TU WIEN Informatics

Super-Resolution von Luftbildern mit Fahrzeug-Fokus

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

Patrick Wahrmann, BSc.
Matrikelnummer 01327120

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Mitwirkung: Univ.Ass. Dipl.-Ing. Dr.techn. Sebastian Zambanini

Wien, 7. Oktober 2021

Patrick Wahrmann	Robert Sablatnig

TU WIEN Informatics

Vehicle-Focused Super-Resolution of Remote Sensing Imagery

DIPLOMA THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Visual Computing

by

Patrick Wahrmann, BSc.
Registration Number 01327120

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Robert Sablatnig
Assistance: Univ.Ass. Dipl.-Ing. Dr.techn. Sebastian Zambanini

Vienna, 7th October, 2021

Patrick Wahrmann	Robert Sablatnig

Technische Universität Wien
A-1040 Wien ▪ Karlsplatz 13 ▪ Tel. +43-1-58801-0 ▪ www.tuwien.at

Erklärung zur Verfassung der Arbeit

Patrick Wahrmann, BSc.

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

Wien, 7. Oktober 2021

Patrick Wahrmann

Danksagung

Es gibt wirklich viele Personen, ohne die diese Arbeit nie entstanden wäre. Neben meinen großartigen und immer konstruktiven Betreuern Sebastian Zambanini und Robert Sablatnig möchte ich auch meiner gesamten Familie danken, ohne deren Rückhalt und bedingungslose Unterstützung ich nie so weit gekommen wäre. Die Berechnungen in dieser Arbeit wurden durch die Benutzung des Vienna Scientific Cluster (VSC) und deren sehr hilfsbereite Support-Mitarbeiter ermöglicht. Weiters gebührt mein Dank Anna, Bernadette, Bernhard, Flo, Iris, Johannes, Kamilla, Lydia, Markus, Miranda, Saman und Verena für die gemeinsamen Schreib-Sessions, Diskussionen und sonstige Unterstützung im Rahmen der Erstellung dieser Arbeit. Ich habe darüber nachgedacht, die vorangegangene Liste durch eine kürzere Floskel („allen, die mich auf diesem Weg begleitet haben“ o. Ä.) zu ersetzen und keine Namen aufzuzählen, damit ich niemanden vergessen kann. Stattdessen habe ich mich dazu entschieden, mich hier in aller Förmlichkeit prophylaktisch zu entschuldigen und die vergessene(n) Person(en) auf ein Abendessen einzuladen. Tut mir leid für alle, die jetzt erwähnt wurden und um ein Abendessen umfallen! Zum Abschluss möchte ich zwecks der Förderung des Humors und der Selbstreflexion in der Wissenschaft auf das Paper „YOLOv3: An Incremental Improvement“ [RF18] verweisen.

Acknowledgements

There are so many people without whom this thesis would never have been written. First and foremost, I would like to thank my excellent and always constructive advisors Sebastian Zambanini and Robert Sablatnig. I would also like to acknowledge the role that my entire family has played, as I would have never been able to get here without their help and unconditional support. The computational results presented in this thesis have been achieved using the Vienna Scientific Cluster (VSC) with the assistance of their incredibly helpful support team. Furthermore, I would like to thank Anna, Bernadette, Bernhard, Flo, Iris, Johannes, Kamilla, Lydia, Markus, Miranda, Saman, and Verena for the joint writing sessions and discussions, and for always standing behind me during the process of writing this thesis. I considered replacing the preceding list with a short phrase ("all who have accompanied me during this journey" or something similar) and leave out all names in order to prevent potentially forgetting someone. Instead, I decided that I will apologize proactively and take the forgotten person(s) out for dinner. Apologies to those who have been mentioned but would have rather received a dinner. Finally, in order to promote humor and self-reflection in science, I recommend reading the paper "YOLOv3: An Incremental Improvement" [RF18].

Kurzfassung

Fahrzeugerkennung auf Luftbildern findet Anwendung in Verkehrsüberwachung und -planung sowie bei Rettungseinsätzen im Katastrophenfall. Super-Resolution kann als Vorverarbeitungsschritt durch das Erhöhen der Auflösung die Fahrzeugdetektion deutlich verbessern. Diese Diplomarbeit stellt eine neue Art des Trainings von Super-Resolution vor: auf Fahrzeuge fokussiertes Super-Resolution-Training, bei welchem Bildausschnitte mit je einem Fahrzeug in der Mitte für das Training verwendet werden. Als Super-Resolution-Algorithmus wird ein Residual Dense Network eingesetzt und für die Fahrzeugerkennung wird auf Faster R-CNN zurückgegriffen. Im Rahmen dieser Diplomarbeit werden sechs Datensets mit annotierten Fahrzeugen kombiniert, um das auf Fahrzeuge fokussierte Datenset, ein konventionelles Datenset für Super-Resolution-Training und ein Datenset für die Fahrzeugerkennung zu erstellen. Zusätzlich wird ein siebtes Datenset zum Testen verwendet, um die Generalisierungsfähigkeit des Ansatzes zu testen. Der Effekt der vorgestellten Trainings-Methode für Super-Resolution auf die Erkennungsgenauigkeit der Fahrzeugerkennung wird durch den Vergleich mit einem identisch aufgebauten, aber konventionell trainierten Modell quantifiziert. Ausführliche Analysen zeigen bei schnellerem Training eine Erkennungsqualität auf Augenhöhe für die vorgestellte Trainings-Strategie.

Abstract

Vehicle detection in remote sensing imagery has various applications in traffic analysis, planning, and rescue operations after natural disasters. Using super-resolution as a pre-processing step to increase the spatial resolution of remote sensing imagery benefits vehicle detection performance. This thesis proposes a novel procedure to train the super-resolution step of this pipeline in a vehicle-focused manner, by cropping the training set to images centered around vehicles. The Residual Dense Network is selected as super-resolution architecture and Faster R-CNN is utilized for vehicle detection. Six existing annotated datasets are combined and unified to create the vehicle-focused crops, a conventional dataset for super-resolution training, and a dataset for vehicle detection training. Additionally, testing on a seventh, completely unseen dataset allows a generalization error to be estimated. The effect of this super-resolution training method on subsequent vehicle detection is quantified by training an identical super-resolution model on unfocused data for comparison. Extensive evaluation shows on par performance of the vehicle-focused approach, while allowing faster training.

Contents

CHAPTER 1

Introduction

Can vehicle detection in remote sensing imagery be improved by training a Super-Resolution (SR) model in a vehicle-focused manner? This is the main question this thesis aims to answer. A vehicle detection algorithm is trained to create axis-aligned bounding boxes, which enclose vehicles present in a given remote sensing image (see Figure 1.1c). Before that, a SR model is applied (Figure 1.1b) to increase the spatial resolution of the original low resolution remote sensing imagery (see Figure 1.1a). The SR model is trained in a vehicle-focused manner (i.e. the training data is cropped to vehicles) and compared with a conventionally trained model.

(a) Low Resolution (b) Super Resolution (c) Vehicle Detection

Figure 1.1: An example for applying SR to a remote sensing image (from the DLR Multi-class Vehicle Detection and Orientation in Aerial Imagery (DLR-MVDA) dataset [LM15]) and then performing vehicle detection on it.

1.1 Motivation

This thesis proposes a method to improve vehicle detection in remote sensing imagery. Since manual annotation of large areas (e.g. whole cities) by human experts is time-consuming, automatic approaches have the potential to speed up this process. There are a number of uses for fast and reliable vehicle detection as described in the following paragraph.

In order to avoid congestion caused by increasing traffic, effective traffic management is needed and could be enhanced by automatic vehicle detection in remote sensing images [ZZW+13]. Vehicle detection facilitates traffic monitoring [SSSB18, ZLP+20] by providing the current number of vehicles, as well as their position and potentially also their speed [LM15], enabling intelligent traffic guidance and efficient transportation planning. Leitloff et al. [LRK+14] demonstrate the practicability of collecting the necessary data by building a traffic monitoring system located in an aircraft. They place three cameras and computer hardware inside the aircraft to directly and independently provide real-time traffic information. Stationary infrastructure, as commonly used for traffic surveillance, could fail in extreme situations, such as mass events or disasters [LRK+14]. An efficient vehicle detection system integrated into an aerial vehicle can provide necessary information in such circumstances [LRK+14].

Zambanini et al. [ZLP+20] mention the usage of vehicle detection methods for urban planning. Apart from traffic flow management, the distribution of air and noise pollution are also relevant when planning cities and vehicle detection can provide an approximation for those distributions.

Gleason et al. [GNB+11] describe another use case of vehicle detection: monitoring of rural areas with oil and gas pipelines. Vehicles, specifically heavy digging equipment, can accidentally damage those pipelines and consequently pollute the environment around it. According to Gleason et al. [GNB+11], there are pilots employed for patrolling above pipelines to manually check for potential threats to the pipelines. They suggest this could be simplified by applying automatic vehicle detection.

Gąszczak et al. [GBH11] suggest the use of vehicle detection algorithms for search and rescue missions. They state that the applicability to large and difficult terrain makes such algorithms suitable to benefit rescue missions in remote areas or after natural disasters.

1.2 Research Question

The central research question of this thesis is: Does training a super resolution network only with images depicting vehicles improve SR-supported vehicle detection? Previous work shows the potential of SR algorithms to significantly improve the performance of vehicle detection methods [CJWL16, FMN19, SVE19]. The idea of this thesis is that this beneficial effect of SR could be increased if the SR is specialized in providing higher resolution images of vehicles. This is tried to be achieved by training the SR model with a

dataset consisting only of fixed-size images centered around a vehicle. By training in this vehicle-focused manner, the model could learn creating High Resolution (HR) versions of vehicles better (e.g. sharper), than if trained with unfocused data. Better reconstructions could enable more precise vehicle detections. The analysis of this potential enhancement is the topic of this thesis.

1.3 Methodological Approach

The research question is answered by training two different SR models: one in a vehicle-focused manner and one conventionally. They are applied to the remote sensing imagery, which is then used to train vehicle detection models.

In order to have enough data available to train and test the SR, as well as the vehicle detection models, six datasets with annotated vehicles are collected and unified. They are used directly to train a SR model (the Residual Dense Network (RDN) architecture [ZTK+18] is used for SR as it provides state-of-the-art performance and a public implementation including pre-trained weights exists). Additionally, a dataset is constructed from the same data, focused on vehicles by cropping a fixed-size image patch centered around every annotated vehicle. This dataset is used to train another SR model, which is identical to the first one, apart from the training data used.

The effect of those two SR models on vehicle detection has to be analyzed in order to answer the research question. Thus, they are used to create two SR versions (i.e. a vehicle-focused and an unfocused version) of a new dataset stemming from the test sets of the previously mentioned datasets. Those two versions of the dataset created by different SR models are then used to train two versions of a vehicle detection model (i.e. Faster Regions with CNN features (Faster R-CNN) [RHGS15] based on an implementation by Co-Supervisor Sebastian Zambanini [ZLP+20]). Consequently, one of the models is trained and tested on a dataset pre-processed by a vehicle-focused SR and one by an unfocused (conventional) SR.

1.4 Structure of the Work

The remaining thesis is organized as follows:

Chapter 2: State of the Art
The reader is provided with the necessary knowledge needed to understand the context of this work and of the models and techniques used. Related work from the fields of deep learning, SR, object detection, and vehicle detection, as well as direct precursors in the field of SR-assisted vehicle detection are presented and explained.

Chapter 3: Methodology
The applied strategy used to answer the research question is presented. This includes detailed descriptions of how the dataset is constructed, how SR is applied

to the datasets, how the RDN architecture [ZTK+18] is working, how the vehicle detection is designed, and how the Faster R-CNN model [RHGS15] generates its predictions.

Chapter 4: Experiments

The achieved results are thoroughly discussed, beginning with the analysis of the SR step and then followed by the vehicle detection outcomes.

Chapter 5: Conclusion & Future Work

The reader is provided with a concise summary of the gained knowledge and a list of potential future research to be conducted on the matter of (vehicle-focused) SR for vehicle detection.

CHAPTER 2

State of the Art

This master thesis alludes the topics of remote sensing, vehicle detection, and super resolution, and applies deep learning techniques. The following chapter aims at providing a summary of the current state of the art of the topics relevant to the thesis and presents the precursors which the work builds upon.

This chapter discusses all the techniques necessary to understand the description of the methodology following in the next chapter, and additionally provides the reader with the ability to see the thesis in its context.

2.1 Deep Learning

Deep learning (also known as deep neural networks) denotes a group of algorithms which aim to learn data representations in multiple layers [LBH15]. The term "deep" derives from there being several of those layers [Mor11]. Every layer calculates an output, determined by parameters called weights, which in turn is used as input for following layers [LBH15]. Layers consist of entities called neurons, which - defined by specific patterns - access and combine inputs of a set of neurons of the previous layer [GBC16]. The name neuron, as well as neuronal network, is inspired by the similar structure of biological neurons, which do also combine multiple inputs to create an output [GBC16]. A neuron in a deep learning architecture calculates its output by, as a first step, multiplying every input value with the corresponding weight value [Nie15]. Then, the results of those multiplications are added together and additionally the bias is added [Nie15]. The weights, as well as the bias, are determined during the training phase [Nie15]. The resulting value is then modified by an activation function, for example the Rectified Linear Unit (ReLU), which is defined by $f(x) = max(x, 0)$ [LBH15]. ReLU is one of the most used activation functions, because empirical experiments show that it delivers superior performance when compared to other activation functions [Nie15].

One type of deep neural networks is the Convolutional Neural Network (CNN), which has "been widely adopted by the computer-vision community" [LBH15] and is also used in this thesis. The distinguishing element of the CNN is the convolutional layer, which consists of filters with trainable filter kernels [AH17]. Every unit of the layer processes the input by applying a convolution and passing the result (called feature map) to the subsequent layer [AH17].

According to LeCun et al. [LBH15], the idea behind the convolutional layer is that the same motives or objects depicted in images can be positioned in every part of the image. Sharing weights to detect the same patterns in different positions of the image takes advantage of that. In convolutional layers this is achieved by filtering the whole image with filters, which have the same trainable weights (= filter kernel) for the whole image.

One possibility to reduce the dimensionality of feature maps is to add a pooling layer after a convolutional layer [AH17]. Pooling is a downsampling approach, working by grouping areas of a specific size (e.g. 2×2) in the feature map and extracting one value per area (max pooling: the highest value, average pooling: the average value) [AH17]. One possible effect of the spatial dimensionality reduction of pooling is shift-invariance (in classification tasks) [GWK+18].

2.2 Super-Resolution

SR is the process of generating a HR image from a Low Resolution (LR) observation [NM14]. SR can generally be considered to be an "ill-posed" [DLHT15] problem, since for every LR image there exist multiple HR images, which would lead to the same low-resolution image if down-sampled. Thus, there is no unique solution to the SR problem.

If the source for generating a higher resolution image is only one single raster image, this is called Single Image Super Resolution (SISR) [van06]. Another possibility is to use multiple images of the same scene as input for generating one image, which is called Multi-Image Super Resolution (MISR) [FRC13]. As only SISR is relevant for this thesis, MISR is not discussed here.

According to Nasrollahi and Moeslund [NM14], the first SR algorithm was proposed by Gerchberg [Ger74] in 1974. Nowadays, approaches applying deep learning methods appear to have a superior performance [FMN19], thus only such methods are discussed in the following sections.

The SR networks described in this thesis are grouped by the basic principle used to increase the spatial resolution. The three common upsampling ideas for this categorization are preceding bicubic upsampling, transposed convolution, and the sub-pixel convolution layer.

2.2.1 Preceding Bicubic Upsampling

The first (according to Shermeyer and Van Etten [SVE19]) deep learning-based approach is Super-Resolution Convolutional Neural Network (SRCNN) by Dong et al. [DLHT15]. It works by first scaling the image to the desired resolution using bicubic interpolation. Then, three convolutional layers are applied to increase the sharpness of the interpolated input image without having to change its resolution anymore. Dong et al. [DLHT15] base their network design on the idea of building something similar to earlier sparse-coding-based SR approaches (e.g. [YWHM10]). The first of the three layers (a 9×9 convolutional layer) is interpreted to extract features of the image. The second layer performs non-linear mapping, by recombining the features of the first layer with 1×1 convolutions. Those created features are then used by the third layer (filter size 5×5) for the final reconstruction of the SR image, by creating three (or only one for gray-scale images) channels.

Kim et al. [KLL16a] propose a model called Very Deep Super Resolution (VDSR), which improves on SRCNN [DLHT15]. They increase the depth of the network from 3 to 20 layers and solely use filters of size 3×3. The depth causes a large receptive field, providing the network with more context around pixels and thus significantly improving the performance. To enable convergence, they apply residual learning (i.e. instead of the HR image itself, the network learns the difference between the interpolated LR image and the HR image, by adding the interpolated input image to the output of the last layer) and a high (i.e. 10^4 times higher than SRCNN) learning rate. In order to avoid exploding gradients, adjustable gradient clipping is applied, which limits gradients to a specified range depending on the current learning rate. Additionally, VDSR is able to generate images with multiple scale factors (i.e. $\times 2$, $\times 3$, $\times 4$) with the same model, by training with datasets in those scales. This allows to use the model for scaling images into multiple sizes without the need for multiple models. It also improves performance for all scale factors, when compared to training on just one scale factor, "implying the redundancy among scale-specific models" [LSK+17].

