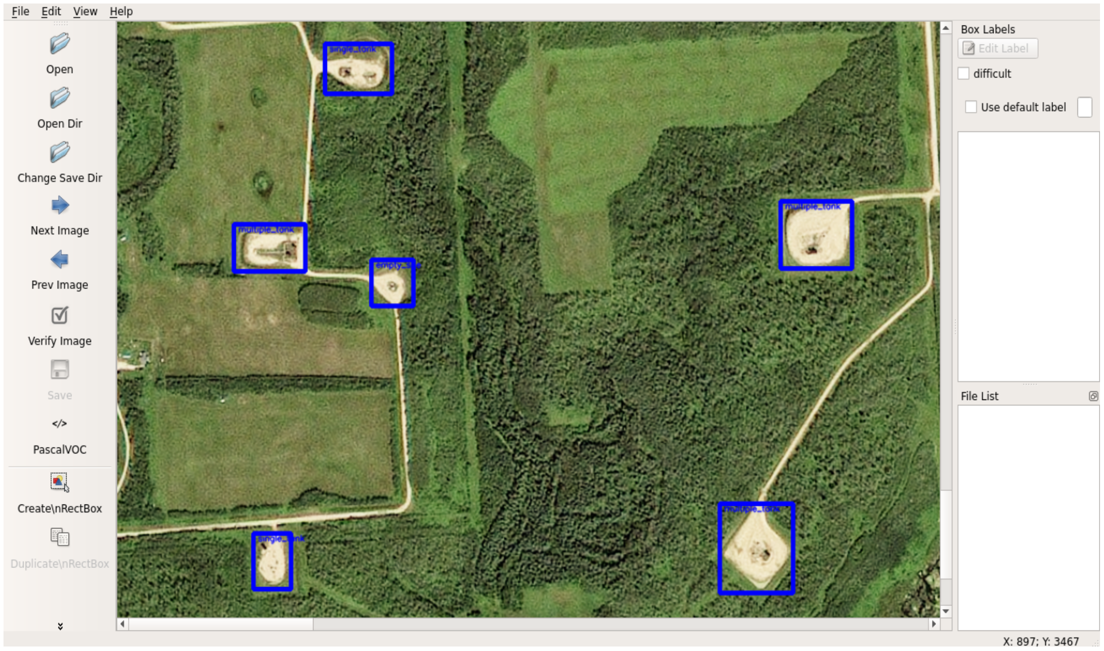


Figure 3.5: An Example of the detections of objects is shown here.

Chapter 4

An End-to-End Deep Learning Architecture for Small-Object Detection in Remote Sensing Images

4.1 Method

In this thesis, we aim to improve the detection performance of small objects on remote sensing imagery. Towards this goal, we propose an end-to-end network architecture that consists of two modules: A GAN based SR network and a detector network. The whole network is trained in an end-to-end manner and HR and LR image pairs are needed for training.

The SR network has three components: generator (G), discriminator (D_{Ra}), and edge-enhancement network (EEN). Our method uses end-to-end training as the gradient of the detection loss from the dectector is backpropagated into the generator. Therefore, the detector also works like a discriminator and encourages the generator G to generate realistic images similar to the ground truth. Our entire network structure can also be divided into two parts: A generator consisting of the EEN and a discriminator, which includes the D_{Ra} and the detector network. In figure 4.1, we show the role of the detector as a discriminator.

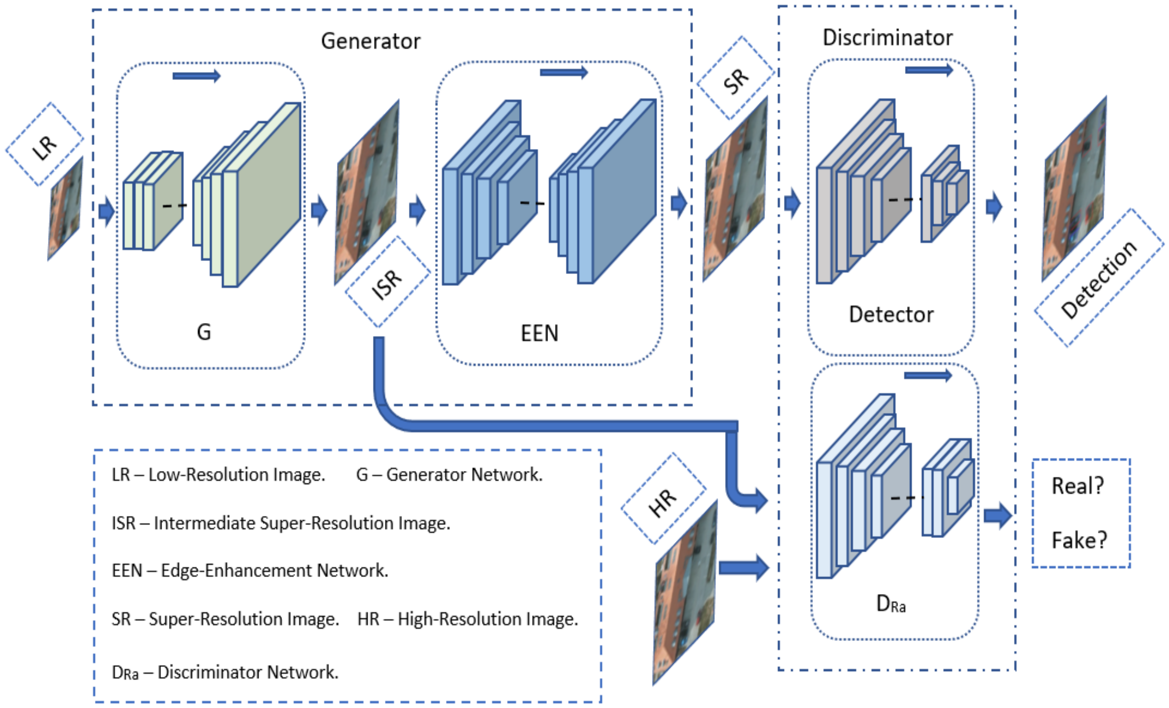

Generator
LR
G
ISR
EEN
SR
Discriminator
Detector
Detection
HR
DRa
Real?
Fake?
LR – Low-Resolution Image.
G – Generator Network.
ISR – Intermediate Super-Resolution Image.
EEN – Edge-Enhancement Network.
SR – Super-Resolution Image.
HR – High-Resolution Image.
DRa – Discriminator Network.

Figure 4.1: Overall network architecture with a generator and a discriminator module.

The generator G generates intermediate super-resolution (ISR) images, and then final SR images are generated after applying the EEN network. The discriminator (D_{Ra}) discriminates between ground truth (GT) HR images and ISR. The inverted gradients of D_{Ra} are backpropagated into the generator G in order to create SR images allowing for accurate object detection. Edge information is extracted from ISR, and the EEN network enhances these edges. Afterwards, the enhanced edges are again added to the ISR after subtracting the original edges extracted by the Laplacian operator and we get the output

SR images with enhanced edges. Finally, we detect objects from the SR images using the detector network.

We use two different loss functions for EEN: one compares the difference between SR and ground truth images, and the other compares the difference between the extracted edge from ISR and ground truth. We also use the VGG19 [65] network for feature extraction that is used for perceptual loss [73]. Hence, it generates more realistic images with more accurate edge information. We divide the whole pipeline as a generator, and a discriminator, and these two components are elaborated in the following.

4.1.1 Generator

Our generator consists of a generator network G and an edge-enhancement network EEN. In this section, we describe the architectures of both networks and the corresponding loss function.

4.1.1.1 Generator Network G

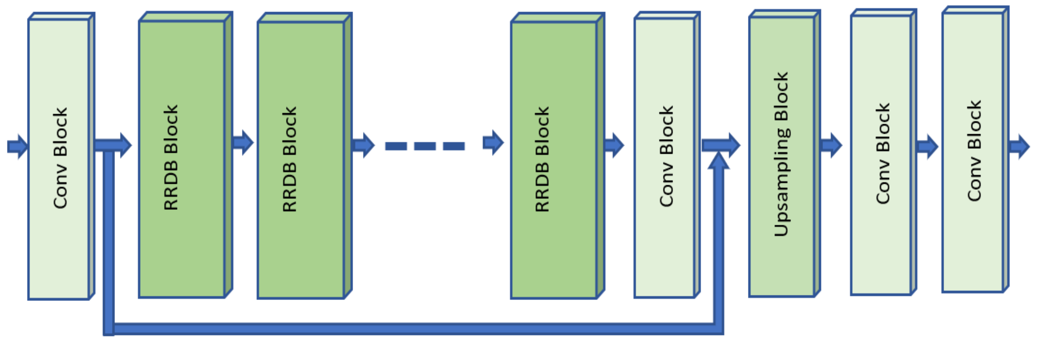
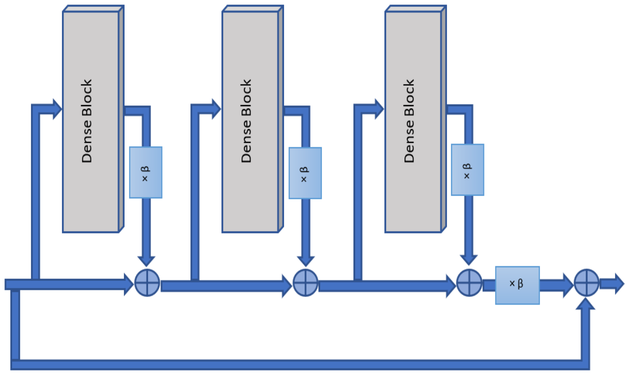


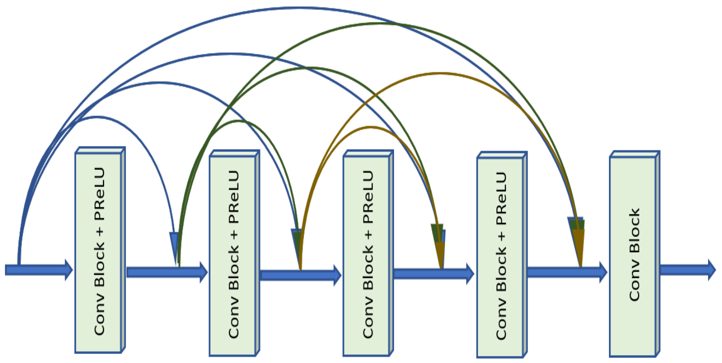
Figure 4.2: Generator G with RRDB (residual-in-residual dense blocks), convolutional and upsampling blocks.

We use the generator architecture from ESRGAN [73], where all batch normalization (BN) layers are removed, and RRDB is used. The overall architecture of generator G is shown in figure 4.2, and the RRDB is depicted in figure 4.3.

Inspired by the architecture of ESRGAN, we remove BN layers to increase the performance of the generator G and to reduce the computational complexity. The authors of ESRGAN also state that the BN layers tend to introduce unpleasant artifacts and limit the generalization ability of the generator when the statistics of training and testing datasets differ significantly.



(a) RRDB from generator.



(b) Dense block from RRDB.

Figure 4.3: Internal diagram of RRDB (residual-in-residual dense blocks).

We use RRDB as the basic blocks of the generator network G that uses a multi-level residual network with dense connections. Those dense connections increase network capacity, and we also use residual scaling to prevent unstable conditions during the training phase [73]. We use the parametric rectified linear unit (PReLU) [22] for the dense blocks to learn the parameter with the other neural network parameters. As discriminator (D_{Ra}), we employ a relativistic average discriminator similar to the work represented in [73].

In equation 4.1 and 4.2, the relativistic average discriminator is formulated for our architecture. Our generator G depends on the discriminator D_{Ra}, and hence we briefly discuss the discriminator D_{Ra} here and then, describe all details in section 4.1.2. The discriminator predicts the probability that a real image (I_{HR}) is relatively more realistic than a generated intermediate image (I_{ISR}).

$$D_{Ra}(I_{HR}, I_{ISR}) = \sigma(C(I_{HR}) - \mathbb{E}_{I_{ISR}}[C(I_{ISR})]) \rightarrow 1 \quad \text{More Realistic than fake data?} \tag{4.1}$$

$$D_{Ra}(I_{ISR}, I_{HR}) = \sigma(C(I_{ISR}) - \mathbb{E}_{I_{HR}}[C(I_{HR})]) \rightarrow 0 \quad \text{Less realistic than real data?} \tag{4.2}$$

In equation 4.1 and 4.2, σ, $C(\cdot)$ and $\mathbb{E}_{I_{ISR}}$ represents the sigmoid function, discriminator output and operation of calculating mean for all generated intermediate images in a mini-batch. The generated intermediate images are created by the generator where $I_{ISR} = G(I_{LR})$. It is evident from equation 4.3 that the adversarial loss of the generator contains both I_{HR} and I_{ISR} and hence, it benefits from the gradients of generated and ground truth images during the training process. The discriminator loss is depicted in equation 4.4.

$$L_G^{Ra} = -\mathbb{E}_{I_{HR}}[\log(1 - D_{Ra}(I_{HR}, I_{ISR}))] - \mathbb{E}_{I_{ISR}}[\log(D_{Ra}(I_{ISR}, I_{HR}))] \tag{4.3}$$

$$L_D^{Ra} = -\mathbb{E}_{I_{HR}}[\log(D_{Ra}(I_{HR}, I_{ISR}))] - \mathbb{E}_{I_{ISR}}[\log(1 - D_{Ra}(I_{ISR}, I_{HR}))] \tag{4.4}$$

We use two more losses for generator G: one is perceptual loss (L_{percep}), and another is content loss (L_1) [73]. The perceptual loss is calculated using the feature map ($vgg_{fea}(\cdot)$) before the activation layers of a fine-tuned VGG19 [65] network, and the content loss calculates the 1-norm distance between I_{ISR} and I_{HR}. Perceptual loss and content loss is shown in equation 4.5 and equation 4.6.

$$L_{percep} = \mathbb{E}_{I_{LR}} ||vgg_{fea}(G(I_{LR}) - vgg_{fea}(I_{HR})||_1 \tag{4.5}$$

$$L_1 = \mathbb{E}_{I_{LR}} ||G(I_{LR}) - I_{HR}||_1 \tag{4.6}$$

4.1.1.2 Edge Enhancement Network EEN

The EEN network removes noise and enhances the extracted edges from an image. An overview of the network is depicted in figure 4.4. In the beginning, Laplacian operator [29] is used to extract edges from the input image. After the edge information is extracted, it is passed through convolutional, RRDB, and upsampling blocks. There is a mask branch with sigmoid activation to remove edge noise as described in [27]. Finally, the enhanced edges are added to the input images where the edges extracted by the Laplacian operator were subtracted.

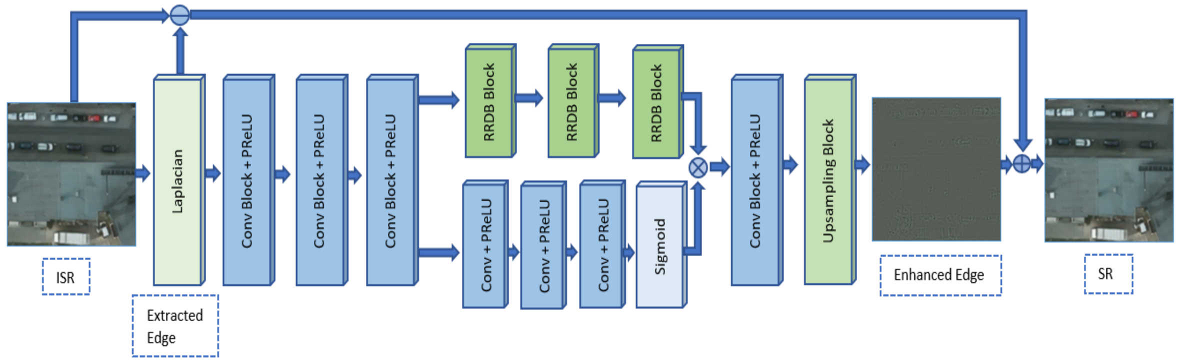


Figure 4.4: Edge-enhancement network where input is an ISR (intermediate super-resolution) image and output is a SR (super-resolution) image.

The EEN network is similar to the edge-enhancement subnetwork proposed in [27] with two improvements. First, we replace the dense blocks with RRDB.

The RRDB shows improved performance according to ESRGAN [73]. Hence, we replace the dense block for improved performance of the EEN network. Secondly, we introduce a new loss term to improve the reconstruction of the edge information. In [27], authors extracted the edge information from I_{ISR} and enhanced the edges using an edge-enhancement subnetwork which is afterwards added to the edge-subtracted I_{ISR}. To train the network, [27] proposed to use Charbonnier loss [8] between the I_{ISR} and I_{HR}. This function is called consistency loss for images (L_{img_cst}) and helps to get visually pleasant outputs with good edge information. However, sometimes the edges of some objects are distorted and produce some noises and consequently, do not give good edge information. Therefore, we introduce a consistency loss for the edges (L_{edge_cst}) as well. To compute L_{edge_cst} we evaluate the Charbonnier loss between the extracted edges (I_{edge_SR}) from I_{SR} and the extracted edges (I_{edge_HR}) from I_{HR}. The two consistency losses are depicted in equation 4.7 and equation 4.8 where $\rho(\cdot)$ is the Charbonnier penalty function [34]. The total consistency loss is finally calculated for both images and edges by summing up the individual loss. The loss of our EEN is shown in equation 4.9.

$$L_{img\_cst} = \mathbb{E}_{I_{SR}}[\rho(I_{HR} - I_{SR})] \tag{4.7}$$

$$L_{edge\_cst} = \mathbb{E}_{I_{edge\_SR}}[\rho(I_{edge\_HR} - I_{edge\_SR})] \tag{4.8}$$

$$L_{een} = L_{img\_cst} + L_{edge\_cst} \tag{4.9}$$

Finally, we get the overall loss for the generator module by adding the losses of the generator G and the EEN network. The overall loss for the generator module is shown in equation 4.10 where λ_1, λ_2, λ_3, and λ_4 are the weight parameters to balance different loss components. We empirically set the values as $\lambda_1 = 1$, $\lambda_2 = .001$, $\lambda_3 = .01$, and $\lambda_4 = 5$.

$$L_{G\_een} = \lambda_1 L_{percep} + \lambda_2 L_G^{Ra} + \lambda_3 L_1 + \lambda_4 L_{een} \tag{4.10}$$

4.1.2 Discriminator

As described in the previous section, we use the relativistic discriminator D_{Ra} for training the generator G. The architecture of the discriminator is taken from ESRGAN [73] which employs the VGG-19 [65] architecture. We use Faster R-CNN [60] and SSD [49] for our detector networks. The discriminator (D_{Ra}) and the detector network jointly act as discriminator for the generator module. We briefly describe these two detectors in the next two sections.

4.1.2.1 Faster R-CNN

The Faster R-CNN [60] is a two-stage object detector and contains two networks: a region proposal network (RPN) to generate region proposals from an image and another network to detect objects from these proposals. In addition, the second network also tries to fit the bounding boxes around the detected objects.