In [KLL16b] the same authors propose another SR model: Deeply-Recursive Convolutional Network (DRCN). The main idea behind this approach is that for better results a deeper model (and thus a wider receptive field) would be beneficial. However, adding new layers increases the number of parameters, potentially leading to overfitting. To avoid this problem, the authors choose to add more layers without adding parameters by simply repeating a layer, thus making it recursive. The same layer (with identical weights) is applied multiple (i.e. up to 16) consecutive times. The authors report a high risk for exploding and vanishing gradients, against which they add supervised training for each iteration. In other words, the output of every iteration is (after an intermediate "reconstruction" layer) compared to the ground truth and used for training. For inference, the outputs of all iterations are averaged to create the final output. Furthermore, the training is facilitated by a skip connection from the input to the reconstruction layer.

Tai et al. [TYL17] also propose a SR network (called Deep Recursive Residual Network

(DRRN)) featuring recursive layers. It features residual blocks consisting of multiple recursive layers, which are applied multiple times sequentially in a residual manner, and always learn the difference to the input of the residual block. The network uses a global residual connection (i.e. learning the difference to the up-scaled LR image) and gradient clipping to achieve convergence. In DRRN, in contrast to DRCN [KLL16b] (in which a single layer is repeated recursively multiple layers), multiple layers are repeated in each one of several blocks.

2.2.2 Transposed Convolution

Transposed convolutions (also named deconvolutions [LHAY17]) are the inverse operation to convolutions and change the resolution of the input feature map by a factor equivalent to the stride of the transposed convolution (e.g. a convolution with stride 2 reduces the size of the feature map per dimension to the half, a transposed convolution with stride 2 doubles it) [DLT16]. Applying a 3×3 convolutional kernel to an image results in a pixel calculated by a weighted (with the weights given by the kernel) combination of the 9 pixels. Inversely, a transposed convolution takes one pixel in an input image and applies a 3×3 kernel to calculate 9 output pixels (which are possibly also influenced by neighboring pixels of the input image, depending on the stride).

This method of upsampling is used by Dong et al. [DLT16] for Fast Super-Resolution Convolutional Neural Network (FSRCNN). The transposed convolution is applied as a last step, after multiple convolutional layers for feature extraction and non-linear mapping. In comparison to achieving the higher resolution by upfront bicubic upsampling, Dong et al. [DLT16] argue that processing the image with convolutions while in the LR space saves a substantial amount of processing (e.g. for a $2\times$ scale factor, the feature maps which have to be processed are four times smaller). FSRCNN is shaped like an "hourglass" [DLT16], as the number of filter is reduced (the first layer has 56 filters, then there are only 12 in the middle) and then increased again (reaching 56 filters again directly before the transposed convolution). Dong et al. [DLT16] note that training multiple scale factors (i.e. $\times 2$, $\times 3$, $\times 4$) can be accelerated by reusing all convolutional layers from an already trained scale factor and only fine-tuning the transposed convolution layer.

In their Laplacian Pyramid Super-Resolution Network (LapSRN), Lai et al. [LHAY17] also use transposed convolution layers for upsampling. Their network is designed to progressively generate a SR image: For every increase in resolution (LR$\Rightarrow \times 2 \Rightarrow \times 4 \Rightarrow \times 8$), several convolutional layers extract and process features, then double (per axis) the resolution of the feature maps with a transposed convolution layer. In parallel to this feature extraction branch of the network, an image reconstruction branch also uses a transposed convolution layer to double the resolution of the input image. Then, the up-scaled features are processed by one more convolutional layer before being added to the up-scaled input image (resulting in the first (i.e. $\times 2$) SR output image). The up-scaled feature maps and the output image are the basis for the next part of the feature extraction (which is designed identically) and the image reconstruction branch, respectively. This progressive architecture leads to a process in which an image with an

even higher resolution is produced after each phase. Additionally, Lai et al. [LHAY17] criticize the commonly used l_2 loss and claim it generates blurry images. Instead they apply the Charbonnier loss function [CBFAB94], which they claim handles outliers better and thus improves the performance.

2.2.3 Sub-Pixel Convolution

For their Efficient Sub-Pixel Convolutional Neural Network (ESPCN), Shi et al. [SCH+16] propose a new technique to increase the resolution of an image or feature map: the sub-pixel convolution. The basic principle is that for SR with scale factor s (resulting in s^2n pixels, where n is the original number of pixels in the image) the elements from s^2 feature maps are rearranged into a single feature map with s^2 times the size of the original feature maps in a periodically alternating manner[1]. This mechanism is applied in the last layer in order to calculate all feature maps in the LR space and is therefore much more efficient (i.e. s^2n times more efficient).

Similarly to Lai et al. [LHAY17], for their LapSRN, Ledig et al. [LTH+17] claim that the l_2 loss (and the Peak Signal-to-Noise Ratio (PSNR) for evaluation) is not suited for creating detailed textures and perceptually pleasing images. Instead, they suggest using an adversarial loss and design a Generative Adversarial Network (GAN) [GPAM+14], which they call Super-Resolution Generative Adversarial Network (SRGAN) [LTH+17]. The principle behind GANs is that two separate models are trained: a generator model to generate something (e.g. a SR image) and a discriminator model to distinguish the generated (e.g. SR) from the ground truth (e.g. HR) [GPAM+14]. According to Ledig et al. [LTH+17], the output of the discriminator network (a scalar indicating how realistic the model thinks the generated image is) is better suited to train a SR model than a common l_2 loss. They combine this loss with a loss calculated from the difference between the feature map representations (of a pre-trained VGG-19 network) of the predicted SR and the ground truth HR image. The network itself uses multiple residual blocks, an additional skip connection, and two sub-pixel convolution layers (each with a $\times 2$ factor, for $\times 2$ SR). The generator network of SRGAN can also be trained in a traditional manner, without adversarial loss, and is then called Super-Resolution Residual Network (SRResNet) [LTH+17]. While SRResNet surpasses all previous approaches in terms of PSNR, SRGAN outperforms every other approach in the accompanying mean opinion score testing, in which 26 humans rate the produced images manually.

Lim et al. [LSK+17] construct the Enhanced Deep Super-Resolution (EDSR) network, intended to directly improve upon the architecture of SRResNet [LTH+17]. Following Nah et al. [NHKML17], they remove all batch normalization layers from the network, because they reduce flexibility by normalizing features. This removal additionally saves 40% of GPU memory, allowing for a larger model. To keep the larger model numerically stable, they add residual scaling as proposed by Szegedy et al. [SIVA17]. Those changes lead to a significant improvement over the original SRResNet by Ledig et al. [LTH+17].

[1] The sub-pixel convolution layer is explained in more detail in Section 3.2.1.

Wang et al. [WYW+18] propose an enhanced version of SRGAN, which they call Enhanced Super-Resolution Generative Adversarial Network (ESRGAN). Similar to the EDSR network by Lim et al. [LSK+17], batch normalization layers are removed and residual scaling is added. They modify the residual blocks to become residual-in-residual dense blocks, containing additional skip connections inside of residual blocks. The discriminator is changed to be a relativistic average GAN, meaning that instead of producing a number estimating if an image is the ground truth or the SR version, it compares both and rates which one is more likely to be the ground truth. The VGG part of the loss is improved by comparing the feature maps before applying the activation function. It is empirically shown by Wang et al. [WYW+18] that initializing the weights with smaller values leads to easier training. According to Wang et al. [WYW+18], these changes add up to higher quality textures, better accuracy at brightness restoration, and in general perceptual improvements, when compared to SRGAN [LTH+17].

Zhang et al. [ZTK+18] propose RDN, a SR network built around so-called Residual Dense Blocks (RDBs). These residual (i.e. their output is added to their input) blocks consist of multiple convolutional layers, where the central concept is that each layer accesses the output of every previous convolutional layer in the same block, as well as the input to the block. This is also valid for the last convolutional layer in each block, which has a special role. In order to fully use all information generated by the layers of this block, it takes their output and reduces the dimensionality to the same number of channels as the input to the block has, by applying 1×1 convolutions. This match in dimensionality allows the output to be combined with the input by addition (for being residual). Similarly, the output of every RDB is fused with a 1×1 convolutional layer after the last RDB. Finally, RDN also uses a long skip connection from the beginning of the network to directly before the up-scaling module, which is a sub-pixel convolution layer. In this thesis, this model is used for generating SR remote sensing imagery as input to the vehicle detection network and is thus explained in more detail in Section 3.2.1.

The Residual Channel Attention Network (RCAN) by Zhang et al. [ZLL+18] contains (as RDN [ZTK+18]) two types of residual learning: long (i.e. after the first convolutional layer to the sub-pixel convolution layer in the end) and short (i.e. within residual groups) skip connections. The other main concept of RCAN is the channel attention mechanism. It works by reducing the size of every channel to 1×1 by using global average pooling, followed by two more convolutional layers. The resulting vector (called channel statistics) is then used to scale the original feature maps by multiplying them by it. This mechanism allows the network to focus on more informative features by scaling according to the channel statistics.

Dai et al. [DCZ+19] improve the channel attention mechanism in the scope of their Second-order Attention Network (SAN). In order to focus on features which are more informative, they replace the global average pooling mechanism by Zhang et al. [ZLL+18]. Instead, they calculate a normalized covariance matrix of the feature maps and calculate a mean per channel of that matrix. After further convolutional processing, this information is used for scaling the features. The second main contribution of Dai et al. [DCZ+19]

are region-level non-local modules. Non-local operations, as originally proposed by Wang et al. [WGGH18], let the whole image (or previous feature map) influence the outcome of every element of the next layer, similar to a fully connected layer. Differently to a fully-connected layer, the influence of an element is weighted by a predefined function (e.g. the Gaussian function) based on the values in the filter (e.g. a similar element could be weighted higher). Dai et al. [DCZ+19] modify this approach to be regional by splitting the feature map into a grid of regions. The non-local operation is only applied per region, which reduces computational complexity. This leads to superior results when compared to earlier papers.

2.2.4 Summary of Super-Resolution

The above excerpt of a range of SR approaches can be summarized by a few observations. Firstly, the majority of recent approaches use the sub-pixel convolution technique to increase the spatial resolution. Secondly, residual learning with skip connections is a frequently (e.g. [KLL16a, TYL17, LTH+17, LSK+17, ZTK+18]) applied technique to enable learning with more layers and thus increasing the capacity of the model. Additional improvements proposed include attention-based approaches [ZLL+18, DCZ+19], densely connected layers [ZTK+18], recursive layers [KLL16b, TYL17], and using GANs [LTH+17, WYW+18] or other improvements [LHAY17] for a better loss function. For a more elaborate insight in the state of the art of SR, refer to reviews compiled by Li et al. [LWZ+19], Wang et al. [WCH20], and Anwar et al. [AKB20].

2.3 Object Detection

According to Liu et al. [LOW+20], object detection is the task of determining "whether there are any instances of objects from given categories (such as humans, cars, bicycles, dogs or cats) in an image and, if present, to return the spatial location and extent of each object instance". In the scope of this thesis, the only relevant class is *vehicle*, as object detection is used in the context of vehicle detection.

This section aims to give an overview of the most common and influential papers about object detection. As the field of object detection is very active (Liu et al. [LOW+20] describe the quantity of papers as "breathtaking"), only a comparably small selection of deep learning-based methods are considered. More extensive insight can be found in object detection surveys (e.g. [ZZXW19, JZL+19, LOW+20, WSH20]).

Following Jiao et al. [JZL+19] and Liu et al. [LOW+20], in this thesis object detection approaches are grouped in two categories. The first category comprises two-stage approaches, where a first stage proposes a set of potential objects and a second stage classifies and refines those proposals. The other category, called one-stage approaches, unites those phases to a single stage and eliminates the need for a set of object proposals.

2.3.1 Two-Stage Approaches

Two-stage approaches for object detection, as "popularized" [LGG+17] by Regions with CNN features (R-CNN) [GDDM14], separate the process into two stages. The first stage involves generating proposals for objects, whereas in the second stage the proposals are selected, classified, and refined. R-CNN by Girshick et al. [GDDM14] utilizes the Selective Search algorithm, proposed by Uijlings et al. [UVDSGS13], as a proposal generation method. Selective Search works by iteratively joining regions in a segmentation, based on multiple measures of similarity. During this iterative process, around 2 000 bounding boxes (also named region proposals) of potential objects are generated around segmented regions. R-CNN transforms those proposals into images of size 227×227 pixels and feeds them into a convolutional neural network based on AlexNet [KSH12], which generates a 4 096-dimensional feature vector per proposal. For every targeted object class, one Support Vector Machine (SVM) [CV95] generates a score for the proposal based on the feature vector. Using the score, non-maximum suppression is applied to prevent multiple detection of an object. The bounding box coordinates are further enhanced by a linear regression model. As the first (according to Jiao et al. [JZL+19]) deep learning approach for object detection, they claim to surpass the state of the art by 30%.

GoogLeNet by Szegedy et al. [SLJ+15], although mainly focusing on object classification, is also applied to the task of object detection. It bases its principle on R-CNN [GDDM14] but uses a deeper, more complex model for feature generation and a combination of Selective Search [UVDSGS13] and Multi-Box [ESTA14] for improved proposal generation.

As suggested by its name, Fast Regions with CNN features (Fast R-CNN) (proposed by Girshick [Gir15]) is also a direct improvement on R-CNN [GDDM14]. One drawback of R-CNN is its speed, as every proposal is fed separately through the network to get the feature vector. Fast R-CNN (as well as Spatial Pyramid Pooling Network (SPP-Net) by He et al. [HZRS15]) improves on that by feeding the whole image only once through the network. For every object proposal bounding box, the part of the feature maps which corresponds to the bounding box is then extracted. As a result, the convolutional layers are only applied one time, instead of around 2 000 times. The variable-sized crop of the feature maps is processed using a Region of Interest (RoI) pooling layer, which results in a fixed-sized output by tiling the feature map crops into regular grids with a predefined size and applying max-pooling inside each grid cell. Based on this fixed-size feature vector, multiple fully connected layers produce probabilities for each class, as well as a bounding box regression for the object proposal. The fully connected layers replace the SVMs [CV95] of R-CNN, making the training process easier, as the SVMs have to be trained separately. The convolutional layers are pre-trained on the ImageNet classification dataset [DDS+09, RDS+15]. These improvements make training significantly faster (8.8× to 18.3× faster, depending on the selection of base model) and improve test speed even more (80× to 146×).

According to Ren et al. [RHGS15], the speed bottleneck of Fast R-CNN [Gir15] is the region proposal step, thus they concentrate on making that faster, which is why the

proposed algorithm is named Faster R-CNN. They replace Selective Search with their Region Proposal Network (RPN). This network shares the convolutional layers of the second part of the network (i.e. Fast R-CNN without Selective Search [UVDSGS13]) and uses the created features to generate bounding box proposals (consisting of coordinates and a value indicating the estimated likelihood of the proposal being an object) with three more fully connected layers. Anchor boxes are introduced to cope with multiple scales and ratios of objects. They work by providing the basis for a regression relative to the defined anchor scales and ratios. The resulting proposals are then used by a model resembling the one from Fast R-CNN, but sharing weights with the RPN, causing a significant speedup. The authors report a speed of 5 fps for Faster R-CNN. It should be noted that this algorithm is used for vehicle detection in this thesis and a more detailed explanation follows in Section 3.3.1.

Dai et al. [DLHS16] propose to make the whole object detection process fully convolutional, calling their approach Region-based Fully Convolutional Network (R-FCN). They build upon Faster R-CNN [RHGS15], but replace the RoI pooling layer with position-sensitive RoI pooling and the fully connected layers with an average voting of the pooled feature maps. The position-sensitive RoI pooling works by assigning a relative position of an object (i.e. top-left, top-center,...) to every feature map in the layer before the pooling. The feature maps are trained to respond to the respective parts of an object (e.g. one feature map responds to the top left part of objects of a certain category). The pooling takes these position-sensitive maps and calculates the average for a part (e.g. the top-left part of the area representing the RoI) of the respective feature map (i.e. the feature map responsible, for example, for the top-left parts of objects). These averages per part and class are combined to vote on a class probability. This process replaces fully connected layers of other approaches (e.g. Faster R-CNN [RHGS15]). Bounding box regression is done similarly, outputting bounding box coordinates instead of class probabilities. Dai et al. [DLHS16] claim to produce results comparable to Faster R-CNN, while being faster.

Lin et al. [LDG+17] propose Feature Pyramid Network (FPN) with the motivation of improving the detection of objects on multiple scales. Inspired by traditional (i.e. without deep learning) approaches, which apply algorithms on a pyramid of images scaled to multiple sizes in order to detect objects of all sizes, they build FPN. The idea is to exploit the different scales of feature maps when going deeper into a network (i.e. later layers have smaller feature maps, a larger receptive field and detect larger objects). Using feature maps at different depths would thus facilitate detection of objects of varying size. Problematically, the early feature maps in a network are not yet very expressive. Lin et al. [LDG+17] solve this problem by beginning with the deepest layer and then creating bigger (i.e. higher resolution) feature maps by upsampling and combining them with earlier feature maps. This is done iteratively to create multiple scales of feature maps, where each of them is used to either create an object proposal (if FPN is used as an enhancement of the RPN) or to classify and regress a proposal (if used as an enhancement for the second part of Faster R-CNN [RHGS15]). This explicit focus on scale invariance

improves on previous state of the art.