The task of the RPN is to return image regions that have a high probability of containing an object. The RPN network uses a backbone network such as VGG [65], ResNet, or ResNet with feature pyramid network [44]. These networks are used as feature extractors, and different types of feature extractors can be chosen based on their performance on public datasets. We use ResNet-50-FPN [44] as a backbone network for our faster R-CNN. We use this network because it displayed a higher precision than VGG-19 and ResNet-50 without FPN (especially for small object detection) [44]. Even though the use of a larger network might lead to a further performance improvement, we chose ResNet-50-FPN due to its comparably moderate hardware requirements and more efficient convergence times.

After the RPN, there are two branches for detection: a classifier and a regressor. The classification branch is responsible for classifying a proposal to a specific object, and the regression branch finds the accurate bounding box of the object. In our case, both datasets contain objects with only one class, and therefore, our classifier infers only two classes: the background class and the object class.

4.1.2.2 SSD

The SSD [49] is a single-shot multibox detector that detects objects in a single stage. Here, single-stage means that classification and localization are done in a single forward pass through the network. Like Faster R-CNN, SSD also has a feature extractor network, and different types of networks can be used. To serve the primary purpose of SSD, which is speed, we use VGG-16 [65] as a feature extractor network. After this network, SSD has several convolutional feature layers of decreasing sizes. This representation can seem like a pyramid representation of images at different scales. Therefore, the detection of objects happens in every layer, and finally, we get the object detection output as class values and coordinates of bounding boxes.

4.1.2.3 Loss of the discriminator

The relativistic discriminator loss (L_D^{Ra}) is already described in the previous section and depicted in equation 4.4. This loss is added to the detector loss to get the final discriminator loss.

Both Faster R-CNN and SSD have similar regression/localization losses but different classification losses. For regression/localization, both use smooth L_1 [60] loss between detected and ground truth bounding box coordinates

(t_*). Classification (L_{cls_frcnn}) and regression loss (L_{reg_frcnn}) and overall loss (L_{det_frcnn}) of Faster R-CNN are given in the following:

$$L_{cls\_frcnn} = \mathbb{E}_{I_{LR}}[-\log(Det_{cls\_frcnn}(G_{G\_een}(I_{LR})))] \tag{4.11}$$

$$L_{reg\_frcnn} = \mathbb{E}_{I_{LR}}[smooth_{L1}(Det_{reg\_frcnn}(G_{G\_een}(I_{LR})), t_*)] \tag{4.12}$$

$$L_{det\_frcnn} = L_{cls\_frcnn} + \lambda L_{reg\_frcnn} \tag{4.13}$$

Here, λ is used to balance the losses, and it is set to 1 empirically. Det_{cls_frcnn} and Det_{reg_frcnn} are the classifier and regressor for the Faster R-CNN. Classification (L_{cls_ssd}), regression loss (L_{reg_ssd}) and overall loss (L_{det_ssd}) of SSD are as following:

$$L_{cls\_ssd} = \mathbb{E}_{I_{LR}}[-\log(softmax(Det_{cls\_ssd}(G_{G\_een}(I_{LR}))))] \tag{4.14}$$

$$L_{reg\_ssd} = \mathbb{E}_{I_{LR}}[smooth_{L1}(Det_{reg\_ssd}(G_{G\_een}(I_{LR})), t_*)] \tag{4.15}$$

$$L_{det\_ssd} = L_{cls\_ssd} + \alpha L_{reg\_ssd} \tag{4.16}$$

Here, α is used to balance the losses, and it is set to 1 empirically. Det_{cls_ssd} and Det_{reg_ssd} are the classifier and regressor for the SSD.

4.1.3 Training

Our architecture can be trained in separate steps or jointly in an end-to-end way. We discuss the details of these two types of training in the next two sections.

4.1.3.1 Separate Training

In separate training, we train the SR network (generator module and discriminator D_{Ra}) and the detector separately. Detector loss is not backpropagated to the generator module. Therefore, the generator is not aware of the detector and thus, it only gets feedback from the discriminator D_{Ra}. For example, in equation 4.11, no error is backpropagated to the G_{G_een} network (the network is detached during the calculation of the detector loss) while calculating the loss L_{cls_frcnn}.

4.1.3.2 End-to-End Training

In end-to-end training, we train the whole architecture end-to-end that means the detector loss is backpropagated to the generator module. Therefore, the generator module revceives gradients from both detector and discriminator D_{Ra}. We get the final discriminator loss (L_{D_det}) as following:

$$L_{D\_det} = L_D^{Ra} + \eta L_{det} \tag{4.17}$$

Here, η is the parameter to balance the contribution of the detector loss and we empirically set it to 1. Detection loss from SSD or Faster R-CNN is denoted by L_{det}. Finally, we get an overall loss ($L_{overall}$) for our architecture as follows.

$$L_{overall} = L_{G\_een} + L_{D\_det} \tag{4.18}$$

4.2 Experiments

As mentioned above, we trained our architecture separately and in an end-to-end manner. For separate training, we first trained the SR network until convergence and then trained the detector networks based on the SR images. For end-to-end training, we also employed separate training as pre-training step for weight initialization. Afterwards SR and object detection networks were jointly trained, i.e., the gradients from the the object detector were propagated into the generator network.

In the training process, the learning rate was set to 0.0001 and halved after every $50k$ iterations. The batch size was set to 5. We used Adam [31] as optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and updated the whole architecture weights until convergence. We used 23 RRDB blocks for the generator G and 5 RRDB blocks for the EEN network. We implemented our architecture with the PyTorch framework [55] and trained/tested using two NVIDIA Titan X GPUs. The end-to-end training with COWC took 96 hours for 200 epochs. The average inference speed using faster R-CNN was approximately 4 images/second and 7 images/second for SSD. Our implementation can be found in GitHub [56].

4.2.1 Datasets

4.2.1.1 Cars Overhead with Context Dataset

Cars overhead with context (COWC) dataset [52] contains 15 cm (one pixel cover 15 cm distance at ground level) satellite images from six different regions. The dataset contains a large number of unique cars and covers regions from Toronto in Canada, Selwyn in New Zealand, Potsdam and Vaihingen in Germany, Columbus and Utah in the United States. Out of these six regions, we used the dataset from Toronto and Potsdam. Therefore, when we refer to the COWC dataset, we refer to the dataset from these two regions. There are 12651 cars in our selected dataset. The dataset contains only RGB images, and we used these images for training and testing.

We used 256-by-256 image tiles, and every image tile contains at least one car. The average length of a car was between 24 to 48 pixels, and the width was between 10 to 20 pixels. Therefore, the area of a car was between 240 to 960 pixels, which can be considered as a small object relative to the other large satellite objects. We used bi-cubic downsampling to generate LR images from the COWC dataset. The downscale factor was 4×, and therefore, we had 64 pixels to 64 pixels size for LR images. We had a text file associated with each image tile containing the coordinates of the bounding box for each car.

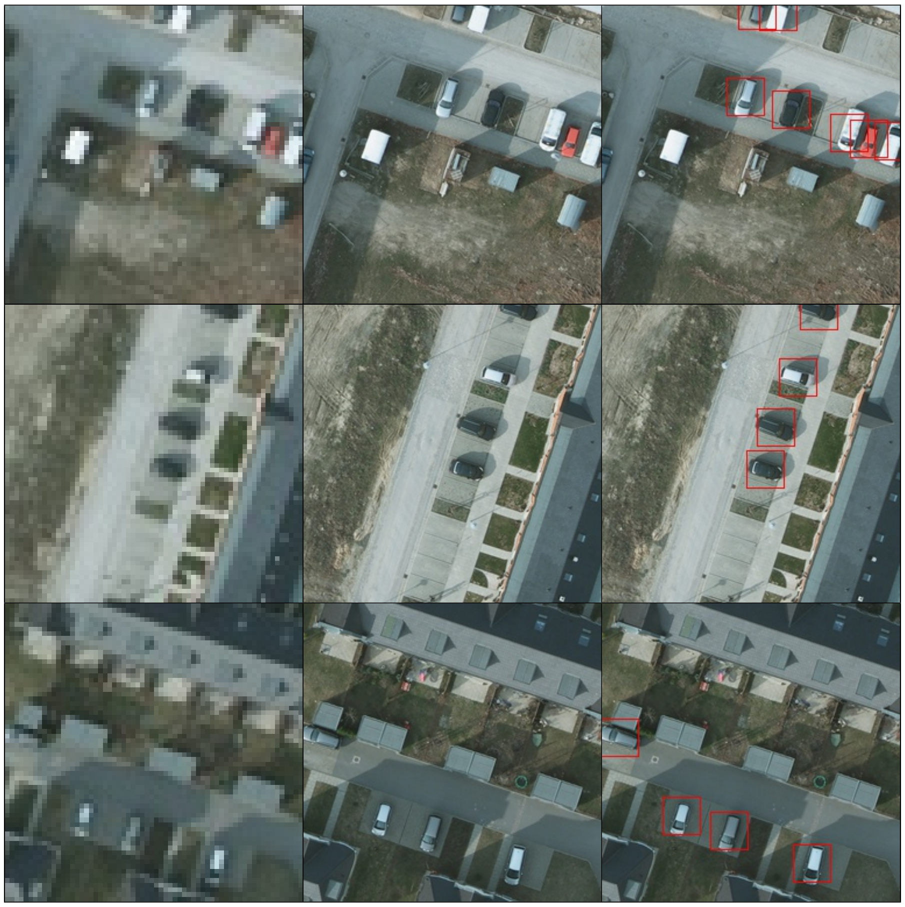


Figure 4.5: COWC (car overhead with context) dataset: LR-HR (low-resolution and high-resolution) image pairs are shown in (a) and (b) and GT (ground truth) images with bounding boxes for cars are in (c).

Our experiments considered the dataset having only one class, car, and did not consider any other type of object. Figure 4.5 shows examples from the COWC dataset. We experimented with a total of 3340 tiles for training and testing. Our train/test split was 80%/20%, and the training set was further divided into a training and a validation set by an 80% to 20% ratio. We trained our end-to-end architecture with an augmented training dataset with random horizontal flips and ninety-degree rotations.

4.2.1.2 Oil and Gas Storage Tank Dataset

The oil and gas storage tank (OGST) dataset has been complied in Alberta Geological Survey (AGS) [2], a branch of the Alberta Energy Regulatory (AER) [1]. AGS provides geoscience information and support to AER's regulatory functions on energy developments to be carried out in a manner to ensure public and environmental safety. To assist AER with sustainable land management and compliance assurance [10], AGS is utilizing remote sensing imagery for identifying the number of oil and gas storage tanks inside well pad footprints in Alberta.

While the SPOT-6 satellite imagery at 1.5 m pixel resolution provided by the AGS has sufficient quality and details for many regulatory functions, it is difficult to detect small objects within well pads, e.g., oil and gas storage tanks with ordinary object detection methods. The diameter of a typical storage tank is about 3 m and their placements are usually vertical and side-by-side with less than 2 m. To train our architecture for this use-case, we needed a dataset for providing pairs of low and high-resolution images. Therefore, we have created the OGST dataset using free imagery from the Bing map [6].

The OGST dataset contains 30 cm resolution remote sensing images (RGB) from the Cold Lake Oil Sands region of Alberta, Canada where there is a high level of oil and gas activities and concentration of well pad footprints. The dataset contains 1671 oil and gas storage tanks from this area.

We used 512-by-512 image tiles, and there was no image without any oil and gas storage tank in our experiment. The average area covered by an individual tank was between 800 to 1600 pixels. Some industrial tanks were large, but most of the tanks covered small regions on the imagery. We downscaled the HR images using bi-cubic downsampling with the factor of $4\times$, and therefore, we got a LR tile of size 128-by-128 pixels. Every image tile was associated with a text file containing the coordinates of the bounding boxes for the tanks on a tile. We have showed examples from the OGST dataset in figure 4.6.

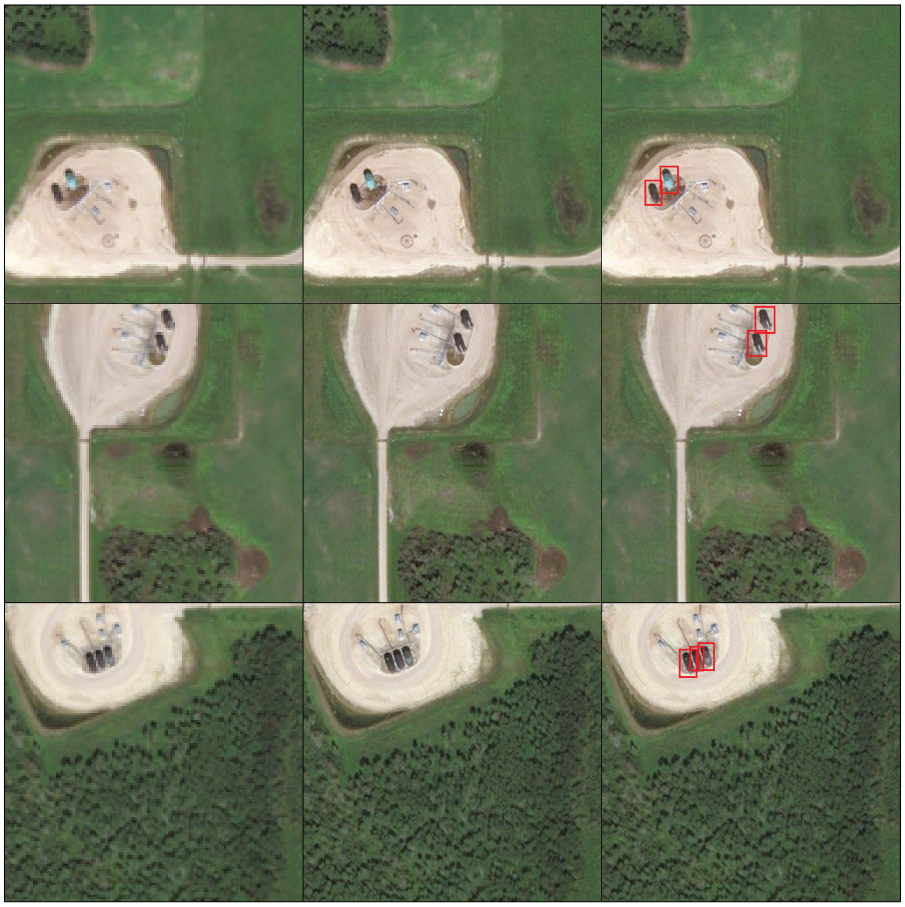

(a) LR image (b) HR image (c) GT image

Figure 4.6: OGST (oil and gas storage tank) dataset: LR-HR (low-resolution and high-resolution) image pairs are shown in (a) and (b) and GT (ground truth) images with bounding boxes for oil and gas storage tanks are in (c).

As with the COWC dataset, our experiments considered one unique class here, tank, and we had a total of 760 tiles for training and testing. We used a 90%/10% split for our train/test data. The training data was further divided by 90%/10% for the train/validation split. The percentage of training data was higher here compared to the previous dataset to increase the training data because of the smaller size of the dataset. The dataset is available at [56].

4.2.2 Evaluation Metrics for Detection

We obtained our detection output as bounding boxes with associated classes. To evaluate our results, we used average precision (AP), and calculated intersection over union (IoU), precision, and recall for obtaining AP.

We denote the set of correctly detected objects as true positives (TP) and the set of falsely detected objects of false positives (FP). The precision is now the ratio between the number of TPs relative to all predicted objects:

$$Precision = \frac{|TP|}{|TP| + |FP|} \tag{4.19}$$

We denote the set of objects which are not detected by the detector as false negatives (FN). Then, the recall is defined as the ratio of detected objects (TP) relative to the number of all objects in the data set:

$$Recall = \frac{|TP|}{|TP| + |FN|} \tag{4.20}$$

To measure the localization error of predicted bounding boxes, IoU computes the overlap between two bounding boxes: the detected and the ground truth box. If we take all the boxes that have an IoU $\geq \tau$ as TP and consider all other detections as FP, then we get the precision at τ IoU. AP at a specific IoU is calculated by averaging precision values from different recall values. If we now vary τ from 0.5 to 0.95 IoU with a step size of 0.05, we receive ten different precision values which can be combined into the average precision (AP) at IoU=0.5:0.95 [60].

Let us note that in the case of multi-class classification, we would need to compute the AP for object each class separately. To receive a single performance measure for object detection, the mean AP (mAP) is computed which is the most common performance measure for object detection quality.

In this thesis, both of our datasets only contain single class, and hence, we used AP as our evaluation metric. We mainly showed the results of AP at IoU=0.5:0.95 as our method performed increasingly better compared to other models when we increased the IoU values for AP calculation. We show this trend in section 4.2.3.4.

4.2.3 Results

4.2.3.1 Detection without Super-Resolution

We ran the two detectors to document the object detection performance on both LR and HR images. We used SSD with vgg16 [65] network and Faster R-CNN (FRCNN) with ResNet-50-FPN [44] detector. We trained the two models with both HR and 4×-downscaled LR images. Testing was also done with both HR and LR images.

In table 4.1, we show the results of the detection performance of the detectors with different train/test combinations. When we only used LR images for both training and testing, we observed 64% AP for Faster R-CNN. When training on HR images and testing with LR images, the AP dropped for both detectors. We also added detection results (using LR images for training/testing) for both the datasets using SSD with RFB modules (SSD-RFB) [48], where AP slightly increased from the base SSD.

The last two rows in table 4.1 depict the AP of both detectors when training and testing on HR images. We have achieved up to 98% AP with the Faster R-CNN detector. This, shows the large impact of the resolution to the object detection quality and sets a natural upper bound on how close a SR-based method can get when working on LR images. Mundhenk et al. [52] reported similar types of detection results using the COWC dataset. In the next sections, we demonstrate that our approaches considerably improve the detection rate on LR imagery and get astonishingly close to the performance of directly working on HR imagery.

Table 4.1: Detection on LR (low-resolution) and HR (high-resolution) images without using super-resolution. Detectors are trained with both LR and HR images and AP (average precision) values are calculated using 10 different IoUs (intersection over union).

Model	**Training image resolution - Test image resolution**	**COWC Dataset (Test Results) (AP at IoU=0.5:0.95) (single class - 15cm)**	**OGST Dataset (Test Results) (AP at IoU=0.5:0.95) (single class - 30cm)**
SSD	LR - LR	61.9%	76.5%
	HR - LR	58%	75.3%
FRCNN	LR - LR	64%	77.3%
	HR - LR	59.7%	75%
SSD-RFB	LR - LR	63.1%	76.7%
SSD	HR - HR	94.1%	82.5%
FRCNN	HR - HR	98%	84.9%

4.2.3.2 Separate Training with Super-Resolution

In this experiment, we created 4× upsampled images from the LR input images using bicubic upsampling and different SR methods. Let us note that no training was needed for applying bicubic upsampling since it is a parameter free function. We used the SR images as test data for two types of detectors. We compared three GAN architectures for generating SR images, our new

EESRGAN architecture, ESRGAN [73] and EEGAN [27]. Each network was trained separately on the training set before the object detector was trained. For the evaluation, we again compared detectors being trained on the SR images from the particular architecture and detectors being directly trained on the HR images.

Table 4.2: Detection on SR (super-resolution) images with separately trained SR network. Detectors are trained with both SR and HR (high-resolution) images and AP (average precision) values are calculated using 10 different IoUs (intersection over union).