2.3.2 One-Stage Approaches

Redmon et al. [RDGF16] present their You Only Look Once (YOLO) network as an improvement over the two-stage principle of detection algorithms as Faster R-CNN [RHGS15]. Instead of having to rely on any form of object proposals of a first step, the network directly produces object predictions from an input image. Being based on a pre-trained GoogLeNet [SLJ+15], the model has 24 convolutional layers and two fully connected layers. The image is split into a 7×7 grid, where each grid cell produces two bounding boxes, but only one class prediction. Thus, the output of the network is a $7 \times 7 \times 30$ tensor (i.e. for each of the 7×7 grid cells there are scores for each of the 20 classes of PASCAL VOC [EVGW+10] and two bounding boxes with four coordinates and a confidence score each). The fully connected layers at the end of the network have a receptive field spanning the whole image, in contrast to approaches like R-CNN [GDDM14], Fast R-CNN [Gir15], and Faster R-CNN [RHGS15], where only an image crop defined by the object proposal is analyzed. This architecture can be trained end-to-end directly. During training, the predicted bounding box with the largest Intersection over Union (IoU) is assigned to be responsible for an object and is optimized to fit that ground truth bounding box. While producing fast and high quality results, YOLO has some inherent limitations. The authors report that the limit of two bounding boxes per grid cell leads to problems with many small objects (e.g. flocks of birds). Furthermore, localization of objects is not very accurate, leading to worse results than Faster R-CNN. On the other hand, the model is very fast (the fastest at the time of publication) and generalizes well to other domains (as demonstrated by Redmon et al. [RDGF16] on art datasets).

Redmon and Farhadi [RF17] propose YOLOv2, which they claim to be "better, faster, stronger" than the original YOLO [RDGF16] network. They achieve this by introducing a long list of improvements, including the following: incorporating batch normalization, pre-training the network on a higher resolution, adding anchor boxes inspired by Faster R-CNN [RHGS15] (but with anchor sizes and ratios picked by analyzing the dataset), overhauled bounding box parametrization, adding higher resolution features to the classification with a skip connection from an earlier layer, training on multiple image sizes, and a new network structure with 19 convolutional layers. In the same paper, Redmon and Farhadi [RF17] present YOLO9000, a semi-supervised modification of YOLOv2, which is capable of performing real-time object detection on 9 000 classes. Despite only having an object detection training set for a small fraction of the classes, they achieve this by combining it with the ImageNet classification dataset, which in turn has no localization data.

The next iteration of YOLO [RDGF16], called YOLOv3, contains only minor improvements according to their authors [RF18]. They do not use softmax in the last layer anymore and instead rely on multiple layers with different scales as direct input to the

bounding box prediction (inspired by FPN [LDG+17]), and add more layers and skip connections.

Similar to YOLO [RDGF16], the Single Shot Detector (SSD) network proposed by Liu et al. [LAE+16] generates bounding boxes and class scores directly from an input image. It does this without any fully connected layers and relies on convolutional layers for this to be achieved. Instead of separating an image or a feature map into a grid, the network predicts bounding boxes for each position in a feature map, by applying a convolutional layer to the feature map. These bounding boxes are parametrized relative to one of k (e.g. 4) default boxes (similar to the anchor boxes of Faster R-CNN [RHGS15]) per feature map element. The convolutional layers responsible for this step have k filters for each of the C classes and the four bounding box coordinate offsets, respectively, resulting in $(c+4)k$ filters, predicting k bounding boxes (including class scores) per feature map element. This is done on several consecutive feature maps of shrinking size to capture multiple scales of objects. Non-maximum suppression reduces the number of predicted objects. Because there is a large amount of predicted detections where no object is (as the number of predictions is fixed and only depends on the size of the feature maps), hard-negative mining is applied to avoid an imbalance in the loss. This is done by only considering the negative examples with the highest loss. The authors claim that this approach does not only produce better results than previous state of the art (e.g. YOLO, Faster R-CNN,...), but is also faster.

Lin et al. [LGG+17] observe that one-stage approaches fail to match the performance of two-stage approaches on the Common Objects in COntext (COCO) benchmark [LMB+14]. They identify class imbalance as a potential reason for this discrepancy and propose a new network called RetinaNet to demonstrate how they fix it. One-stage detectors usually have to evaluate a large number of potential objects (e.g. YOLO [RDGF16] uses several boxes for every single element in multiple feature maps), while two-stage approaches start with a small selection of object proposals pre-selected by a first stage. The vast majority of the potential bounding boxes do not match any existing object and thus provide no substantial value to learning. This imbalance is inefficient and according to Lin et al. [LGG+17] leads to "degenerate models". They propose a loss function, named focal loss, to alleviate this by weighting the contribution of a prediction to the loss by its difficulty. The difficulty is defined by the model's estimated probability for a class (or one minus the estimated probability, if the ground truth defines that there is no object of that class). The loss function thus weights samples higher that are farther away from the ground truth. The effectiveness of this modified loss function is demonstrated in experiments and using the RetinaNet, which exhibits better performance than "all existing state-of-the-art two-stage detectors" [LGG+17], despite being a one-stage approach. RetinaNet is based on FPN [LDG+17] and uses anchors as in FPN [LDG+17]. It features two sub-nets processing the multi-scale features: one for bounding box regression and one for classification.

2.3.3 Summary of Object Detection

A wide variety of object detection methods exists. The R-CNN [GDDM14] takes 13 seconds per image during testing. While this is fast when compared to earlier approaches (according to Girshick et al. [GDDM14]), later approaches did not only focus on improving performance, but also explicitly emphasize their achievements in increasing speed (e.g. [Gir15, RHGS15, DLHS16, RDGF16, RF17, LAE+16]). Apart from various improvements made to the architectures of the networks (e.g. making it deeper [SLJ+15, Gir15], sharing convolutional features by utilizing various techniques [Gir15, RHGS15], adding position sensitive layers [DLHS16], using multiple scales of feature maps for classification and regression [LAE+16, LDG+17, LGG+17],...), another main difference can be found in the underlying method of creating bounding boxes (e.g. by external approaches without deep learning [GDDM14, SLJ+15, Gir15], anchor-boxes [RHGS15, DLHS16, RF17, LAE+16], grid-based [RDGF16],...). A rule of thumb is that two-stage approaches tend to perform better, while one-stage approaches are faster and use fewer resources [LOW+20]. However, there are papers claiming to be the exception of the rule (e.g. [LGG+17]).

2.4 Vehicle Detection

Vehicle detection in the context of this thesis is the generation of bounding boxes that enclose vehicles[2] for given remote sensing imagery. It is difficult to define the edges of the term vehicle detection, as there are varying sub-tasks of different complexity as listed hereinafter. The resolution ranges from very high (e.g. [MA18]), as captured by low flying Unmanned Aerial Vehicles (UAVs), to very low (e.g. [ZJH19]), as captured by satellites. Additional modes of recording are capturing a highway from a bridge (e.g. [SWG+18]) or from a camera mounted on/in a car (e.g. [HXX+18]). The type of images ranges from gray-scale (e.g. [ZJH19]), color images (e.g. [MKSB16]), stereo images (e.g. [ZLP+20]), digital surface models (e.g. [WLH+20]), and infrared images (e.g. [LYL18]) to videos (e.g. [ZJH19]). Some approaches generate axis-aligned bounding boxes (e.g. [WLH+20]), while others produce oriented bounding boxes (e.g. [TZD+17a]), or limit themselves to generating fixed-sized square bounding boxes (e.g. [MKSB16]). All these scenarios are called vehicle detection, but pose completely different problems. The type of vehicle detection relevant to this thesis is the generation of axis-aligned bounding boxes of vehicles in remote sensing imagery with a Ground Sampling Distance (GSD) of around 0.3 m.

In general, this task is a special case of object detection, which is already thoroughly discussed in the previous section. This section provides a concise overview of how the approaches are modified to fit the task of vehicle detection.

Sommer et al. [SSB17] modify Fast R-CNN [Gir15] and Faster R-CNN [RHGS15] for vehicle detection. They analyze eight object proposal algorithms for their suitability

[2] In this thesis, only land-based multi-track vehicles like cars and trucks are considered. Ships, planes, bicycles, motorcycles, etc. are not in the scope.

for vehicle detection in conjunction with those two object detection methods. The best-performing proposal method in their experiments is RPN [RHGS15] (followed by Selective Search [UVDSGS13]), which is configured to use smaller anchor base sizes and produce a higher number of proposals, to suit the needs of vehicle detection. Additionally, they develop a new network used as backbone for Faster R-CNN, which consists of four convolutional layers and three fully connected layers (as well as ReLU, dropout, and pooling layers). With this smaller network, the resolution of the final feature maps is higher, allowing the detection of objects with a low spatial resolution (i.e. vehicles). Sommer et al. [SSB17] present experiments showing that Faster R-CNN with RPN, using their modified backbone, achieves the best results.

Carlet and Abayowa [CA17] adapt the YOLOv2 object detection network [RF17] for their vehicle detection study. They reason that while deeper convolutional neural networks are generally better, previous work [SKM17] shows that shallower networks are preferable for remote sensing imagery. Thus, they remove a few layers from YOLOv2, causing the resolution of the final feature maps to double. This allows the network to detect vehicles better because without this change the size of a vehicle in the final layer would be around one element. Furthermore, the larger anchors used in the original YOLOv2 are removed. The authors claim to have reached a near state of the art performance, but emphasize the speed of their network, as it is based on YOLOv2.

Tang et al. [TZD+17b] find two problems with using Faster R-CNN [RHGS15] for vehicle detection: problems with small vehicles and with complex backgrounds. The first one is alleviated by using features from the shallower layers of the RPN part of Faster R-CNN. The output of three different layers is concatenated and forms the features fed into the layers generating the object proposals. Those feature maps (from different layers) combine the advantages of different depths of layers and thus improve the performance on smaller vehicles. The same multi-layer feature maps, together with the proposals, are used as an input to the second part of Faster R-CNN. Tang et al. [TZD+17b] replace the fully connected layers of the original Faster R-CNN with a cascade of boosted decision trees. By training them with mined hard negatives (i.e. structures mistaken for vehicles), the second problem of Faster R-CNN (i.e. false positives on complex backgrounds) is mitigated.

Sommer et al. [SSSB18] also base their vehicle detection approach on Faster R-CNN [RHGS15]. They also observe that deeper layers do not have sufficient resolution to provide enough information for reliable detection of small vehicles. Their approach is to add transposed convolution layers before and after the last convolutional layer in order to increase the feature map resolution. This change is accompanied by two skip connections, concatenating higher resolution data from earlier layers to the output of the transposed convolution layers.

In conclusion, it can be said that most of the recent vehicle detection algorithms focus on adapting successful deep learning object detection models [ZLP+20]. One of the differences between general object detection and vehicle detection is the small size of the vehicles, which can cause problems (e.g. [TZD+17b]). On the other hand, the

approximate range of sizes of vehicles in a given dataset is known, allowing to focus the model on those sizes (e.g. by removing large anchor boxes [CA17]). Another problem of vehicle detection is overlapping bounding boxes (as for densely parked cars), which can be challenging to accurately detect [CA17].

2.5 Super-Resolution Vehicle Detection

There are studies applying SR to remote sensing imagery and investigating the effect on subsequent vehicle detection performance. They can be seen as precursors to this thesis and are summarized in this section.

Cao et al. [CJWL16] identify the low resolution of satellite images as a primary problem of vehicle detection. By applying SR as a preprocessing step, they are able to improve the vehicle detection performance. For that they use two conventional (i.e. no deep learning) approaches: coupled dictionary learning for SR and SVMs for vehicle detection.

Ferdous et al. [FMN19] apply SRGAN as SR algorithm and use SSD to generate object detections. They observe an increase of object detection performance when applying SR, especially with higher GSDs. In detail, performing SR with a scale ratio of $\times 2$ to a GSD of 0.25 m produces drastically better results (the mean Average Precision (mAP) is about 30 percent points higher) than from 1.0 m to 0.5 m. It is even remarkably better to apply $\times 4$ than $\times 2$ SR to LR imagery (i.e. 1.0 m $\Rightarrow$ 0.5 m GSD results in a 23 percent points lower mAP than 1.0 m $\Rightarrow$ 0.25 m).

Shermeyer and Van Etten [SVE19] conduct an extensive study, analyzing the effect of two different SR approaches on two object detection techniques with three scale ratios (i.e. $\times 2$, $\times 4$, and $\times 8$) on multiple GSDs ranging from 4.8 m to 0.3 m. One of the many insights in this study is that it improves the result to generate images of a higher GSD than a ground truth exists for, by applying a SR model trained on a different resolution. This means that generating images with a GSD of 0.15 m from 0.3 m imagery with a model trained to predict 0.3 m images from 0.6 m images, still performs better than directly applying vehicle detection on native ground truth 0.3 m images.

Mostofa et al. [MFRN20] propose a model that trains a SR algorithm jointly with a vehicle detection algorithm. They use a multi-scale GAN, allowing the network to produce a more detailed SR image by training on two different resolutions. The SR model produces an output more suitable for vehicle detection, by including the vehicle detection loss in the SR training. The combination of these techniques leads to an increased visual quality and vehicle detection accuracy compared to the state of the art.

Although there are not many papers exploring the potential of SR for vehicle detection, existing papers clearly show that there is an improvement in detection performance, especially when generating imagery with high resolution (i.e. ≤ 0.3 m GSD). Yet, no paper (to the knowledge of the author of this thesis) analyses the main idea of this thesis: if training the SR algorithm with training data focused on vehicles improves the results.

2.6 Summary

This thesis is built around a combination of several disciplines, each summarized with explanations of a selection of relevant state of the art papers. Both algorithms used in this thesis (RDN [ZTK+18] and Faster R-CNN [RHGS15]) are described in the context of other related methods.

In the last years, superior results in SR, as well as object detection (and its special case vehicle detection), are achieved with deep learning methods [FMN19, WSH20]. The analyzed SR models use one of three techniques to reach a higher resolution: bicubic upsampling, transposed convolutions, and sub-pixel convolutions. Based on these core ideas, the reconstruction performance is enhanced by various architectural choices, for example skip connections [KLL16a], attention mechanisms [ZLL+18], dense connections [ZTK+18], recursive layers [KLL16b], adversarial losses [LTH+17], and other sophisticated loss functions [LHAY17].

Recent vehicle detection approaches can be divided in two major categories: one-stage and two-stage architectures. Two-stage methods use a separate object proposal step, which provides the input to the main object detections step, leading to a higher performance than the faster one-stage approaches provide [LOW+20].

CHAPTER 3

Methodology

The goal of this thesis is to examine the differences of applying a vehicle-focused SR algorithm prior to vehicle detection, as opposed to applying conventional unfocused SR. In order to answer the research question, a pipeline of multiple consecutive steps is necessary. For a clearer understanding, these steps are visualized in Figure 3.1. The first step (starting from the left side in Figure 3.1) is to create the necessary datasets: one conventional dataset comprising remote sensing imagery and one vehicle-focused dataset, containing the same remote sensing imagery but only the parts showing vehicles. This is achieved by extracting crops at the position of every annotated vehicle.

After generating those datasets, one RDN [ZTK+18] is trained for each of the two different datasets. The purpose of the RDNs is to transform the remote sensing images from a GSD of 0.6 m to the higher resolution of 0.3 m (i.e. SR)[1]. By doing this step with two different datasets, two trained models are gained: a vehicle-focused one and an unfocused one.

Before the vehicle detection itself can be trained, datasets for this step have to be created. In order to examine the research question, the dataset (based on the test set of the dataset used for SR), is scaled up from a GSD of 0.6 m to 0.3 m with each SR model. This generates two different datasets with GSD 0.3 m (one created by vehicle-focused SR and one by unfocused SR).

This preparation step is followed by the vehicle detection. For this, the Faster R-CNN [RHGS15] object detection algorithm is trained to generate axis-aligned bounding boxes for vehicles. The network is trained once with the dataset scaled up by vehicle-focused SR, and once with the dataset created by unfocused SR.

[1]The GSD of 0.3 m is chosen as it is the lowest one provided by the source datasets.