Model	**Training image resolution - Test image resolution**	**COWC Dataset (Test Results) (AP at IoU=0.5:0.95) (single class - 15cm)**	**OGST Dataset (Test Results) (AP at IoU=0.5:0.95) (single class - 30cm)**
Bicubic + SSD	SR - SR	72.1%	77.6%
	HR - SR	58.3%	76%
Bicubic + FRCNN	SR - SR	76.8%	78.5%
	HR - SR	61.5%	77.1%
EESRGAN + SSD	SR - SR	86%	80.2%
	HR - SR	83.1%	79.4%
EESRGAN + FRCNN	SR - SR	**93.6%**	**81.4%**
	HR - SR	92.9%	80.6%
ESRGAN + SSD	SR - SR	85.8%	80.2%
	HR - SR	82.5%	78.9%
ESRGAN + FRCNN	SR - SR	92.5%	81.1%
	HR - SR	91.8%	79.3%
EEGAN + SSD	SR - SR	86.1%	79.1%
	HR - SR	83.3%	77.5%
EEGAN + FRCNN	SR - SR	92%	79.9%
	HR - SR	91.1%	77.9%

In table 4.2, the detection outputs of the different combinations of SR methods and detectors are shown with the different combinations of train/test pairs. As can be seen, our new EESRGAN architecture displayed the best results already getting close to the detection rates which could be observed when working with HR images only. However, after training EESRGAN can be directly applied to LR imagery where no HR data is available and still achieved very good results. Furthermore, we could observe that other SR methods EEGAN and ESRGAN have already improved the AP considerably when used for preprocessing of LR images. However, for both data sets, EESRGAN have outperformed the other two methods.

4.2.3.3 End-to-End Training with Super-Resolution

We trained our EESRGAN network and detectors end-to-end for this experiment. The discriminator (D_{Ra}), and the detectors jointly acted as a discriminator for the entire architecture. Detector loss was backpropagated to the SR network, and therefore, the loss contributed to the enhancement of LR images.

At training time, LR-HR image pairs were used to train the EEGAN part, and then the generated SR images were sent to the detector for training. At test time, only the LR images were fed to the network. Our architecture first generated a SR image of the LR input before object detection was performed.

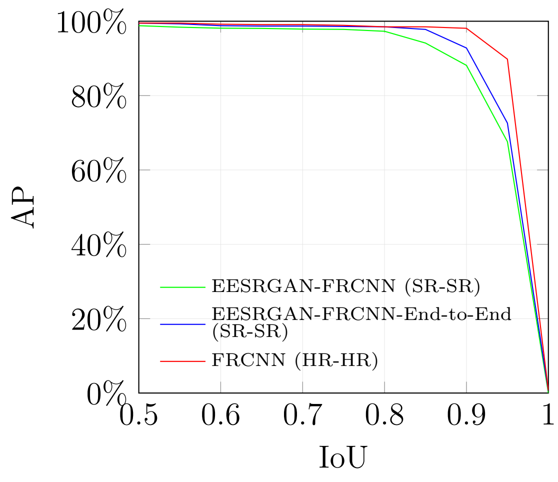
We also compared our results with different architectures. We used ESRGAN [73] and EEGAN [27] with the detectors for comparison. Table 4.3 clearly shows that our method delivers superior results compared to others.

Table 4.3: Detection with end-to-end SR (super-resolution) network. Detectors are trained with SR images and AP (average precision) values are calculated using 10 different IoUs (intersection over union).

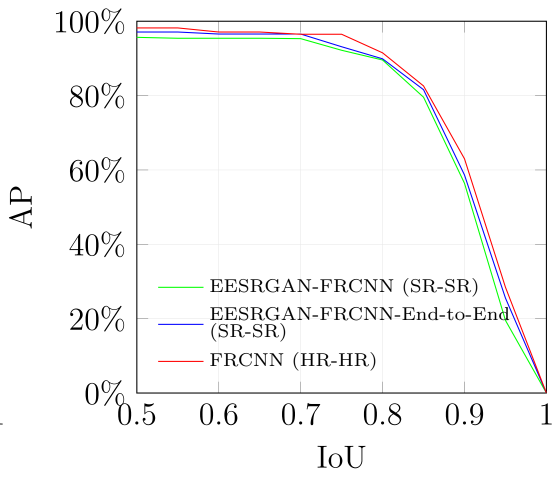
Model	**Training image resolution - Test image resolution**	**COWC Dataset (Test Results) (AP at IoU=0.5:0.95) (single class - 15cm)**	**OGST Dataset (Test Results) (AP at IoU=0.5:0.95) (single class - 30cm)**
EESRGAN + SSD	SR - SR	89.3%	81.8%
EESRGAN + FRCNN	SR - SR	**95.5%**	**83.2%**
ESRGAN + SSD	SR - SR	88.5%	81.1%
ESRGAN + FRCNN	SR - SR	93.6%	82%
EEGAN + SSD	SR - SR	88.1%	80.8%
EEGAN + FRCNN	SR - SR	93.1%	81.3%

4.2.3.4 AP versus IoU curve

We have calculated the AP values on different IoUs. In figure 4.7, we plot the AP versus IoU curves for our datasets. The performance of EESRGAN-FRCNN, end-to-end EESRGAN-FRCNN, and FRCNN is shown in the figure. The end-to-end EESRGAN-FRCNN network has performed better than the separately trained network. The difference is most evident for the higher IoUs on the COWC dataset. Our results indicate excellent performance compared to the highest possible AP values obtained from standalone FRCNN (trained and tested on HR images)



(a) COWC dataset



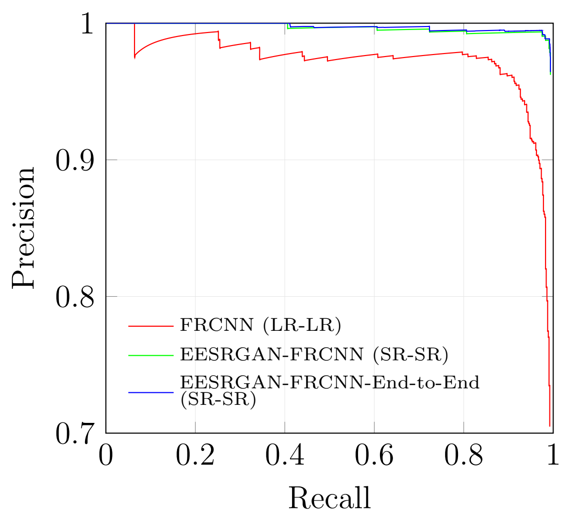
(b) OGST dataset

Figure 4.7: AP-IoU (average precision-intersection over union) curves for the datasets. Plotted results show the detection performance of standalone faster R-CNN on HR (high-resolution) images and our proposed method (with and without end-to-end training) on SR (super-resolution) images.

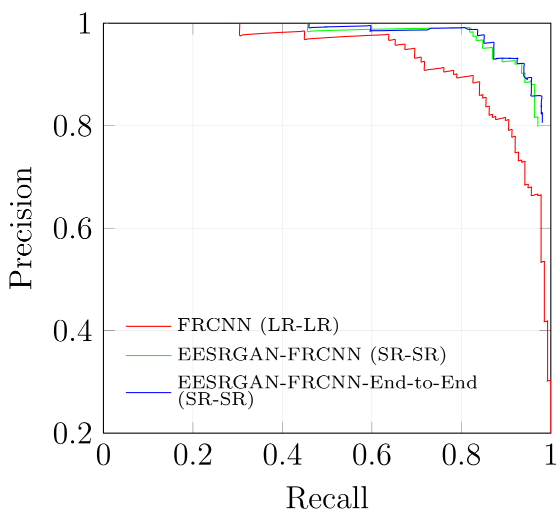
The OGST dataset has displayed less performance variation compared to the COWC dataset. The object size of the OGST dataset is larger than that of the COWC dataset. Therefore, the performance difference was not similar to the COWC dataset when we compared between standalone FRCNN and our method on the OGST dataset. To conclude, training our new architecture in an end-to-end manner has displayed an improvement for both the datasets.

4.2.3.5 Precision versus Recall

In figure 4.8, precision-recall curves are shown for both of our datasets. The precision-recall curve for COWC dataset is depicted in 4.8a and 4.8b represents the curve for OGST dataset. For each dataset, we plot the curves for standalone faster R-CNN with LR training/testing images, and our method with/without end-to-end training. We used IoU=0.5 to calculate precision and recall.



(a) COWC dataset



(b) OGST dataset

Figure 4.8: Precision-recall curve for the datasets. Plotted results show the detection performance of standalone faster R-CNN on LR (low-resolution) images and our proposed method (with and without end-to-end training) on SR (super-resolution) images.

The precision-recall curves for both datasets show that our method has higher precision values in higher recall values compared to the standalone faster R-CNN models. Our models with end-to-end training performed better than our models without the end-to-end training. In particular, the end-to-end models have detected more than 99% of the cars with 96% AP in the COWC dataset. For the OGST dataset, our end-to-end models have detected more than 81% of the cars with 97% AP.

4.2.3.6 Effects of Dataset Size

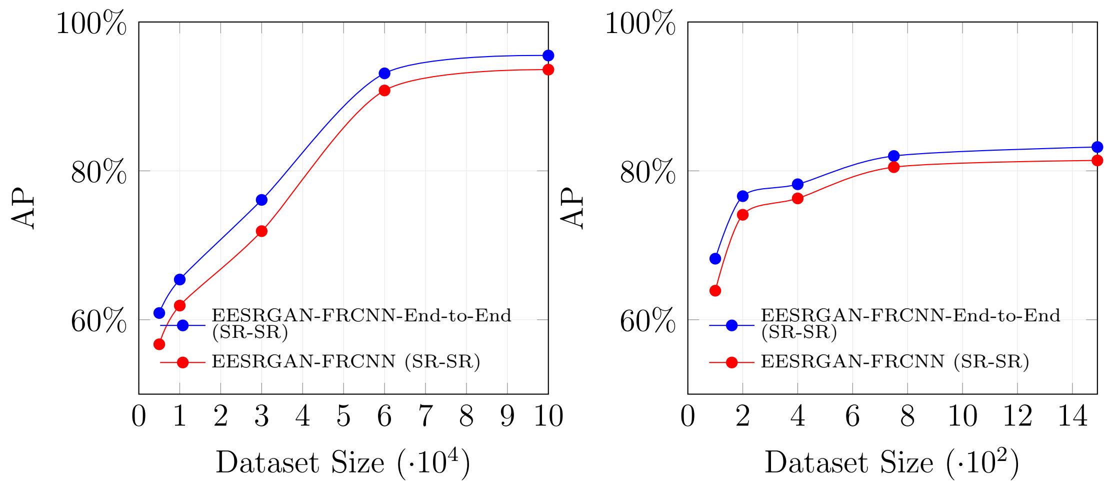
We trained our architecture with different training set sizes and tested with a fixed test set. In figure 4.9, we plot the AP values (IoU=0.5:0.95) against different numbers of labeled objects for both of our datasets (training data). We used five different dataset sizes: $\{500, 1000, 3000, 6000, 10000(cars)\}$ and $\{100, 200, 400, 750, 1491(tanks)\}$ to train our model with and without the end-to-end setting.

We got the highest AP value of 95.5% with our full COWC training dataset (10000 cars), and we used the same test dataset (1000 cars) for all combinations of the training dataset (with end-to-end setting). We also used another set of 1000 labeled cars for validation. Using 6000 cars, we got an AP value near to the highest AP, as shown with the plot of AP versus dataset size (COWC). The AP value decreased significantly when we used only 3000 labeled cars as training data. We got the lowest AP using only 500 labeled cars, and the trend of AP was further decreasing as depicted in figure 4.9a. Therefore, we can infer that we needed around 6000 labeled cars to get precision higher than 90% for the COWC dataset. We observed slightly lower AP values for all sizes of COWC datasets when we did not use the end-to-end setting, and we observed higher differences between the two settings (with and without end-to-end) when we used less than 6000 labeled cars.

The OGST dataset gave 83.2% AP (with end-to-end setting) using the full training dataset (1491 tanks), and we used 100 labeled tanks as test and same amount as validation data for all combinations of the training dataset. We got high AP values with 50% of our full training dataset as depicted in 4.9b. AP values dropped below 80% when we further decreased the training data. Similar to the COWC datasets, we also got comparatively lower AP values for all sizes of OGST datasets. We observed slightly higher differences between the two settings (with and without end-to-end) when the dataset consisted of less than 400 labeled tanks, as shown in the plot of AP versus dataset size (OGST dataset).

We used 90% of the OGST dataset for training while we used the 80% of the COWC dataset for the same purpose. The AP of the testing data (OGST) slightly increased when we added more training data, as depicted in figure 4.9b.

Therefore, we used a larger percentage of training data for the OGST dataset than for the COWC dataset, and it slightly helped to improve the relatively low AP of the OGST test data.



(a) COWC dataset

(b) OGST dataset

Figure 4.9: AP (average precision) with varying number of training sets from the datasets. Plotted results show the detection performance of our proposed method (with and without end-to-end training) on SR (super-resolution) images.

4.2.3.7 Enhancement and Detection

In figure 4.10, we have shown our input LR images, corresponding generated SR image, enhanced edge information and final detection. The image enhancement has helped the detectors to get high AP values and also makes the images visually good enough to identify the objects easily. It is evident from the figure that the visual quality of the generated SR images is quite good compared to the corresponding LR images, and the FRCNN detector has detected most of the objects correctly.

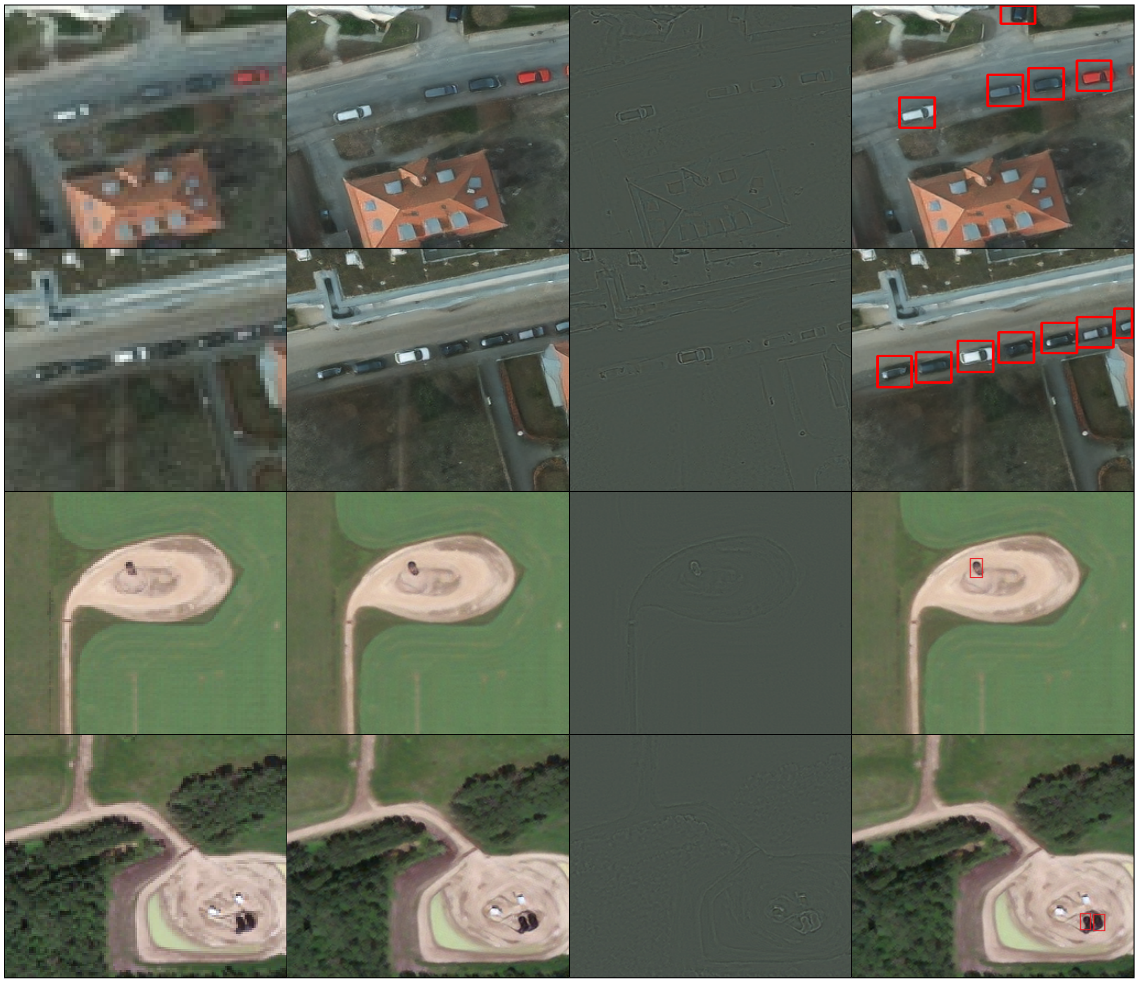
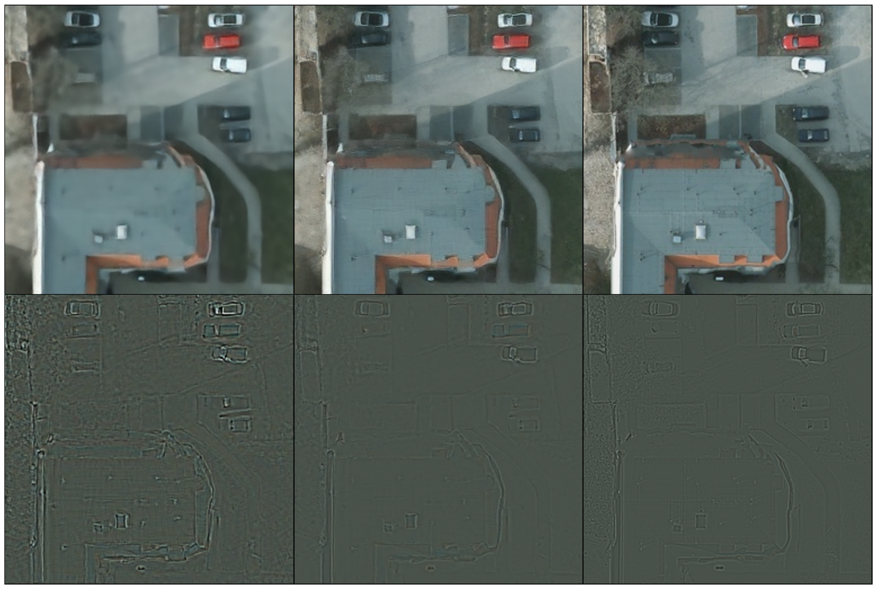

(a) Input LR image (b) Generated SR image (c) Enhanced edge (d) Detection

Figure 4.10: Examples of SR (super-resolution) images that are generated from input LR (low-resolution) images are shown in (a) and (b). The enhanced edges and detection results are shown in (c) and (d).

4.2.3.8 Effects of Edge Consistency Loss (L_{edge_cst})

In EEGAN [27], only image consistency loss (L_{img_cst}) was used for enhancing the edge information. This loss generated edge information with noise, and as a result, the final SR images became blurry. The blurry output with noisy edge using only L_{img_cst} loss is shown in figure 4.11a. The blurry final images gave lower detection accuracy compared to sharp outputs.