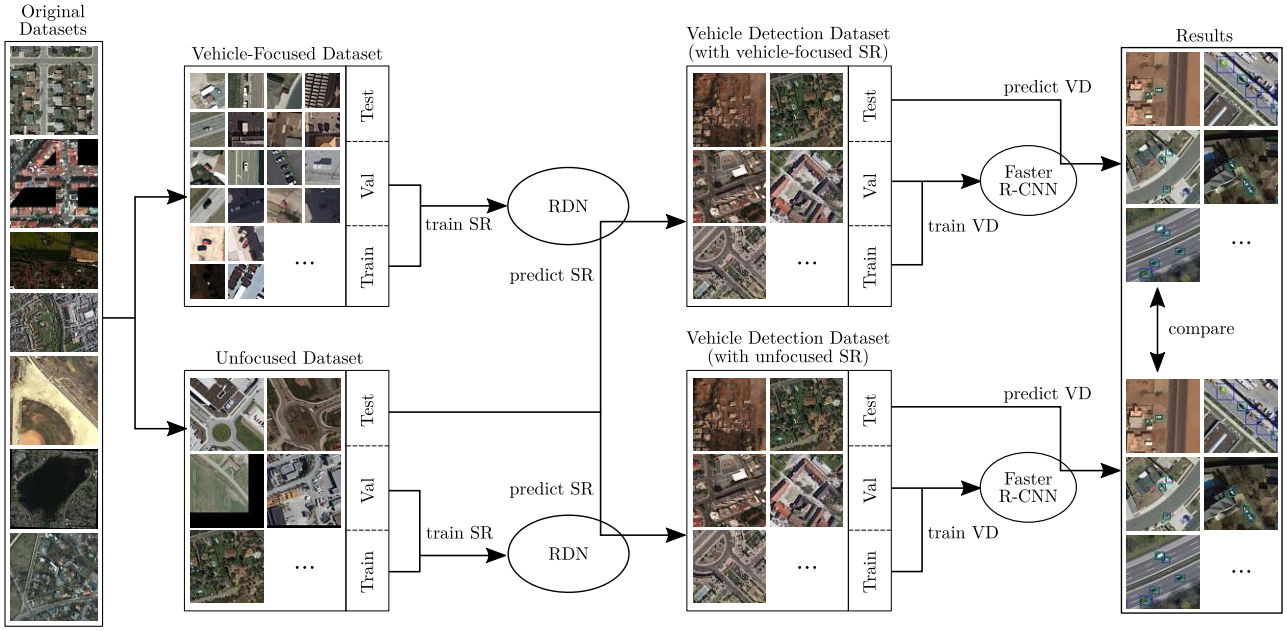


Figure 3.1: The basic steps of the study conducted in this thesis.

The following chapter provides details about the previously mentioned steps. It begins with how the dataset is created, then the creation of the SR models is elaborated, followed by the vehicle detection step.

3.1 Dataset Construction

The data needed to train and test the SR, as well as the vehicle detection step, is gathered from multiple source datasets, with the goal of having a diverse and sufficiently large combined dataset. As the size of the training set has an influence on the performance (according to Sun et al. [SSSG17], the performance of vision tasks improves logarithmically with the training set size), seven datasets are combined in this thesis, to form a more comprehensive dataset. An additional reason for this idea is to form a more diverse dataset, including a higher variety of lighting situations and geographic areas. The VEDAI dataset [RJ16] for example, is based on the Utah AGRC (HRO 2012 6 inch photography) dataset[2], which contains only aerial images of selected areas of Utah. By combining several datasets, the potential drawbacks (e.g. a limited variability of vegetation, population density and thus car density) of a homogeneous dataset are avoided.

For the different steps in the pipeline, multiple versions of the dataset are necessary. For the vehicle-focused SR, a dataset consisting of one image per vehicle is created (called the *vehicle-focused dataset* in this thesis), while a dataset based on the same imagery, but

[2] `https://gis.utah.gov/data/aerial-photography/#HRO`, last retrieved on 2020-12-21

not focused on vehicles is needed for the unfocused SR (named the *unfocused dataset*). In addition, the same data is processed to be available in the PASCAL format [EVGW+10] in a uniform manner (i.e. all images have the same size) for the vehicle detection step (subsequently called the *vehicle detection dataset*).

3.1.1 Composition of the Datasets

The dataset composition consists of multiple steps, beginning with the general analysis of all source datasets. As every dataset encodes annotation information differently (e.g. as one single geojson or as one xml per image), all data has to be read in separately and converted to the same format. During this process of dataset creation, the classes used have to be harmonized, as there are datasets with classes of annotated vehicles, such as aircraft, which are not in the scope of this thesis and have to be removed. Details of how each of the source datasets are processed are provided in the next section. Additionally, the GSD has to be harmonized to 0.3 m for all resulting datasets.

The unfocused dataset consists of the whole (i.e. not cropped) images of the source datasets, edited to have a uniform GSD of 0.3 m. Apart from that the images are not modified for that dataset. As this dataset is only used for the SR training, it is not necessary to consider any information from the annotations.

The vehicle-focused dataset is created by saving one crop per annotated vehicle as a separate image. These images with a resolution of 64×64 pixels (and three color channels), a size large enough to provide context around vehicles and also contain large vehicles, are centered around the annotated position of a crop. This process is visualized in Figure 3.2, where you can see three example crops.

The vehicle detection dataset comprises of 3 676 images, depicting an area containing 4 3862 vehicles. The source images are cropped into squares of 540×540 pixels (with an overlap of 20 pixels to adjacent crops). If the crop crosses the border of the image due to the fixed resolution, the remaining crop is filled with black pixels. Crops which only have an area of 500 pixels or less inside the image border and crops without any vehicles are excluded. This process can be seen in Figure 3.3, where three example crops are depicted (*B*, *C*, and *D*). Crop *A* is an example for an ignored crop, because it contains no vehicles. Note that crop *D* contains 184×245 (area: 45 080) pixels of the source image, and is thus above the 500 pixel threshold and not dismissed. The annotations are transformed into a unified format using axis-aligned bounding boxes. The reason for axis-aligned bounding boxes is that not all source datasets provide all necessary information (i.e. orientation of the vehicle) for oriented bounding-boxes. The set is based only on the images from the test set of the unfocused dataset. This is done in order to ensure that the training of the SR step does not imply that the images used for testing the vehicle detection step were already seen (during the SR step) and therefore influence the result.

Figure 3.2: For the vehicle-focused dataset, crops are extracted from remote sensing images. This example shows part of an image from the Dataset for Object deTection in Aerial images (DOTA) dataset [XBD+18] and three vehicle-centered crops of that image.

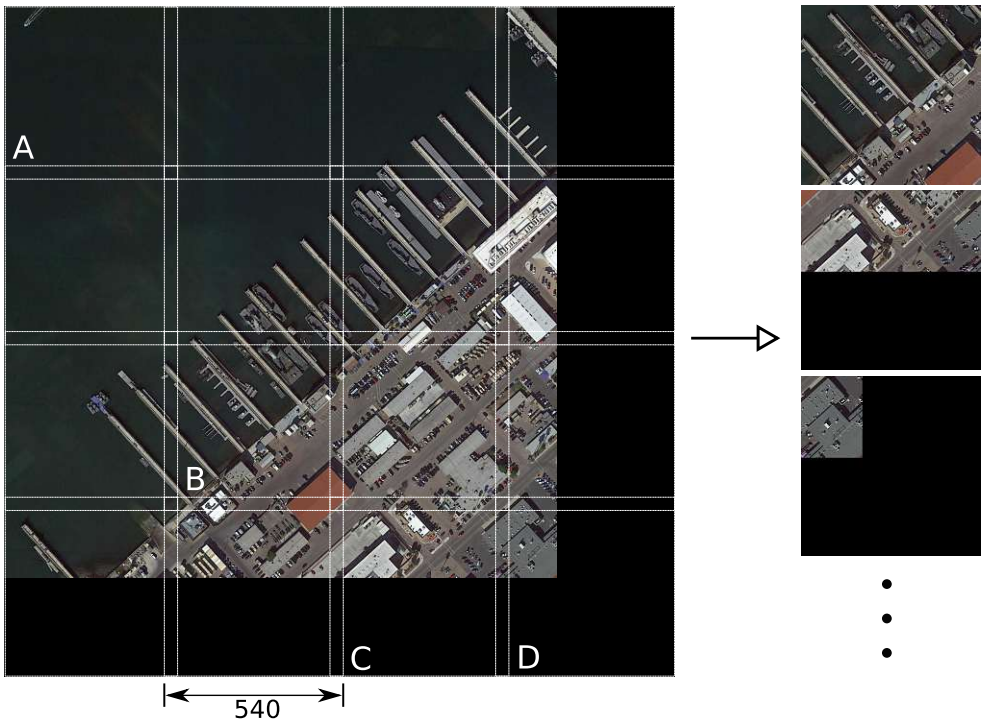


Figure 3.3: The vehicle detection dataset is created by splitting images into overlapping crops of size 540 × 540. Crop A is skipped because it contains no vehicles. Crops B, C, and D are shown as examples for accepted crops on the right. Example image taken from the DOTA dataset [XBD+18].

3.1.2 Source Datasets

The data used in this thesis is a combination of seven separate datasets: Vehicle Detection in Aerial Imagery (VEDAI) [RJ16], DLR-MVDA [LM15], Cars Overhead With Context (COWC) [MKSB16], DOTA [XBD+18], DIUx xView 2018 Detection Challenge Dataset (xView) [LKM+18], Overhead Imagery Research Data Set (OIRDS) [TCP+09], and Parking Cars Barcelona (PaCaBa) [ZLP+20]. Main properties and usage of these datasets are summarized in Table 3.1 and explained in detail in the following sections. Examples for images from all of those datasets can be seen in Figure 3.4.

Dataset	Location	Covered Area (km^2)	Annotated Objects	GSD (in m)	Used in
VEDAI	Utah (US)	166.2	3 757	0.125	all (test)
xView	various	1 415	1 000 000	0.3	all
DOTA	various	unknown	188 282	varying	all
DLR-MVDA	Munich (DE)	54.7	3 505	0.13	all (test)
COWC	4 countries	unknown	32 716	0.15	SR
OIRDS	unknown	unknown	1 796	varying	all (test)
PaCaBa	Barcelona (ES)	4.6	24 336	0.34	VD (test)

Table 3.1: Properties of used source datasets. *SR* denotes the vehicle-focused and the unfocused dataset, *VD* denotes the vehicle detection dataset, *all* is the combination of *SR* and *VD*, and *test* means that the dataset is only used in the respective test set (of the vehicle detection, the vehicle-focused, or the unfocused datasets).

The originally considered ISPRS 2D Semantic Labeling Contest dataset[3] is not used in this thesis, as it appears to be contained in the COWC dataset [MKSB16]. The dataset NWPU VHR-10[4] is not used, as it does not provide a defined GSD per image, rendering it impossible to scale this dataset to the same GSD as the other datasets.

VEDAI

The VEDAI dataset [RJ16] consists of 1 268 color images with a resolution of $1\,024 \times 1\,024$ pixels per image. In addition to these images, there is a matching gray-scale image per color image, which contains an infrared capture of the same area. According to the dataset authors [RJ16] the GSD of every image is 12.5 cm and every annotated vehicle is assigned to one of nine classes (“plane”, “boat”, “camping car”, “car”, “pick-up”, “tractor”, “truck”, “van”, and “other”). The annotations describe 3757 vehicles with their oriented bounding box, the class, and two flags indicating if the vehicle is completely contained in the image and if it is occluded. For this thesis, vehicles which are not completely depicted (according to the flag) and vehicles with the classes “plane” and

[3] http://www2.isprs.org/commissions/comm3/wg4/semantic-labeling.html, last retrieved on 2019-06-23

[4] http://www.escience.cn/people/JunweiHan/NWPUVHR10dataset.html, last retrieved on 2019-06-23

(a) COWC [MKSB16] (b) DLR-MVDA [LM15] (c) DOTA [XBD+18]

(d) OIRDS [TCP+09] (e) PaCaBa [ZLP+20] (f) VEDAI [RJ16]

(g) xView [LKM+18]

Figure 3.4: Unedited (i.e. apart from creating a squared crop) examples of all datasets used in this thesis.

"boat" are ignored, while occluded vehicles are included to make the model more robust in difficult situations like occlusions.

The dataset contains 3 757 vehicles (in 1 268 images), of which only one is annotated to be occluded. 54 vehicles are marked as not completely contained in the image and are therefore excluded. Also excluded are 228 vehicles belonging to the "plane" and "boat"

classes. An additional 710 vehicles are left out because their position near the border of the image does not allow a centered crop. This results in 2 770 image crops[5] for the vehicle-focused dataset.

xView

The xView dataset [LKM+18] is the largest[6] dataset used in this thesis and contains around one million labeled objects of 60 classes. As only the annotations of the training subset are public, only this subset is used, which reduces the number of annotated objects to 601 937. The 60 classes are not only limited to vehicles, they also mark buildings and types of vehicles (e.g. aircraft, ships, trains,...) not of interest for this thesis. 19 out of the 60 classes are selected and used[7]. This selection of classes removes 344 280 annotated objects from other classes. After also discarding annotations which are too close to the image border and therefore prevent a centered crop around them, and annotations of an image missing from the dataset, there are 218 967 annotations left for the vehicle-focused dataset. The images of the xView dataset already have a GSD of 0.3 m, thus no scaling is needed.

DOTA

The DOTA dataset [XBD+18] contains 2 806 images of varying GSD with 188 282 objects in 15 classes. Of the 1 411 images in the training set of DOTA, 15 are removed because they do not have an annotated GSD, and another 456 images are not used because the GSD is below 0.3 m. The 15 classes of the dataset are: “plane, ship, storage tank, baseball diamond, tennis court, swimming pool, ground track field, harbor, bridge, large vehicle, small vehicle, helicopter, round-about, soccer ball field and basketball court” [XBD+18]. In order to match the other datasets used, only two categories are selected: large vehicle and small vehicle. Objects of other classes are ignored, resulting in 97 776 remaining objects. Every object is annotated with an oriented bounding box, which is converted to an axis-aligned bounding box.

DLR-MVDA

The DLR-MVDA dataset [LM15] consists of 10 annotated images (and 10 images without annotations in the test set) captured over Munich, Germany from a plane flying approximately 1 000m above ground. The images have a GSD of 0.13 m. The dataset contains 3 505 vehicles of the classes car, car with trailer, truck, truck with trailer, van with trailer, long truck, and bus. The majority (i.e. 3 419) of the vehicles belong to the

[5] The number of accepted vehicle crops (2 770) is higher than the naive calculation would suggest ($3\,757 - 54 - 228 - 710 = 2\,765$), because a single crop can be excluded because of more than one reason.

[6] most annotated vehicles

[7] The following xView classes are used: Passenger Vehicle, Small Car, Bus, Pickup Truck, Utility Truck, Truck, Cargo Truck, Truck Tractor w/ Box Trailer, Truck Tractor, Trailer, Truck Tractor w/ Flatbed Trailer, Truck Tractor w/ Liquid Tank, Crane Truck, Engineering Vehicle, Reach Stacker, Dump Truck, Front loader/Bulldozer, Excavator, and Cement Mixer

car class. For the vehicle-focused dataset, 293 vehicles are not included, either because their bounding box intersects with the image border, or because they are too close to the image border. The oriented bounding boxes are converted to axis-aligned bounding boxes.

COWC

The COWC dataset [MKSB16] is constructed from six (partially gray-scale) image sets captured in locations across four countries (Canada: Toronto, New Zealand: Selwyn, Germany: Potsdam and Vaihingen, and United States: Columbus and Utah). The vehicles contained in the dataset have no specific class assigned (i.e. all are cars). The GSD is 0.15 m and therefore has to be scaled down to 0.3 m. The dataset also contains a list of (hard) negatives, which are not used in this thesis.

The dataset does not contain bounding box annotations. Instead, it provides the center of every vehicle. This is enough information for using it as part of the vehicle-focused dataset, as a fixed-size crop around every center can be created. But it is not possible to train vehicle detection without a bounding box and consequently COWC does not appear in the vehicle detection dataset.

OIRDS

OIRDS [TCP+09] is the smallest dataset used in this thesis. It consists of 1 796 vehicles, of which 1 070 cannot be used for the vehicle-focused dataset as they are too close to the image border and there is not enough space around them to create a crop centered around the vehicle, as necessary for the vehicle-focused dataset. 177 vehicles are discarded because the assigned probability is not 100% (i.e. not all human raters agreed that there is a vehicle). The vehicles are annotated with a polygon, which is converted to an axis-aligned bounding box. One additional vehicle is discarded because it has no polygon assigned. All vehicle classes (e.g. “vehicle/pick-up”, “vehicle/car”,...) are accepted. The images of the dataset have different GSDs and are scaled to 0.3 m.

PaCaBa

The PaCaBa dataset [ZLP+20] contains stereo images from four regions in Barcelona, Spain. The images originate from a satellite, which captured the same area twice within one minute and moved around the earth in between the two shots. For this thesis, no stereo images are needed and thus only one (i.e. “Image1”) of the two images per region is used. The dataset contains oriented bounding boxes, which are converted to axis-aligned bounding boxes. The GSD is changed from 0.34 m to 0.3 m by scaling the image.

This dataset is neither part of the datasets for SR (i.e. vehicle-focused dataset and unfocused dataset), nor of the vehicle detection dataset. It is only used as a separate unseen evaluation dataset for vehicle detection.

Source dataset	Images
COWC	32 744
DLR-MVDA	3 212
DOTA	97 776
OIRDS	547
VEDAI	2 770
xView	218 967

Table 3.2: Images per source datasets for the vehicle-focused dataset.

3.1.3 Description of the Datasets

There are three datasets created for this thesis, each needed for a different step in the pipeline: the vehicle-focused dataset, the unfocused dataset, and the vehicle detection dataset. Detailed descriptions of the three datasets follow below.

Vehicle-Focused Dataset

As the central goal of this thesis is to determine if vehicle detection is improved by training a SR algorithm with training data focused on vehicles, such training data is essential. The so-called vehicle-focused dataset is created for this purpose. It contains remote sensing images with a vehicle in the center of each image. By depicting at least one vehicle per training image, the SR network can be trained to generate higher-resolution versions of vehicles.

The dataset is generated from images of the source datasets (VEDAI, DLR-MVDA, COWC, DOTA, xView, and OIRDS). Every image in the generated dataset has a size of 64×64 pixels and is normalized to a GSD of 0.3 m.

The 356 016 images of which this dataset consists originate from the different source datasets as described in Table 3.2. It can be seen that the majority of the images are from xView.

This dataset is split into three subsets for training, testing, and validation, respectively. The source datasets DLR-MVDA [LM15], OIRDS[TCP+09], VEDAI[RJ16] are smaller in size than the other three source image sets. As they would only account for a negligible part of the training set, they are not included in the training set. Instead, they are only used in the test subset, providing a clue on the cross-domain performance (i.e. how is the performance on those datasets, which were not part of the training at all) of the SR step. The other three datasets (i.e. COWC, DOTA, and xView) are defined to be split by the ratio $80 : 10 : 10$ to training, validation, and test set, respectively. This results in the composition of the three subsets as given by Figure 3.5, where it can be seen that xView contributes more than 50% of the images to each subset. In total, the training set comprises 279 535, the test set 41 490, and the validation set 34 991 images, resulting in 356 016 images in total.

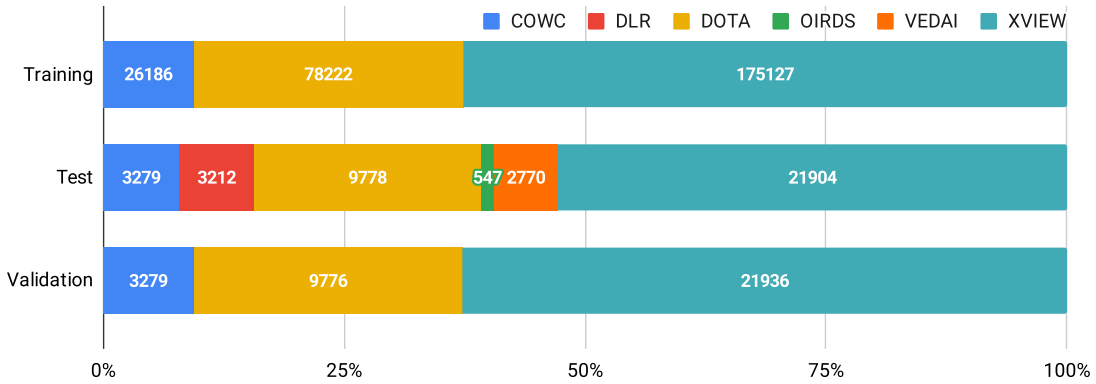


Figure 3.5: The composition of the training, test, and validation subsets of the vehicle-focused dataset. The numbers inside the bars indicate the number of images contributed to each subset.

For examples of the resulting vehicle-focused images see Figure 3.6. Note how there is always one vehicle in the center of the image, and potentially several other vehicles around it. Because there is a crop for every annotated vehicle, one vehicle can be depicted on multiple crops (i.e. on their "own" crop and on the crops created because of adjacent vehicles).

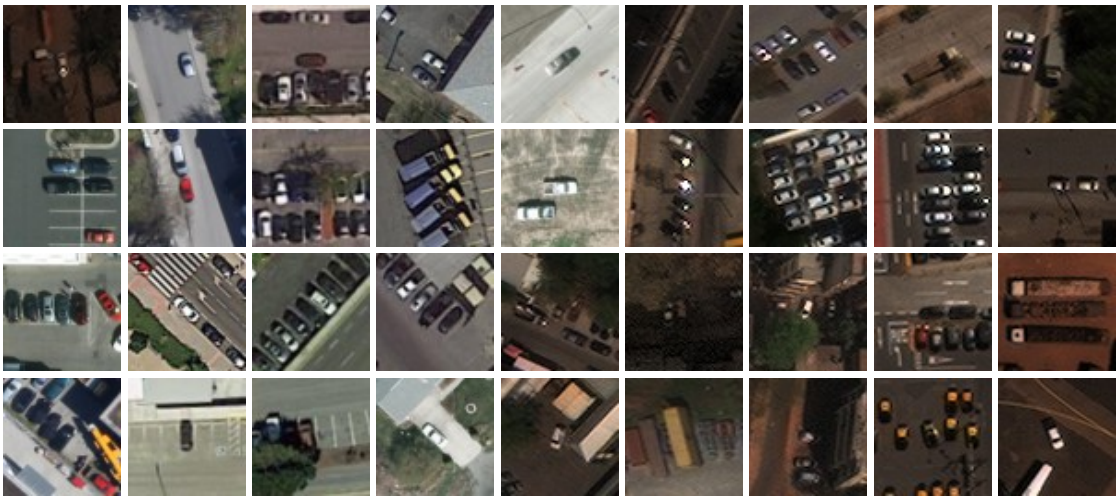


Figure 3.6: Examples of images in the vehicle-focused dataset.

Every image in this dataset has a GSD of 0.3 m with a resolution of 64×64 pixels, which equals a depicted area of 19.2×19.2 meters ($368.64\ m^2$). For SR it is necessary to provide a training set with high resolution (in this case the target resolution of 0.3 m) and low resolution (i.e. GSD 0.6 m) image pairs. For this dataset, this is achieved by applying bi-linear downsampling to every high resolution image and saving the result as

Dataset split	Images	Area covered (in km^2)	Average area per image (in km^2)
Training	1 061	665.60	0.627
Test	1 753	102.74	0.058
Validation	199	89.98	0.452

Table 3.3: The dataset split of the unfocused dataset.

a low-resolution version of that image. This results in an additional 356 016 images with a resolution of 32×32 pixels, depicting the same 368.64 m^2 as their corresponding high resolution versions.

Unfocused Dataset

The center of this thesis is to explore the idea of training the preprocessing SR step, before vehicle detection, in a vehicle-focused manner. This vehicle-focus is achieved by using the vehicle-focused dataset. In order to be able to measure how this influences the result, a comparison dataset is necessary, namely the unfocused dataset. It is based on the same remote sensing imagery datasets as the vehicle-focused dataset, but uses them directly (i.e. not focused/cropped). Consequently, every vehicle depicted in the vehicle-focused dataset is also depicted in the unfocused dataset, although the density of vehicles is much lower in the unfocused dataset (i.e. it contains 6.54 times the pixels, while containing approximately[8] the same number of vehicles as the vehicle-focused dataset). Images with not at least one depicted vehicle are excluded from the dataset. The only editing step done is to scale all images to a GSD of 0.3 m and convert all images to the Portable Network Graphics (PNG) format, if necessary. The images are not cropped or limited to a specific size, causing a wide variety of image sizes to be part of the dataset. The smallest image has a size of 128×128 pixels and the largest one has a size of $9\,200 \times 9\,037$ pixels.

To ensure comparability, the split into training, test, and validation sets follows the split in the vehicle-focused dataset. The results of this split can be seen in Table 3.3. Note that the average area varies by more than an order of magnitude between the splits. The small average area per image in the test set can be explained by the different source datasets. The OIRDS [TCP+09] dataset is entirely contained in the test set and its 411 images only have an average area of 20 410 pixels (1837 m^2), while the xView dataset, which contributes substantially to the training set, has an average size of 9 832 515 pixels (884 926 m^2) per image.

Figure 3.7 shows examples of images in the unfocused dataset. Note that Figure 3.7a shows the previously mentioned largest image, and Figure 3.7d shows the smallest image

[8] In general, the same vehicles are contained in both datasets, but vehicles too close to the borders of the images are left out of the vehicle-focused dataset and there are vehicles depicted multiple times, due to overlapping areas between image crops.

in the dataset. The images vary from mainly depicting agricultural areas (with nearly no vehicles) to densely populated areas and airports (with a high density of vehicles).

(a) Dataset source: COWC [MKSB16]

(b) Dataset source: DOTA [XBD+18]

(c) Dataset source: DOTA [XBD+18]

(d) Dataset source: OIRDS [TCP+09]

(e) Dataset source: VEDAI [RJ16]

(f) Dataset source: xView [LKM+18]

Figure 3.7: Examples of images in the unfocused dataset.

As with the vehicle-focused dataset, a low-resolution version of this dataset is created for SR. It has a GSD of 0.6 and is gained by bi-linear downsampling.

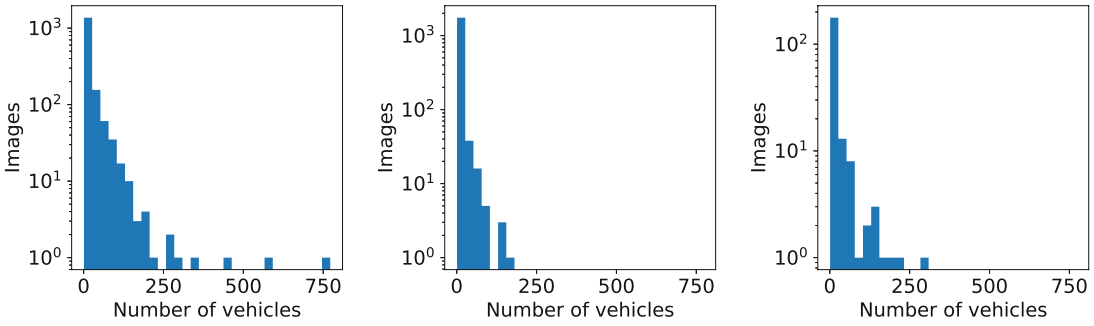
Vehicle Detection Dataset

The second major step in this thesis besides SR is vehicle detection. In order to train a deep neural network for this task, a dataset providing annotated vehicle positions (i.e. bounding boxes) is used. The dataset created and used for this task in the scope of this thesis is denoted vehicle detection dataset. It consists of 3 676 images with a resolution of 540×540 pixels and is generated from the test set of the unfocused dataset. The test set is chosen as the source because the dataset has to be processed by the SR step. In order to allow this step to be performed on data not yet seen by the SR, the test set is used as the basis for the vehicle detection dataset.

The vehicle detection dataset annotations are formatted in the PASCAL format [EVGW+10], meaning that per image there is one Extensible Markup Language (XML) file, containing information about the (axis-aligned) bounding box of every vehicle depicted in the image.

The dataset is split into training, test, and validation subsets. Of the images left for this dataset (i.e. the images of the test set of the unfocused dataset) 80% of the DOTA and the xView datasets form the training set. Another 10% of those two source datasets constitute the validation set, and the remaining 10% are in the test set, together with the other source datasets (i.e. VEDAI, DLR-MVDA, and OIRDS, but not COWC as it does not provide bounding boxes).

For a deeper understanding of the dataset, Figure 3.8 visualizes the amount of vehicles per image in the training, test, and validation subsets. As one can see, images with more than 100 vehicles are rare and the histogram bin with the highest number of images is the lowest one. Note the logarithmic scale of the y-axis, which indicates a difference of around an order of magnitude to the second bin. While the training set contains several images with a high number of vehicles (maximum: 772 vehicles, compared to 302 in the validation set and 156 in the test set), the mean is similar to the validation set (training set: 18.4, validation set: 18.6 vehicles per image). The testing set, on the other hand, shows a lower average of 5.2 vehicles per image, as it contains other source datasets.



(a) Training (b) Test (c) Validation

Figure 3.8: Distribution of the number of vehicles per image in the vehicle detection dataset.

The training set contains 30 537 vehicles, the validation set 3 870, and the test set 9 455. Example images of this dataset can be seen in Figure 3.9. Note how there are black bars in two of the images. This is because the images in the vehicle detection dataset are cropped to have a resolution of 540 × 540 pixels. For crops near the border (or small source images), the pixels lying outside the source image are padded in black.

Figure 3.9: Examples of images in the vehicle detection dataset. Source datasets of the images from left to right: VEDAI [RJ16], xView [LKM+18], and DOTA [XBD+18]

3.2 Super-Resolution

Super-Resolution (SR), in this context Single Image Super Resolution (SISR), is the process of creating a high resolution image from a low resolution version of that image [YZT+19]. In this thesis, SR is applied as a preprocessing step to improve the performance of vehicle detection, as shown by Shermeyer and Van Etten [SVE19]. The central idea is to compare two SR training strategies: conventional and vehicle-focused training.

3.2.1 RDN

The SR architecture chosen for this thesis is the RDN, proposed by Zhang et al. [ZTK+18], which is a convolutional neural network. The source code used is based on the implementation by Francesco Cardinale et al., as published on GitHub[9]. The criteria for choosing a SR algorithm for this thesis are the performance of the algorithm, its support of a scaling factor of 2 (i.e. doubling the image resolution per axis by applying SR), and the public availability of a pre-trained model (because not having to start from scratch could save valuable computation time and speed up convergence [HGD19]). The appearance in recent surveys about SR (see Yang et al. [YZT+19], Anwar et al. [AKB20], and Wang et al. [WCH20]), multiple available implementations[10], and the high number of citations[11]

[9] `https://github.com/idealo/image-super-resolution`, last retrieved on 2021-03-14

[10] Incomplete list of publicly available RDN implementations:
`https://github.com/yjn870/RDN-pytorch`,
`https://github.com/hengchuan/RDN-TensorFlow`,
`https://github.com/seathiefwang/RDN-Tensorflow`,
`https://github.com/rajatkb/RDNSR-Residual-Dense-Network-for-Super-Resolution-Keras`,
`https://github.com/lingtengqiu/RDN-pytorch`, and
`https://github.com/idealo/image-super-resolution`, all last retrieved on 2021-03-14

[11] 961 citations according to `https://www.researchgate.net/publication/329744327_Residual_Dense_Network_for_Image_Super-Resolution`, 1 086 citations according to `https://scholar.google.com/scholar?cluster=10362327168909609533`, both last retrieved on 2021-03-14

indicate the popularity and quality of the algorithm. The other two requirements are fulfilled as a model file for the scaling factor 2 is available[12].

The Edge-Enhanced Generative Adversarial Network by Jiang et al. [JWY+19], designed directly with a focus in remote sensing imagery, is also publicly available[13], but the pre-trained model file for the scaling factor 2 is not provided, which is why RDN [ZTK+18] was selected for this thesis instead.

Model Architecture

The architecture of the RDN [ZTK+18] can be seen in Figure 3.10 and is described in this section. The network can be interpreted as consisting of multiple consecutive parts, which Zhang et al. [ZTK+18] call Shallow Feature Extraction Net (SFENet), then a group of RDBs, followed by the Dense Feature Fusion (DFF), and finally the Up-Sampling Net (UPNet). Every step is operating in the LR space, until the resolution is increased in the last step (i.e. the UPNet).

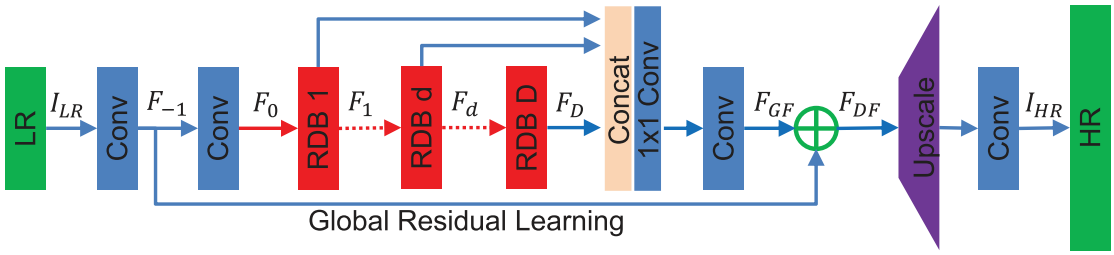


Figure 3.10: Architecture of the RDN [ZTK+18]. Image taken from [ZTK+18].

The image is first processed by SFENet, which is the name for the first two convolutional layers in RDN. Both layers have G_0 filters of size 3×3 and the input is padded with zeros in order to keep the input and output size fixed for all layers.[14]

The output of that is fed into a series of D RDBs. A RDB, as shown in Figure 3.11, is a group of convolutional layers that is arranged following a specific pattern: It contains C convolutional layers, each of size 3×3 with G filters and followed by a ReLU [GBB11] activation function. Note that every convolutional layer uses feature maps produced by all previous convolutional layers (in the same RDB) and additionally the input to the RDB. Thus, the number of feature maps fed into the convolutional layers grows with each layer by G (which is why it is called growth rate).

After those layers, every RDB contains a 1×1 convolutional layer, which is applied to the concatenated output of all convolutional layers and the input of the RDB. This layer reduces the number of feature maps coming out of an RDB to G_0, a process which Zhang

[12] https://github.com/idealo/image-super-resolution/tree/master/weights/sample_weights/rdn-C6-D20-G64-G064-x2/PSNR-driven, last retrieved on 2021-03-14

[13] https://github.com/kuijiang0802/EEGAN, last retrieved on 2021-03-14

[14] Note that this is valid for nearly all layers.

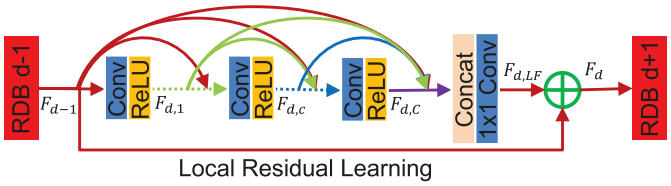


Figure 3.11: Architecture of the RDB, a part of the RDN architecture. Image taken from [ZTK+18].

et al. [ZTK+18] call local feature fusion. This is necessary due to another mechanism in RDBs: local residual learning. As empirically shown by Zhang et al. [ZTK+18], this technique enhances the network's performance, by adding the input of each RDB to the output of the same RDB, forcing the network to learn to predict a difference to the input. For this addition to work trivially, local feature fusion reduces the dimensionality of the output of the convolutional layers to match the input.