Therefore, we have introduced edge consistency loss (L_{edge_cst}) in addition to L_{img_cst} loss that gives noise-free enhanced edge information similar to the edge extracted from ground truth images and the effects of the L_{edge_cst} loss is shown in figure 4.11b. The ground truth HR image with extracted edge is depicted in figure 4.11c.


(a) Final SR image and enhanced edge with L_{img_cst} loss

(b) Final SR image and enhanced edge with L_{img_cst} and L_{edge_cst} losses

(c) Ground truth HR image with extracted edge

Figure 4.11: Effects of edge consistency loss (L_{edge_cst}) on final SR (super-resolution) images and enhanced edges compared to the extracted edges from HR (high-resolution) images.

4.3 Discussion

The detection results of our method presented in the previous section have indicated that our end-to-end SR-detector network improved detection accuracy compared to several other methods. Our method outperformed the standalone state-of-the-art methods such as SSD or faster R-CNN when implemented in low-resolution remote sensing imagery. We used EESRGAN, EEGAN, and ESRGAN as the SR network with the detectors. We showed that our EESRGAN with the detectors performed better than the other methods and the edge-enhancement helped to improve the detection accuracy. The AP improvement was higher in high IoUs and not so much in the lower IoUs. We have also showed that the precision increased with higher resolution. The improvement of AP values for the OGST dataset was lower than that for the COWC dataset because the area covered by a tank was slightly bigger than that of a car, and tanks sizes and colors were less diverse than the cars.

Our experimental results indicated that AP values of the output could be improved slightly with the increase of training data. The results also demonstrated that we could use less training data for both the datasets to get a similar level of accuracy that we obtained from our total training data.

The faster R-CNN detector gave us the best result, but it took a longer time than an SSD detector. If we need detection results from a vast area, then SSD would be the right choice sacrificing some amount of accuracy.

We had large numbers of cars from different regions in the COWC dataset, and we obtained high AP values using different IoUs. On the other hand, the OGST dataset needed more data to get a general detection result because we used data from a specific area and for a specific season and this was one of the limitations of our experiment. Most likely, more data from different regions and seasons would make our method more robust for the use-case of oil and gas storage tank detection. Another limitation of our experiment was that we showed the performance of the datasets that contain only one class with less variation. We would be looking forward to exploring the performance of our method on a broader range of object types and landscapes from different satellite datasets.

We have used LR-HR image pairs to train our architecture, and the LR images were generated artificially from the HR counterparts. To our knowledge, there is no suitable public satellite dataset that contains both real HR and real LR image pairs and ground truth bounding boxes for detecting small objects. Therefore, we have created the LR images which do not precisely correspond to true LR images. However, improvement of resolution through deep learning always improved object detection performance on remote sensing images (for both artificial and real low-resolution images), as discussed in the introduction and related works section of this thesis [64]. Impressive works [7], [84] exist in literature to create realistic LR images from HR images. For future work, we are looking forward to exploring the works to create more accurate LR images for training.

Chapter 5

Conclusion and Future Work

In this thesis, we propose an end-to-end architecture that takes LR satellite imagery as input and gives object detection results as outputs. We also create a tool that makes it easy to label, train, and detect objects from remote sensing images that cover a large area. Our proposed architecture contains a SR network and a detector network. We have used a different combination of SR systems and detectors to compare the AP values for detection using two different datasets. Our experimental results show that the proposed SR network with faster R-CNN has yielded the best results for small objects on satellite imagery. However, we need to add more diverse training data in the OGST dataset to make our model robust in detecting oil and gas storage tanks. We also need to explore diverse datasets and the techniques to create more realistic LR images.

Our proposed method solved some problems related to small-object detection. Large satellite imagery has another notable problem in the case of object detection: many image regions without any object of interest. Therefore, we get many image tiles generated from a large image without any object for detection. Consequently, our model might generate false detections for those tiles, and total inference time increases. For future work, we are looking forward to implementing a tiny model incorporated with our end-to-end architecture for filtering out the undesired image tiles during inference time. We need both types of tiles (with and without an object of interest) for the training process. It might increase the precision and decrease the total inference time for large satellite images.

In conclusion, our method has combined different strategies to provide a better solution to the task of small-object detection on LR imagery.

References

[1] *Alberta energy regulator*, `https://www.aer.ca`, Accessed: 2020-02-05.

[2] *Alberta geological survey*, `https://ags.aer.ca`, Accessed: 2020-02-05.

[3] N. Ammour, H. Alhichri, Y. Bazi, B. Benjdira, N. Alajlan, and M. Zuair, "Deep learning approach for car detection in uav imagery," *Remote Sensing*, vol. 9, no. 4, p. 312, 2017.

[4] *Arcgis online*, `https://www.arcgis.com/index.html`, Accessed: 2020-03-12.

[5] Y. Bai, Y. Zhang, M. Ding, and B. Ghanem, "Sod-mtgan: Small object detection via multi-task generative adversarial network," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 206–221.

[6] *Bing map*, `https://www.bing.com/maps`, Accessed: 2020-02-05.

[7] A. Bulat, J. Yang, and G. Tzimiropoulos, "To learn image super-resolution, use a gan to learn how to do image degradation first," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 185–200.

[8] P. Charbonnier, L. Blanc-Féraud, G. Aubert, and M. Barlaud, "Two deterministic half-quadratic regularization algorithms for computed imaging," *Proceedings of 1st International Conference on Image Processing*, vol. 2, 168–172 vol.2, 1994.

[9] Z. Chen, T. Zhang, and C. Ouyang, "End-to-end airplane detection using transfer learning in remote sensing images," *Remote Sensing*, vol. 10, no. 1, p. 139, 2018.

[10] S. Chowdhury, D. K. Chao, T. C. Shipman, and M. A. Wulder, "Utilization of landsat data to quantify land-use and land-cover changes related to oil and gas activities in west-central alberta from 2005 to 2013," *GIScience & Remote Sensing*, vol. 54, no. 5, pp. 700–720, 2017.

[11] I. Colomina and P. Molina, "Unmanned aerial systems for photogrammetry and remote sensing: A review," *ISPRS Journal of photogrammetry and remote sensing*, vol. 92, pp. 79–97, 2014.

[12] *Computer vision annotation tool (cvat)*, `https://github.com/opencv/cvat`, Accessed: 2020-03-12.

[13] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, Feb. 2016, ISSN: 2160-9292. DOI: 10.1109/tpami.2015.2439281. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2015.2439281.

[14] *Envi*, https://www.harrisgeospatial.com/Software-Technology/ENVI, Accessed: 2020-03-12.

[15] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[16] M. Fromm, M. Schubert, G. Castilla, J. Linke, and G. McDermid, "Automated detection of conifer seedlings in drone imagery using convolutional neural networks," *Remote Sensing*, vol. 11, no. 21, p. 2585, 2019.

[17] R. Girshick, "Fast r-cnn," *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. DOI: 10.1109/iccv.2015.169. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.169.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014. DOI: 10.1109/cvpr.2014.81. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2014.81.

[19] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[20] M. Haris, G. Shakhnarovich, and N. Ukita, "Task-driven super resolution: Object detection in low-resolution images," *arXiv preprint arXiv:1803.11316*, 2018.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. DOI: 10.1109/iccv.2015.123. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.123.

[22] ——, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *2015 IEEE International Conference on Computer Vision (ICCV)*, Dec. 2015. DOI: 10.1109/iccv.2015.123. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2015.123.

[23] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and et al., "Speed/accuracy trade-offs for modern convolutional object detectors," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: 10.1109/cvpr.2017.351. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.351.

[24] Jakaria, *Label-detect. git code*, `https://github.com/Jakaria08/Label-Detect`, 2019.

[25] H. Ji, Z. Gao, T. Mei, and B. Ramesh, "Vehicle detection in remote sensing images leveraging on simultaneous super-resolution," *IEEE Geoscience and Remote Sensing Letters*, pp. 1–5, 2019, ISSN: 1558-0571. DOI: `10.1109/LGRS.2019.2930308`.

[26] J. Jiang, J. Ma, Z. Wang, C. Chen, and X. Liu, "Hyperspectral image classification in the presence of noisy labels," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 2, pp. 851–865, Feb. 2019, ISSN: 1558-0644. DOI: `10.1109/TGRS.2018.2861992`.

[27] K. Jiang, Z. Wang, P. Yi, G. Wang, T. Lu, and J. Jiang, "Edge-enhanced gan for remote sensing image superresolution," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5799–5812, Aug. 2019, ISSN: 1558-0644. DOI: `10.1109/TGRS.2019.2902431`.

[28] A. Jolicoeur-Martineau, *The relativistic discriminator: A key element missing from standard gan*, 2018. arXiv: `1807.00734 [cs.LG]`.

[29] B. Kamgar-Parsi, B. Kamgar-Parsi, and A. Rosenfeld, "Optimally isotropic laplacian operator," *IEEE Transactions on Image Processing*, vol. 8, no. 10, pp. 1467–1472, Oct. 1999, ISSN: 1941-0042. DOI: `10.1109/83.791975`.

[30] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. DOI: `10.1109/cvpr.2016.182`. [Online]. Available: `http://dx.doi.org/10.1109/CVPR.2016.182`.

[31] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[32] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[33] *Labelbox*, `https://labelbox.com`, Accessed: 2020-03-12.

[34] W.-S. Lai, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Deep laplacian pyramid networks for fast and accurate super-resolution," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: `10.1109/cvpr.2017.618`. [Online]. Available: `http://dx.doi.org/10.1109/CVPR.2017.618`.