After D RDBs, the next part of the network is called DFF. The mechanism has the same purpose as the local feature fusion inside each RDB: Reducing the number of feature maps, while incorporating all outputs of previous layers, in this case the outputs of all RDBs. This is done by a 1×1 and a consecutive 3×3 convolutional layer, after which the output of the very first convolutional layer of the network is added to the result (a technique called *global residual learning*).

The last step of the network is the UPNet (denoted as *Upscale* and a convolutional layer in Figure 3.10). Zhang et al. [ZTK+18] use a technique proposed by Shi et al. [SCH+16] in their ESPCN, called sub-pixel convolution layer. This method makes use of a convolutional layer, which increases the number of filter maps by a factor of r^2, where r is the desired SR scale factor (i.e. $r = 2$ in this thesis). Then, the periodic shuffling operator rearranges elements from r^2 filter maps to 1 filter map. Every block of $r \times r$ elements in the filter map contains exactly one element per connected feature map from the previous layer. This process of periodic rearrangement is illustrated in Figure 3.12. After that, Zhang et al. [ZTK+18] add another 3×3 convolutional layer, which outputs 3 channels, giving the final SR color image.

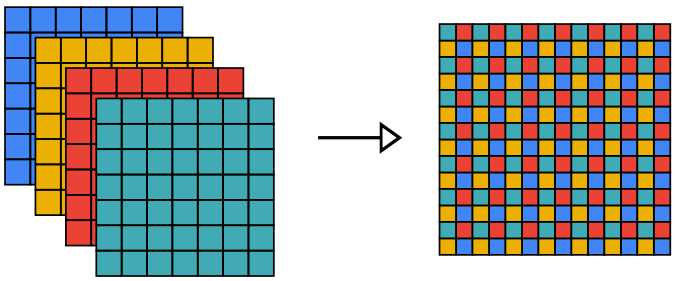


Figure 3.12: Own illustration of the periodic shuffling operator in the sub-pixel convolution layer as proposed by Shi et al. [SCH+16]

The RDN [ZTK+18] network provides several parameters to change the properties of the architecture described above. The parameters are set to the values given in the following paragraph.

The number of RDBs is denoted D and set to 20. The parameter G_0 denotes the number of feature maps in the input and the output of each RDB and is 64. The growth rate G defines the number of feature maps produced by each convolutional layer inside a RDB and is also set to 64. Each RDB contains C convolutional layers, a parameter that is set to 6 in this thesis. All parameters are chosen to match the parameters of the pretrained model[15] in order to be compatible with it.

Data augmentation is applied by randomly flipping (horizontally and vertically) and rotating the training images. For training, the L_1 loss function is used. The learning rate is set to 0.0004 and learning rate decay reduces the learning rate by half every 30 epochs.

3.2.2 Workflow

The training of the RDN [ZTK+18] is done on the Vienna Scientific Cluster 3 (VSC-3)[16], more specifically on one of the computing nodes equipped with an NVIDIA Pascal GeForce GTX 1080 GPU.

The training (and validation) data is saved in a compressed archive on the storage system of the VSC-3[17], in order to save disc space. At the beginning of a training session, the data is decompressed and loaded to a temporary storage partition, which is entirely located in the Random-Access Memory (RAM). This ensures fast read times during the training process and reduces the number of requests to the persistent storage system of the VSC-3, which is shared with other users (possibly influencing their storage performance).

Similarly to Girshick [Gir15] and Ren et al. [RHGS15], a mini-batch is created by sampling all (i.e. 8) 64×64 image patches in a mini-batch from the same image. This strategy does fit for the vehicle-focused approach, where every image is only of size 64×64, thus different random images are sampled.

The weights of the model are saved automatically whenever the evaluation based on the validations set shows an improvement (i.e. PSNR is higher than after any other epoch) and at the end of each training run. In order to be able to react to unexpected problems (e.g. bugs, no convergence, bad PSNR validation results), the network is only trained in runs of 25 epochs, after which the intermediate results are analyzed. If everything is as expected, the training is continued based on the weights after the last epoch. This is intended to limit the amount of computing time wasted with training that produces no usable results.

[15] https://github.com/idealo/image-super-resolution/tree/master/weights/sample_weights/rdn-C6-D20-G64-G064-x2/PSNR-driven, last retrieved on 2021-03-14

[16] https://vsc.ac.at/systems/vsc-3/, last retrieved on 2021-03-16

[17] https://vsc.ac.at/systems/vsc-3/, last retrieved on 2021-03-16

3.3 Vehicle Detection

Vehicle detection in the scope of this thesis is the process of obtaining a bounding box around a vehicle on an remote sensing image, and is thus a subclass of the discipline of object detection [CH16]. In order to answer the research question, the same neural network is trained twice to detect vehicles. It is trained once with the vehicle detection dataset based on the vehicle-focused SR, and once with the vehicle detection dataset based on the unfocused SR.

3.3.1 Faster R-CNN

For vehicle detection, the object detection algorithm Faster R-CNN [RHGS15] is used. This algorithm outputs a bounding box per detected vehicle on the input image. The reason for selecting this object detection algorithm is – apart from its widespread use[18] – that it is used in a previous project (see [ZLP+20]) by Sebastian Zambanini, of which the source code is used as a starting point for this part of the master thesis. It is a modification of the Faster R-CNN implementation by Yang et al. [19]

The network is trained with a batch size of 4 in combination with a learning rate of 0.004, as this is one of the benchmark configurations used by Yang et al. [20]. In contrast to this benchmark configuration, no learning rate decay is used, as empirical evaluation shows no benefit in this case. The setting for the maximum number of predicted boxes is set to 772, because this is the highest number of vehicles per image in the dataset.

Data augmentation for the training set is implemented by applying random rotation and flipping. As during SR training, the model file is saved additionally to the end of the training session, after each epoch that caused an improvement in the validation loss.

Model Architecture

The Faster R-CNN [RHGS15] model consists of two main parts: the RPN and the detection network itself, which equals the detection network from the predecessor called Fast R-CNN [Gir15]. The main idea of Faster R-CNN [RHGS15] is the performance benefit gained by sharing convolutions in the region proposal step and the actual detection step, and thus saving time (i.e. the proposal step takes 10 ms per image, compared to older approaches taking between 0.2 s and 2 s according to Ren et al. [RHGS15]).

The RPN uses the output of the shared (i.e. also used by the detection network part) convolutional layers, for which the first 13 convolutional layers of the VGG-16 model

[18] 10 133 citations according to https://www.researchgate.net/publication/277722488_Faster_R-CNN_Towards_Real-Time_Object_Detection_with_Region_Proposal_Networks and 20 554 citations according to https://scholar.google.com/scholar?cluster=16436232259506318906, both last retrieved on 2021-04-06

[19] https://github.com/jwyang/faster-rcnn.pytorch, last retrieved on 2021-04-06

[20] https://github.com/jwyang/faster-rcnn.pytorch#benchmarking, last retrieved on 2021-04-06

by Simonyan and Zisserman [SZ15] are chosen[21]. Those VGG-16 layers are pre-trained on ImageNet [DDS+09, RDS+15]. It applies a sliding-window approach using a 3×3 window[22], where the first layer outputs a 512-dimensional feature vector, which is then further processed by two fully connected layers in parallel (i.e. sibling layers). One is responsible for box-regression (producing the bounding box coordinates for the object), the other one is for classification, producing an objectness score (i.e. a number indicating the probability of the box to be background or containing an object). Instead of producing one bounding box proposal including a corresponding objectness score per 3×3 window, the RPN produces k proposals per window. By basing those proposals on so-called anchors with defined sizes, they cover multiple scales and aspect ratios, making the approach scale-invariant. The RPN regresses a bounding box per anchor, changing the original size and aspect ratio of the anchor to fit the potential object. k is 3 for this thesis, as only one anchor scale (16^2) and three aspect ratios (1:1, 1:2, and 2:1) are used. In the original paper, Ren et al. [RHGS15] use the same aspect ratios but three anchor scales ranging from 128^2 to 512^2, causing k to be 9. The reason for the different anchor scales is the different size of the target objects (detecting vehicles of roughly the same size in remote sensing imagery compared to general object detection with objects that could cover the whole image or just a small part of it).

The RPN is implemented using convolutional layers, which are equivalent to a sliding window with fully connected layers. This results in the RPN consisting of solely three convolutional layers in total, two of which being in parallel.

The second part of the network is called Fast R-CNN because it equals the object detection network of the same name by Girshick [Gir15]. In the original paper (i.e. [Gir15]), the region proposals that this network uses as input are provided by Selective Search [UVDSGS13]. This algorithm provides proposals in multiple scales, by joining segmented parts of an image iteratively based on similarity (measured in different color spaces, to capture a diverse set of regions) between adjacent regions, until there is only one region left. For every segmentation region, which appears during this iterative process, a bounding box is generated as a proposal [UVDSGS13]. The main contribution of Faster R-CNN [RHGS15], compared to its predecessor Fast R-CNN [Gir15], is the replacement of Selective Search [UVDSGS13] with the RPN, leading to a training that is 9× faster according to Ren et al. [RHGS15].

Fast R-CNN is given the region proposals of RPN and starts with the RoI pooling layer. This layer, as proposed by Girshick [Gir15], projects the bounding box of a region proposal to the output of the last shared convolutional layer. This area, which contains the convolutional features generated from the region in the image, which is defined by the bounding box, can vary in size. In order to enable further processing with fully connected layers, a fixed size (i.e. $H \times W$, set to 7×7) is necessary, which is achieved

[21] The original Faster R-CNN paper by Ren et al. [RHGS15] additionally experiments with the usage of the ZF network by Zeiler and Fergus [ZF14], as an alternative to VGG-16 [SZ15].

[22] Note that the 3×3 input is from the feature maps of the VGG-16 model [SZ15], resulting in a receptive field of 228×228 pixels of the original image.

by RoI pooling. By dividing the feature map into a regular grid of $H \times W$ rectangles, and then using max-pooling in every of those rectangles, the size is effectively reduced to $H \times W$ for every input size. Due to this architecture, Faster R-CNN can process (during training and testing) images of arbitrary size which is why Ren et al. claim it to be "fully convolutional" [RHGS15].

After the RoI pooling layer, the fixed size (7×7) feature map is fed into the first two fully connected layers (taken from VGG-16 [SZ15], with each 4096 channels). After that, two sibling fully connected layers produce a class label (in the case of this thesis, there are only two classes: vehicle and background) and a refined bounding box (four values per non-background class, resulting in just four values for this thesis), respectively.

The loss function L of the RPN is a multi-task loss, aiming to guide learning the classification part (which produces the objectness score), as well as the bounding box regression. Ren et al. [RHGS15] define it as:

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*). \tag{3.1}$$

This function takes the predicted probability p_i (i.e. the objectness score; 1 for an object/vehicle, 0 for background) and the predicted bounding box coordinates t_i as input. Normalization is achieved by including the mini-batch size N_{cls} and the number of anchor locations N_{reg}. The balance of the two parts of the loss function is given by the parameter λ, for which Ren et al. [RHGS15] show in experiments that it only has a minor influence on the results and suggest that $\lambda = 10$. The index i iterates over the anchors in a mini-batch, for which a log loss L_{cls}, comparing p_i to the ground truth label p_i^* (1 for an object/vehicle, 0 for background), and a regression loss L_{reg} (in this case the robust loss function as defined by Girshick [Gir15][23].), comparing the predicted bounding box coordinates t_i to the ground truth bounding box coordinates t_i^* is calculated:

$$L_{reg}(t, t^*) = \sum_{i \in \{x,y,w,h\}} \text{smooth}_{L_1}(t_i - t_i^*), \tag{3.2}$$

where

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{3.3}$$

Ren et al. [RHGS15] define a rule as to how the produced bounding boxes are matched with ground truth bounding boxes based on the IoU. This works by assigning a ground truth box to a calculated bounding box if this bounding box has the highest IoU value of all boxes intersecting this ground truth box, or if the IoU value is higher than 0.7. Additionally, every bounding box produced by the network, for which none of the two rules apply, and the IoU is lower than 0.3 for all ground truth boxes, is considered negative (i.e. background). All other bounding boxes do not contribute to the loss.

[23] Girshick [Gir15] calls this loss function L_{loc}

The loss function for the Fast R-CNN part of the network is given by[24]:

$$L(p, p^*, t, t^*) = -\log p_{p^*} + \lambda p^* L_{reg}(t, t^*), \tag{3.4}$$

where p is the predicted probability distribution for both classes, p^* is the ground truth class (1 for an object/vehicle, 0 for background), t is the predicted bounding box, and t^* is the ground truth bounding box. Consequently, p_{p^*} is the predicted probability that the proposal belongs to the ground truth class. λ balances the two terms in the loss function (the first term handles the classification part, the second the bounding box regression) and is set to 1.0. x, y, w, and h are coordinates and dimensions of the bounding box. L_{reg} is the function as already used by RPN and can be seen in Equation 3.2.

3.4 Summary

Three datasets are constructed (by combining six source datasets) in the scope of this thesis: the vehicle-focused dataset, the unfocused dataset, and the vehicle detection dataset. The first two are used for SR training and testing, the latter for vehicle detection training and testing. Two main SR models (using the RDN architecture [ZTK+18]) are trained: the vehicle-focused model (on the vehicle-focused dataset) and the unfocused model (on the unfocused dataset). After that, both models are used to create two SR-based versions of the vehicle detection dataset. The vehicle detection (applying the Faster R-CNN object detection algorithm [RHGS15]) is then trained twice on the vehicle detection dataset: once on the version created by the vehicle-focused SR model and once on the version created by the unfocused SR model.

[24] For this thesis the formulation of the loss function and its notations are changed from the original by Girshick [Gir15] for consistency with Equation 3.1 and for simplifications, as only two classes (i.e. vehicle and background) are used ($u = p^*$, $t^u = t$, $v = t^*$). The resulting loss is equivalent for two classes.

CHAPTER 4

Experiments

In order to investigate the main differences between the proposed vehicle-focused approach and the conventional unfocused approach, the performance of the vehicle detection models has to be analyzed on the test set. This chapter provides this analysis, as well as more thorough experiments to provide insight into potential reasons for similarities and differences between the approaches. After explaining the measures used for quantification, details about the outcome of the SR step and subsequently of the vehicle detection performance are elaborated.

4.1 Evaluation Metrics

The PSNR is a common[1] metric for evaluating SR approaches and is also used in this thesis to express the quality of a SR prediction, by comparing it with a given ground truth image. Hore and Ziou [HZ10] define the PSNR of a predicted image as follows:

$$PSNR(f,g) = 10\log_{10}\left(\frac{255^2}{MSE(f,g)}\right), \tag{4.1}$$

where

$$MSE(f,g) = \frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}(f_{ij}-g_{ij})^2 \tag{4.2}$$

for two images denoted f and g with each of size $M \times N$, if 255 is the maximum value that a pixel can have and i and j define the position of a pixel. The higher the similarity between the two images (which is desirable for SR), the higher the PSNR [HZ10].

For binary classification (e.g. vehicle detection), there are four basic metrics necessary for calculating the precision and the recall metrics [CJ20]:

[1] The PSNR is used, for example, by Dong et al. [DLHT15], Kim et al. [KLL16a], and Zhang et al. [ZTK+18].

- True positives (TP): The number of correctly classified positives.
- False negatives (FN): The number of positives, incorrectly predicted as negatives.
- True negatives (TN): The number of correctly predicted negatives.
- False positives (FP): The number of negatives, incorrectly interpreted as positives.

In the context of vehicle detection, a *positive* is a vehicle, and a *negative* is background (no vehicle). A predicted vehicle is matched with a ground truth vehicle if the bounding boxes have an IoU of at least 0.3, following Zambanini et al. [ZLP+20]. In the context of object detection, the IoU is calculated by dividing the number of overlapping pixels between the ground truth and the predicted bounding box by the union of those two areas [EVGW+10].

The precision is calculated using the formula [CJ20]

$$precision = \frac{TP}{TP + FP} \tag{4.3}$$

and gives the ratio of correctly identified positives out of all elements classified as positive.

The recall is defined as [CJ20]

$$recall = \frac{TP}{TP + FN} \tag{4.4}$$

and therefore gives the proportion of correctly identified positives to all existing positives.

Note that precision and recall depend heavily on the chosen IoU threshold. A higher threshold can lead to fewer false positives, having a positive influence on the precision metric, while there could be more false negatives and thus a negative impact on the recall. A lower IoU threshold would have the opposite effect. Therefore, metrics like the F_1-Score and the average precision, which are based on a combination of both precision and recall, are more expressive and cannot be manipulated by just modifying the threshold.

The F_1-Score [CJ20] is defined as the following combination of precision and recall:

$$F_1\text{-}Score = 2 \times \frac{precision \times recall}{precision + recall}. \tag{4.5}$$

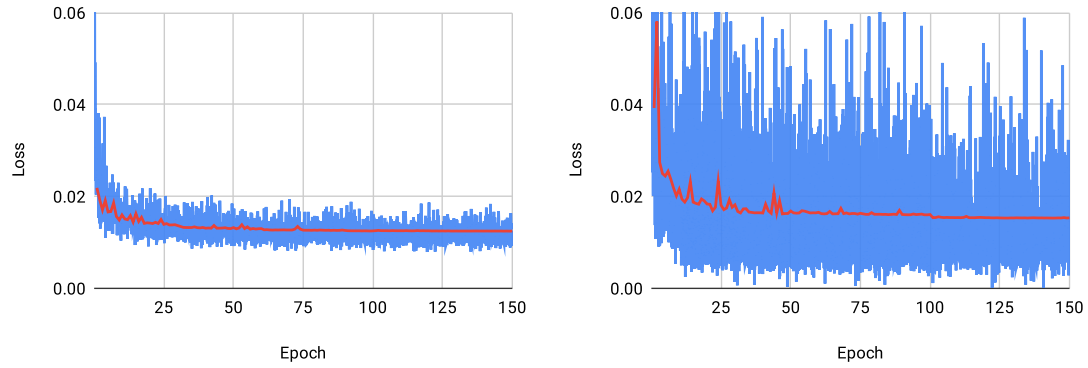
The average precision metric (as described by Henderson and Ferrari [HF16]) is calculated by analyzing the predicted bounding boxes for a given image. First, the predictions are sorted by the confidence the model has in the prediction. Then, the precision and the recall are calculated, only considering the first prediction in this list. These two values are saved as a pair and the process is repeated by additionally considering the second ranked prediction, then the third, and so on. The resulting pairs of precision and recall values can be plotted in a two-dimensional graph. The average precision metric is the area under this curve.

4.2 Super Resolution

The focus of this thesis lies on analyzing the performance differences of vehicle detection, when preprocessing the remote sensing image data with two differently trained SR models (i.e. the vehicle-focused and the unfocused SR). To understand the performance of the vehicle detection, it is also essential to understand the differences of the output of the SR models, which is applied as a preprocessing step.

4.2.1 Training Process

During training, the progress is captured by saving the PSNR, as well as the loss (i.e. the L_1 loss). This happens after each epoch by calculating both these values on the whole validation set and additionally by saving the same metrics after every few batches for the batch used in training. Figure 4.1 shows the difference in convergence during training for both approaches. It can be seen that the loss on the training batch stays much more constant during vehicle-focused training (see Figure 4.1a) than during unfocused training (see Figure 4.1b), which produces values with a high variance, making it impossible to gain any insights about the training progress. As the red lines show, the training progress itself, as calculated on the validation set once per epoch, is much more stable than the training batch analysis suggests. Still, it can be seen that the validation loss is slightly less stable for unfocused training in the beginning (see epochs 25 to 50 in Figure 4.1b).



(a) Vehicle-Focused Training

(b) Unfocused Training

Figure 4.1: The loss during training as calculated on the training batch (in blue), and after each epoch on the validation set (in red).

An explanation for the high variance in the analysis on the training batches is the higher diversity on the unfocused dataset (showing not only vehicles, but also fields, water, forests,...), when compared to the vehicle-focused dataset (depicting exclusively vehicles). As a higher loss in a batch indicates that the network has not learned to predict a good SR representation of the input in the batch yet, the many higher loss values even at the end of the training show how the unfocused network has problems with certain training

images - a phenomenon not observable for vehicle-focused training. Nonetheless, the unfocused training converges too, but at a slightly worse validation loss (i.e. unfocused training: 0.0153, vehicle-focused training: 0.0124). It is not possible to conclude a generally lower performance of the unfocused approach from this validation loss data, as the validation sets differ (i.e. for the vehicle-focused training the validation set is vehicle-focused too, for the unfocused training not).

4.2.2 Pre-Training

To save computational time and thus costs and energy by faster convergence (as shown by He et al. [HGD19]), a pre-trained model[2] is used. This model is already trained for SR, but on the dataset DIV2K [AT17], not remote sensing imagery. For comparison, the network is trained with the same parameters from scratch (i.e. without pre-training). The comparison of the PSNRs on the validation set during the training are shown in Figure 4.2. Surprisingly, the network performs better when trained from scratch. It surpasses the pre-trained model's best PSNR value (achieved after 201 epochs in addition to the pre-trained 85 epochs) after only 30 epochs and is better by 0.80 dB at the end of training.

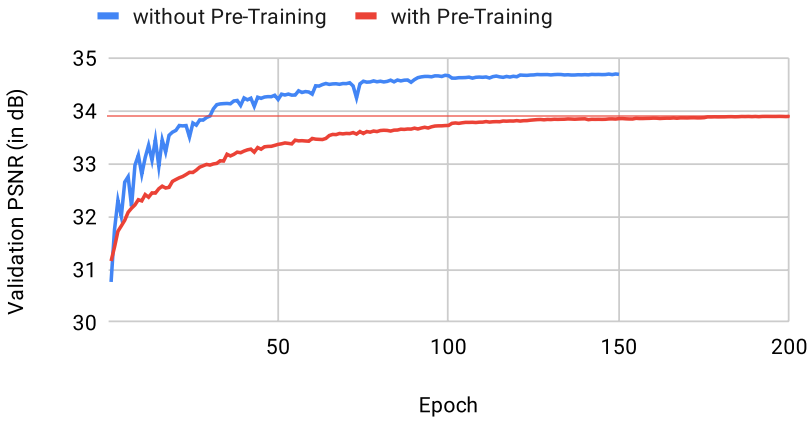


Figure 4.2: The validation PSNR with 85 epochs of pre-training (in red) and no pre-training (in blue). The thin red line shows the best PSNR achieved with pre-training.

A possible interpretation of these results is that the pre-trained model is captured in a local minimum, which may be better suited for general SR tasks, but not for the specifics of remote sensing SR or this specific dataset. A higher learning rate could help avoid that scenario. This possibility is not verified as there is no necessity: The model trained from scratch instantly performs significantly better than the PSNR of 29.78 dB[3] achieved by

[2] https://github.com/idealo/image-super-resolution/tree/master/weights/sample_weights/rdn-C6-D20-G64-G064-x2/PSNR-driven, last retrieved on 2021-03-14

[3] For comparison: On this dataset (i.e. the validation set of the vehicle-focused dataset) bicubic scaling achieves a PSNR of 28.05 dB.

the pre-trained weights (i.e. without any fine-tuning). As a consequence, all SR models in this thesis are trained without pre-training, unless noted otherwise.

Layer Freezing

According to He et al. [HGD19], it is "common practice" to freeze several layers at the beginning of the network when fine-tuning pre-trained networks. The idea is to not change the weights of selected layers during training as they only learn low-level features (e.g. edges), which stay the same for different tasks [HGD19]. The previous experiment shows that pre-training has no positive effect on the training in this thesis and this experiment analyses if layer freezing can alter this conclusion. This technique is evaluated for the pre-trained SR algorithm with two ratios: with 33% and with 75%[4] of the layers frozen. The number of layers reused in this experiment is higher than the "few" layers mentioned by He et al. [HGD19] because the tasks of general SR and remote sensing imagery SR are very similar as both are SR, thus more layers can be reused.

Figure 4.3 shows the results of this experiment, where it can be seen that freezing 33% or 75% of the layers does not improve the PSNR on the validation set. The training process with no freezing performs best for 6 out of 10 epochs, including the last 5 epochs. In the remaining 4 epochs it is not more than 0.039 dB worse than one of the other approaches.

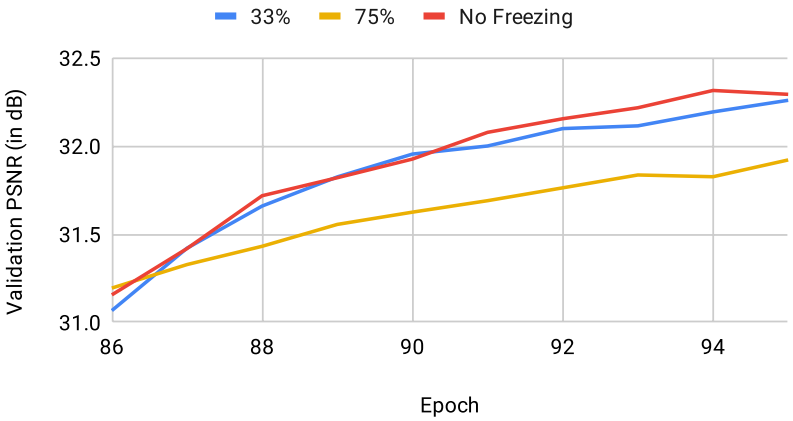


Figure 4.3: The validation PSNR with 33%, 75%, and no frozen layers.

This shows that layer freezing likely does not improve the SR performance. Therefore this modified pre-trained model does not produce better results than training from scratch (i.e. no pre-training at all, neither with frozen layers nor without). It cannot be guaranteed from the data that no different ratio of frozen layers than 33% and 75% produces better

[4]This translates to 137 and 311 frozen layers out of 414 layers in total. Note that in those numbers as provided by the framework, not only convolutional and fully connected layers count as layers, but also activation functions, concatenations, additions (for residual learning), the input layer, etc.

results, but due to resource and time constraints, testing more ratios does not seem reasonable.

4.2.3 Learning Rate Evaluation

Three different learning rates are examined to optimize training. This experiment is conducted for vehicle-focused training only, but the results are likely also valid for unfocused training, since only the dataset changes, while all the model parameters (e.g. D, G_0, and C) stay the same. In order to save computational expenses, the experiment only comprises eleven epochs and three different learning rates, as a full training run would amount to around a full day each (see Section 4.2.6).

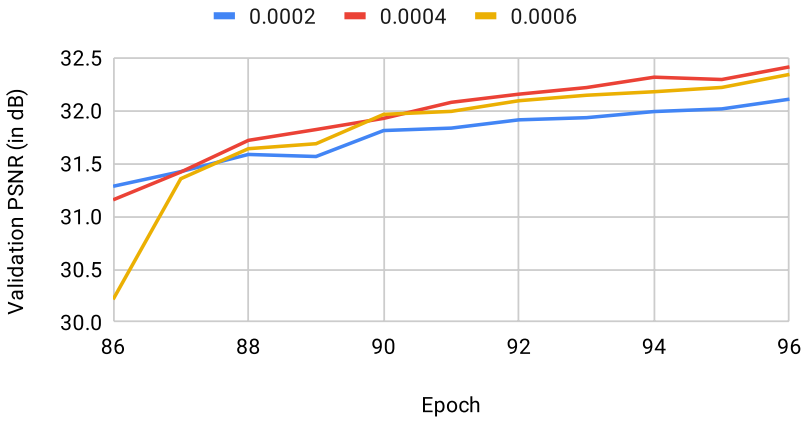


Figure 4.4: Learning rate evaluation

The results for the learning rates 0.0002, 0.0004, and 0.0006[5] can be seen in Figure 4.4. Note that the x-axis starts at epoch 86, because a model[6] pre-trained for 85 epochs is used[7]. The PSNR on the validation set is similar for all three reviewed learning rates, but 0.0004 performs slightly better than the others for 8 out of 11 epochs. The fact that the middle learning rate performs better than the others could indicate that all higher and lower learning rates are worse. This would imply that 0.0004 is near the optimum, however the data is not sufficient to prove that. It can be seen that the difference in the resulting PSNRs for the three learning rates is only marginal (i.e. the average difference in PSNR is only 0.31 dB and every learning rate is best for at least one epoch) and seems to be of minor influence to the result.

[5]Note that the actual learning rates are lower for those epochs, since 0.0002, 0.0004, and 0.0006 are the learning rates for the first epoch and learning rate decay reduces the learning rates by half every 30 epochs.

[6]`https://github.com/idealo/image-super-resolution/tree/master/weights/sample_weights/rdn-C6-D20-G64-G064-x2/PSNR-driven`, last retrieved on 2021-03-14

[7]The experiment starts on a pre-trained model, since the experiment showing the superior performance of training from scratch was conducted at a later point in time.

	Vehicle-Focused Training	Unfocused Training
Vehicle-Focused Dataset	33.28 dB	31.42 dB
Unfocused Dataset	33.77 dB	33.79 dB

Table 4.1: PSNRs on the test sets of the unfocused dataset and the vehicle-focused dataset.

4.2.4 PSNR Comparison

To quantify the general performance of both SR approaches (i.e. vehicle-focused and unfocused training), the final models after 150 epochs of training are used to predict SR versions of the LR images. This is done for the LR images in the test sets of both the unfocused and the vehicle-focused dataset. These predicted images are then compared with the ground truth HR images to calculate a PSNR value.

For the vehicle-focused dataset, the PSNR can be calculated as the average of the PSNRs of all images in the dataset. For the unfocused dataset, two additional definitions are necessary: Firstly, due to the varying image sizes in the dataset, the PSNRs of the images is weighted by the image size, as the PSNR of a 128×128 pixel image[8] should not contribute equally to the PSNR of the whole dataset as an image with a resolution of $9\,200 \times 9\,037$ pixels[9,10].

Secondly, the images are of arbitrary size, which causes a problem: If an image possesses an odd size in one or more dimensions (e.g. a width of 101 pixels), creating a LR version of that image results in a rounded size (e.g. an image with a width of 50 pixels). Applying SR to that image does not yield an image with the same resolution as the original image (e.g. the SR version would have a width of 100 instead of 101 pixels). To make the image comparable nonetheless, the additional pixel column or row is ignored for the calculation of the PSNR.

Table 4.1 shows the outcome of this analysis. It can be seen that for the unfocused dataset, both approaches perform similarly (i.e. the unfocused approach is 0.014 dB better), but for the vehicle-focused dataset the difference is substantial (i.e. 1.867 dB). This indicates that both approaches are able to increase the resolution of satellite imagery similarly, but the area directly around vehicles is reproduced better by the vehicle-focused training approach.

As both approaches - even the vehicle-focused one, which was not trained on it - achieve better results on it, it can be seen that the unfocused dataset seems to be easier than the vehicle-focused set. This can be explained by the nature of the two datasets. While the unfocused dataset can contain vast empty spaces where homogeneous colors (e.g. fields, water,...) dominate the image, those are never the focus of an image of the

[8] 128×128 pixels is the smallest resolution appearing in the unfocused dataset.

[9] The largest image in the unfocused dataset has a resolution of $9\,200 \times 9\,037$ pixels.

[10] In this example the smaller image would have a weight of 16 384 and the larger image 83 140 400, as it depicts an area 5 074.49 times as large.

vehicle-focused dataset, where always at least one vehicle is contained, thus making the image inhomogeneous. Large homogeneous areas are easier to predict with a low PSNR because with low contrast, the predicted pixels are usually of a very similar color. This is empirically shown by Figure 4.5, where the image with the highest PSNR (Figure 4.5a) in the dataset is filled with a similar color and little contrast, except for three small vehicles. Only the area around the cars would be contained in the vehicle-focused dataset, leaving out the easy to predict homogeneous areas of that image. The image with the lowest PSNR (Figure 4.5b) contains much more complexity, contrast, and cars. There are no large low-contrast areas that would be easier to predict and thus the PSNR is lower.

(a) PSNR: 45.35 dB, image source: OIRDS [TCP+09]

(b) PSNR: 22.06 dB, image source: DOTA [XBD+18]

Figure 4.5: The images with the highest and lowest PSNR for unfocused SR in the unfocused dataset test subset.

4.2.5 Vehicle Reconstruction

As previously mentioned, the vehicle-focused approach achieves a higher PSNR on the vehicle-focused dataset than the unfocused approach, which hints to a superior reconstruction quality around vehicles. In order to further analyze whether the vehicle-focused approach is, in fact, able to reconstruct vehicles better, the following experiment is conducted. Taking the test subset of the vehicle-focused dataset with its vehicle-centered 64×64 pixel crops as a starting point, the size of the crops is reduced step by step (every step decreasing the size of the image by 4 pixels per side) until only 4×4 pixels are left – not even enough to cover a vehicle. For every version of the test subset generated like this (64×64, 60×60,..., 8×8, 4×4), the PSNR is calculated for both approaches. This provides the possibility of concluding which approach is better able to reconstruct vehicles: the smaller the size of the vehicle-centered crop, the larger the fraction that

is covered by a vehicle. Therefore, a higher PSNR on smaller vehicle-centered crops represents a better capability to reconstruct vehicles.

Figure 4.6 shows the results of this experiment. The blue line depicts the difference in PSNR between the two approaches, while the x-axis gives the crop-size (per dimension). The difference is calculated by subtracting the PSNR of the unfocused approach from the PSNR of the vehicle-focused approach. Therefore, the positive numbers (ranging between 1.86 and 2.70) show that the vehicle-focused approach produces higher PSNRs on all crop sizes. The difference is higher for smaller crop sizes, showing that the vehicle-focused approach is indeed superior in comparison to the unfocused approach, in terms of the area directly around vehicles. The nearer a pixel is to the vehicle, the higher[11] the difference in reconstruction quality is. This confirms the superior reconstruction quality around vehicles.