[35] *Landsat 8*, `https://www.usgs.gov/land-resources/nli/landsat/landsat-8`, Accessed: 2020-02-11.

[36] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and et al., “Photo-realistic single image super-resolution using a generative adversarial network,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: 10.1109/cvpr.2017.19. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.19.

[37] S. Lei, Z. Shi, and Z. Zou, “Super-resolution for remote sensing images via local–global combined network,” *IEEE Geoscience and Remote Sensing Letters*, vol. 14, no. 8, pp. 1243–1247, 2017.

[38] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, “Object detection in optical remote sensing images: A survey and a new benchmark,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 296–307, 2020.

[39] Q. Li, L. Mou, Q. Xu, Y. Zhang, and X. X. Zhu, “R3-net: A deep network for multioriented vehicle detection in aerial images and videos,” *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 7, pp. 5028–5042, Jul. 2019, ISSN: 1558-0644. DOI: 10.1109/tgrs.2019.2895362. [Online]. Available: http://dx.doi.org/10.1109/TGRS.2019.2895362.

[40] W. Li, H. Fu, L. Yu, and A. Cracknell, “Deep learning based oil palm tree detection and counting for high-resolution remote sensing images,” *Remote Sensing*, vol. 9, no. 1, p. 22, 2017.

[41] Z. Li and F. Zhou, “Fssd: Feature fusion single shot multibox detector,” *arXiv preprint arXiv:1712.00960*, 2017.

[42] L. Liebel and M. Körner, “Single-image super resolution for multispectral remote sensing data using convolutional neural networks,” *ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 41, pp. 883–890, 2016.

[43] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, “Enhanced deep residual networks for single image super-resolution,” *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, Jul. 2017. DOI: 10.1109/cvprw.2017.151. [Online]. Available: http://dx.doi.org/10.1109/CVPRW.2017.151.

[44] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, “Feature pyramid networks for object detection,” *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017. DOI: 10.1109/cvpr.2017.106. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2017.106.

[45] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, “Focal loss for dense object detection,” *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017. DOI: 10.1109/iccv.2017.324. [Online]. Available: http://dx.doi.org/10.1109/ICCV.2017.324.

[46] 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*, Springer, 2014, pp. 740–755.

[47] H. Liu, Z. Fu, J. Han, L. Shao, and H. Liu, "Single satellite imagery simultaneous super-resolution and colorization using multi-task deep neural networks," *Journal of Visual Communication and Image Representation*, vol. 53, pp. 20–30, 2018.

[48] S. Liu, D. Huang, *et al.*, "Receptive field block net for accurate and fast object detection," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 385–400.

[49] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," *Lecture Notes in Computer Science*, pp. 21–37, 2016, ISSN: 1611-3349. DOI: `10.1007/978-3-319-46448-0_2`. [Online]. Available: `http://dx.doi.org/10.1007/978-3-319-46448-0_2`.

[50] W. Ma, Z. Pan, J. Guo, and B. Lei, "Super-resolution of remote sensing images based on transferred generative adversarial network," in *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, IEEE, 2018, pp. 1148–1151.

[51] Q. Mao, S. Wang, S. Wang, X. Zhang, and S. Ma, "Enhanced image decoding via edge-preserving generative adversarial networks," in *2018 IEEE International Conference on Multimedia and Expo (ICME)*, IEEE, 2018, pp. 1–6.

[52] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, "A large contextual dataset for classification, detection and counting of cars with deep learning," in *European Conference on Computer Vision*, Springer, 2016, pp. 785–800.

[53] A. O. Ok and E. Başeski, "Circular oil tank detection from panchromatic satellite images: A new automated approach," *IEEE Geoscience and Remote Sensing Letters*, vol. 12, no. 6, pp. 1347–1351, 2015.

[54] J. Pang, C. Li, J. Shi, Z. Xu, and H. Feng, "$\mathcal{R}^2$ -cnn: Fast tiny object detection in large-scale remote sensing images," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 57, no. 8, pp. 5512–5524, Aug. 2019, ISSN: 1558-0644. DOI: `10.1109/TGRS.2019.2899955`.

[55] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: `http://papers.neurips.cc/paper/9015-`

pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[56] J. Rabbi, *Edge Enhanced GAN with Faster RCNN for end-to-end object detection from remote sensing imagery*, https://github.com/Jakaria08/Filter_Enhance_Detect, 2020.

[57] J. Rabbi, S. Chowdhury, and D. Chao, *Oil and gas tank dataset*, Mendeley Data, V3, 2020. DOI: 10.17632/bkxj8z84m9.3.

[58] M. Radovic, O. Adarkwa, and Q. Wang, "Object recognition in aerial images using convolutional neural networks," *Journal of Imaging*, vol. 3, no. 2, p. 21, 2017.

[59] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016. DOI: 10.1109/cvpr.2016.91. [Online]. Available: http://dx.doi.org/10.1109/CVPR.2016.91.

[60] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017, ISSN: 2160-9292. DOI: 10.1109/tpami.2016.2577031. [Online]. Available: http://dx.doi.org/10.1109/TPAMI.2016.2577031.

[61] Y. Ren, C. Zhu, and S. Xiao, "Small object detection in optical remote sensing images via modified faster r-cnn," *Applied Sciences*, vol. 8, no. 5, p. 813, 2018.

[62] 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*, 2014. arXiv: 1409.0575 [cs.CV].

[63] *Sentinel-2*, http://www.esa.int/Applications/Observing_the_Earth/Copernicus/Sentinel-2, Accessed: 2020-02-11.

[64] J. Shermeyer and A. Van Etten, "The effects of super-resolution on object detection performance in satellite imagery," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–10.

[65] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. arXiv: 1409.1556 [cs.CV].

[66] K. Stankov and D.-C. He, "Detection of buildings in multispectral very high spatial resolution images using the percentage occupancy hit-or-miss transform," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 7, no. 10, pp. 4069–4080, 2014.

[67] Y. Tai, J. Yang, X. Liu, and C. Xu, "Memnet: A persistent memory network for image restoration," *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct. 2017. DOI: `10.1109/iccv.2017.486`. [Online]. Available: `http://dx.doi.org/10.1109/ICCV.2017.486`.

[68] T. Tang, S. Zhou, Z. Deng, H. Zou, and L. Lei, "Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining," *Sensors*, vol. 17, no. 2, p. 336, 2017.

[69] H. Tayara and K. Chong, "Object detection in very high-resolution aerial images using one-stage densely connected feature pyramid network," *Sensors*, vol. 18, no. 10, p. 3341, 2018.

[70] H. Tayara, K. G. Soo, and K. T. Chong, "Vehicle detection and counting in high-resolution aerial images using convolutional regression neural network," *IEEE Access*, vol. 6, pp. 2220–2230, 2017.

[71] F. Tong, H. Tong, J. Jiang, and Y. Zhang, "Multiscale union regions adaptive sparse representation for hyperspectral image classification," *Remote Sensing*, vol. 9, no. 9, p. 872, 2017.

[72] Tzutalin, *Labelimg. git code*, `https://github.com/tzutalin/labelImg`, 2015.

[73] X. Wang, K. Yu, S. Wu, J. Gu, Y. Liu, C. Dong, Y. Qiao, and C. C. Loy, "Esrgan: Enhanced super-resolution generative adversarial networks," *Computer Vision – ECCV 2018 Workshops*, pp. 63–79, 2019, ISSN: 1611-3349. DOI: `10.1007/978-3-030-11021-5_5`. [Online]. Available: `http://dx.doi.org/10.1007/978-3-030-11021-5_5`.

[74] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[75] W. Yang, J. Feng, J. Yang, F. Zhao, J. Liu, Z. Guo, and S. Yan, "Deep edge guided recurrent residual learning for image super-resolution," *IEEE Transactions on Image Processing*, vol. 26, no. 12, pp. 5895–5907, Dec. 2017, ISSN: 1941-0042. DOI: `10.1109/tip.2017.2750403`. [Online]. Available: `http://dx.doi.org/10.1109/TIP.2017.2750403`.

[76] X. Yang, H. Sun, K. Fu, J. Yang, X. Sun, M. Yan, and Z. Guo, "Automatic ship detection in remote sensing images from google earth of complex scenes based on multiscale rotation dense feature pyramid networks," *Remote Sensing*, vol. 10, no. 1, p. 132, 2018.

[77] X. Yu and Z. Shi, "Vehicle detection in remote sensing imagery based on salient information and local shape feature," *Optik-International Journal for Light and Electron Optics*, vol. 126, no. 20, pp. 2485–2490, 2015.

[78] Y. Yuan, X. Zheng, and X. Lu, "Hyperspectral image superresolution by transfer learning," *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, vol. 10, no. 5, pp. 1963–1974, 2017.

[79] C. Zhan, X. Duan, S. Xu, Z. Song, and M. Luo, "An improved moving object detection algorithm based on frame difference and edge detection," in *Fourth International Conference on Image and Graphics (ICIG 2007)*, IEEE, 2007, pp. 519–523.

[80] F. Zhang, B. Du, L. Zhang, and M. Xu, "Weakly supervised learning based on coupled convolutional neural networks for aircraft detection," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 54, no. 9, pp. 5553–5563, 2016.

[81] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-shot refinement neural network for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4203–4212.

[82] Y. Zhang, Y. Tian, Y. Kong, B. Zhong, and Y. Fu, "Residual dense network for image super-resolution," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018. DOI: `10.1109/cvpr.2018.00262`. [Online]. Available: `http://dx.doi.org/10.1109/CVPR.2018.00262`.

[83] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019.

[84] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2223–2232.

[85] R. Zhu, S. Zhang, X. Wang, L. Wen, H. Shi, L. Bo, and T. Mei, "Scratchdet: Training single-shot object detectors from scratch," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2019, pp. 2268–2277.