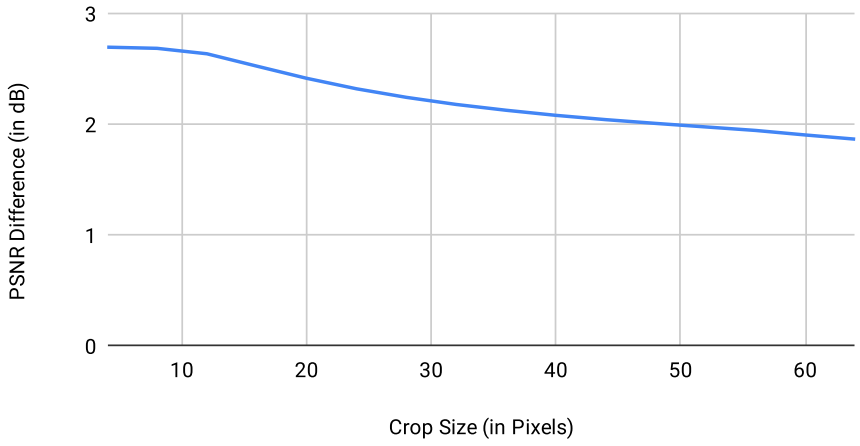


Figure 4.6: The difference in the PSNR between the two approaches, depending on the crop size of the test set.

4.2.6 Training Duration

The time needed for training an epoch of the SR model differs between the two approaches (i.e. vehicle-focused SR and unfocused SR). The training time per epoch for vehicle-focused SR is 533.7 seconds on average (standard deviation: 6.2 seconds). For the unfocused training, the mean is 612.8 seconds (standard deviation 9.7 seconds), which is 14.8% longer. For 150 epochs this equals approximately 22 hours 14 minutes for the vehicle-focused and 25 hours 32 minutes for the unfocused approach.

The only difference in training the two approaches is in the usage of two different datasets (i.e. vehicle-focused dataset and unfocused dataset). The vehicle-focused dataset contains

[11] The difference is even strictly monotonic.

only images of size 64×64 pixels, while the unfocused dataset contains images of varying sizes, although the training requires a uniformly sized input[12]. This causes an extra step during training for the unfocused approach: each mini-batch has to be created by extracting random 64×64 patches from an image of the training set. This is more time-intensive than having to load several images of the correct size. The maximum size of an image in the unfocused dataset is $9\,200 \times 9\,037$ pixels, so (assuming no intersection of image patches) for loading a mini-batch of 8 crops with size 64×64 pixels, up to 99.96% of the pixels are unused. This inefficient process to create a mini-batch causes a performance loss, even though the whole dataset is loaded into RAM before training.

4.2.7 Training Convergence Speed

As discussed in the preceding paragraph, the vehicle-focused approach needs less time to train per epoch. This does not translate to a real advantage so long as it remains unclear as to whether or not more epochs are needed for convergence with this approach. In this thesis, both approaches are trained for 150 epochs, which is the point at which absolutely no improvements of the validation PSNR (or the loss) can be observed anymore. This is likely longer than necessary[13] and therefore this experiment aims at giving an answer to the question regarding which approach trains faster. In order to achieve this, the saved model files from several points in the progress of training (i.e. after the epochs 2, 4, 6, 10,...) are used to predict SR versions of the images in the test sets and calculate their PSNR. This is done for both approaches on both datasets (i.e. vehicle-focused and unfocused) for a limited number of epochs (as this takes 146 minutes per analyzed epoch on the VSC-3).

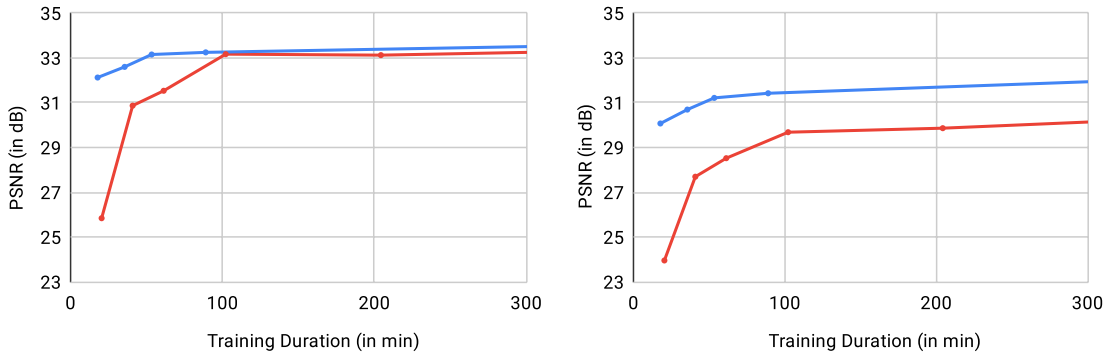
The results of this can be seen in Figure 4.7, where the x-axis gives the training time instead of the epoch, in order to accommodate for the different time needed per epoch. On the unfocused dataset (Figure 4.7a), it can be seen how the vehicle-focused approach provides higher PSNR values earlier (i.e. a PSNR of 33.15 db after only 53 minutes, which is close to the final 33.77 db reached after 1 334 minutes and a similar value is reached by the unfocused approach after around 100 minutes). Both approaches show a similar development after around 100 minutes, slowly approaching their maximum.

On the vehicle-focused dataset (Figure 4.7b), a similar behavior is shown: the vehicle-focused dataset converges quicker and provides higher PSNR values earlier. Additionally, the vehicle-focused approach performs better in general and is never matched by the unfocused approach.

This analysis shows that the vehicle-focused approach provides better results earlier and is therefore faster to train. This is especially important if there are limited training

[12] Note that even though the datasets do not have the same number of images, the duration of an epoch is defined the same way for both approaches: as 1 000 batches of 8 images with the size 64×64 pixels. Thus, the training set size does not have a direct influence on the training duration.

[13] The PSNR on the validation increases only from 33.72 dB to 33.75 dB between epochs 100 and 150, indicating that shorter training would be sufficient.



(a) Unfocused Dataset

(b) Vehicle-Focused Dataset

Figure 4.7: Test set PSNR calculated on both datasets at different points during the training process for the vehicle-focused (blue lines) and the unfocused (red lines) approach. Dots indicate epochs after which the PSNR is calculated.

resources available, as the unfocused approach can provide similar results (at least for the unfocused dataset), but is slower in doing so.

4.2.8 Visual Comparison

For a visual impression of the results of the SR step, refer to Figure 4.8, where the LR input image is compared to both SR approaches for three examples. It is hard to see any substantial differences between the SR results. The third row is an exception to that, as the white lines in the left half of the image are not correctly interpreted by the unfocused approach, which explains why this image has the largest PSNR difference between the two approaches. Also note how the vehicle-focused approach makes the windshield of the red car wider in the second row.

4.2.9 Summary of Super Resolution Results

When analyzing the results of both SR approaches, it can be inferred that both perform similarly well. While no substantial differences can be detected with the naked eye, numbers (i.e. PSNR analysis) suggest that both strategies for training a SR network succeed in producing a model capable of up-sampling remote sensing imagery, but the vehicle-focused approach is better at creating SR versions of vehicles. Another important finding is the faster training convergence of the vehicle-focused approach: The model produces significantly more accurate (i.e. with a higher PSNR) in the first 100 minutes of training.

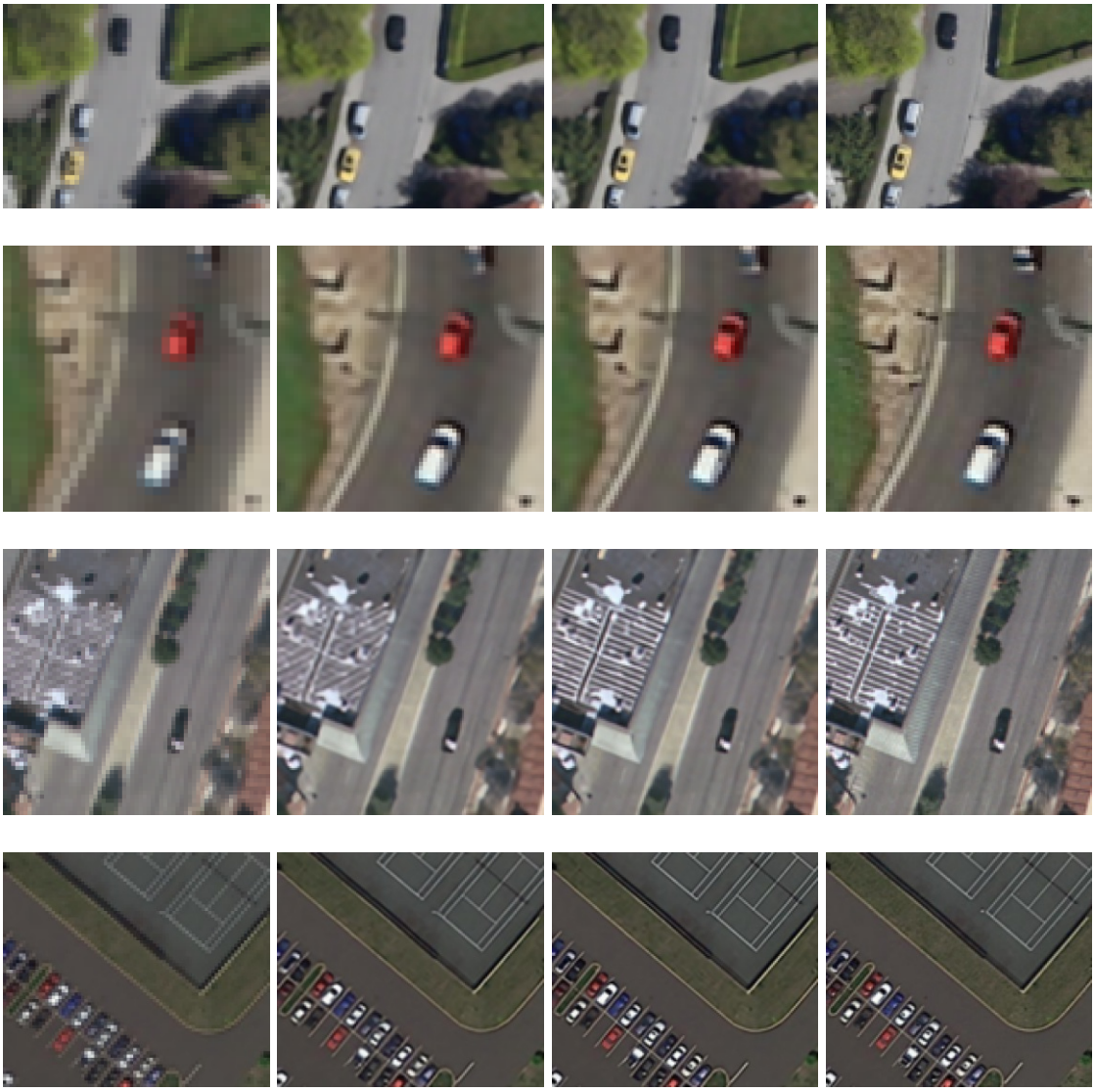


Figure 4.8: Examples for SR images. PSNRs (in dB) from top to bottom (unfocused, vehicle-focused): (23.00, 22.45), (30.88, 32.23), (26.20, 30.80), (29.17, 28.72). Original image sources: DLR-MVDA [LM15] (first row), DOTA [XBD+18] (second and fourth row), OIRDS [TCP+09] (third row)

4.3 Vehicle Detection

An in-depth analysis of the vehicle detection model is the key to answer the research question of this thesis. To see if vehicle-focused SR is a suitable approach to enhance the results of a subsequent vehicle detection, the deciding part is the performance of the

	Vehicle-Focused	Unfocused
F_1-Score	0.7421	0.7350
Precision	0.8000	0.8027
Recall	0.6920	0.6778
Average Precision	0.6721	0.6892

Table 4.2: Vehicle detection results on the respective test sets.

vehicle detection model.

4.3.1 Vehicle-Focused and Unfocused

The central research question of this thesis (i.e. whether or not vehicle-focused SR is superior to unfocused SR as a preprocessing step for vehicle detection) can be answered by analyzing the performance of the vehicle detection algorithm on data generated by both SR approaches. By applying both trained models (i.e. the vehicle-focused and the unfocused vehicle detectors) to the test subset of the vehicle detection dataset[14], predictions of vehicle bounding boxes are generated. From the generated bounding boxes and the ground truth, the metrics F_1-Score, precision, recall, and average precision are obtained.

Those four metrics can be seen in Table 4.2. While the vehicle-focused approach results in a slightly (i.e. 0.0071 and 0.017) higher F_1-Score and recall, the unfocused approach produces a minimally (i.e. 0.0027 and 0.0171) higher precision and average precision. Therefore, when considering all four metrics, neither of the approaches is distinctively superior to the other.

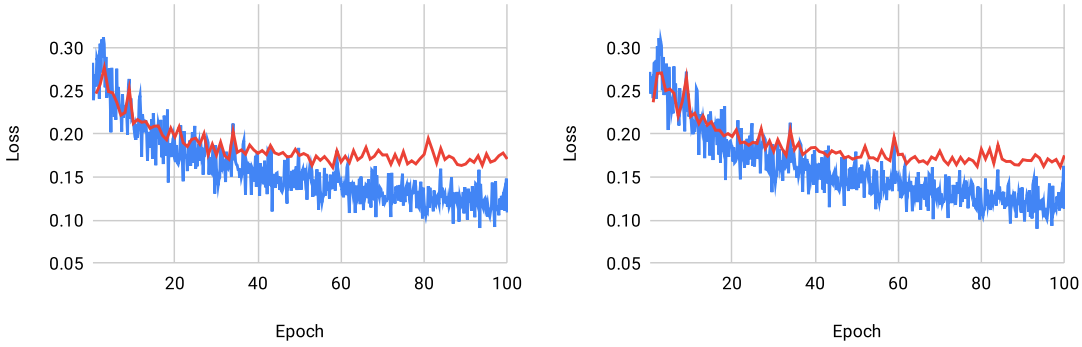
The lack of a distinct difference in performance between both approaches can be explained by the similarity of the output of the two SR models (see Section 4.2). The vehicle detection models presumably perform similarly because the only difference between them is the underlying datasets, which are also similar (see analysis of Section 4.2).

4.3.2 Training Process

After each training epoch, the validation set is predicted to be able to track progress and convergence. The four parts of the loss (the losses for the objectness score and bounding box regression for each of the two parts of the network), as well as the loss itself (i.e. the combination of those four parts), are calculated for those predictions and saved. The same metrics are calculated after each batch during the epochs for the training batches used. The average of those values is saved every five epochs.

[14]The models are not only trained on the vehicle detection dataset as generated by the vehicle-focused and the unfocused SR respectively, but they are also tested on the same version (but the test subset) of that dataset.

Figure 4.9 shows the training and validation loss of both approaches (i.e. two identical models, one trained on data as predicted from the vehicle-focused SR approach, and one trained on data of the unfocused SR). It can be seen that for both models the validation losses converge and no overfitting occurs.



(a) Vehicle-Focused Training

(b) Unfocused Training

Figure 4.9: The loss during training calculated on the training batch (in blue), and after each epoch on the validation set (in red).

The loss for training consists of two terms: One for the RPN and one for the Fast R-CNN part of the model. Each term consists of a part for bounding box regression and a part for classification (for details refer to Equation 3.1 and Equation 3.4). The sum of these four losses (if calculated on the validation set) is considered to be the validation loss. Figure 4.10 depicts how each of them converges. It can be seen that the losses of the RPN part are generally lower, and each bounding box loss is lower than its corresponding classification loss. All of the four losses converge and thus do not overfit.

Figure 4.11 allows a direct comparison of the validation losses during training of both approaches. No distinct differences can be seen between both approaches.

From the training process, no valid conclusions about the quality differences of the models can be drawn. For that, other experiments (i.e. on the test set) provide insight.

4.3.3 Data Augmentation

By default, the implementation used originally supports data augmentation only by flipping images horizontally. This makes sense for images depicting landscapes (where the sky has to be always on the top side) or portraits (where an upside-down face would make no sense), but for remote sensing imagery this assumption does not apply: A vertical flip produces a valid image. The same is valid for rotation. Thus, the data augmentation implementation is expanded by rotation and vertical flipping. Without this full data augmentation, the algorithm has problems converging. Figure 4.12 shows a training run without the expanded data augmentation (in blue), with signs of overfitting and a

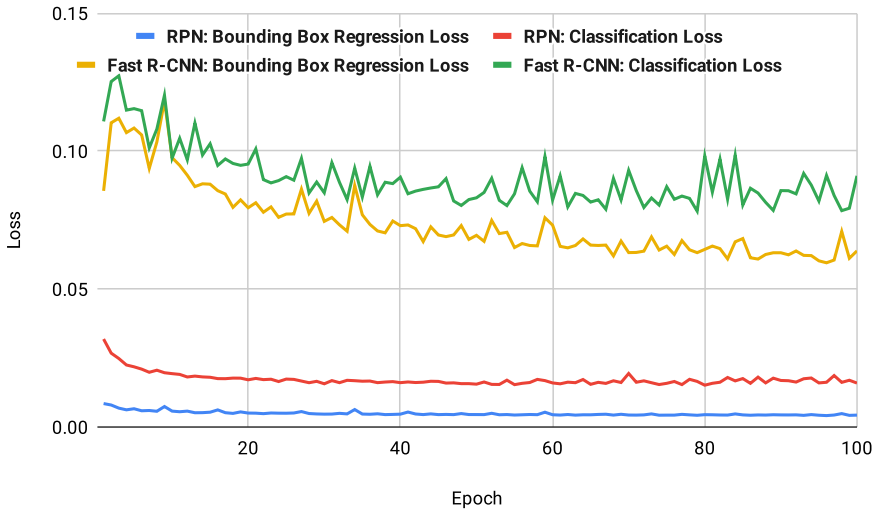


Figure 4.10: The different parts of the validation loss for a vehicle-focused training run.

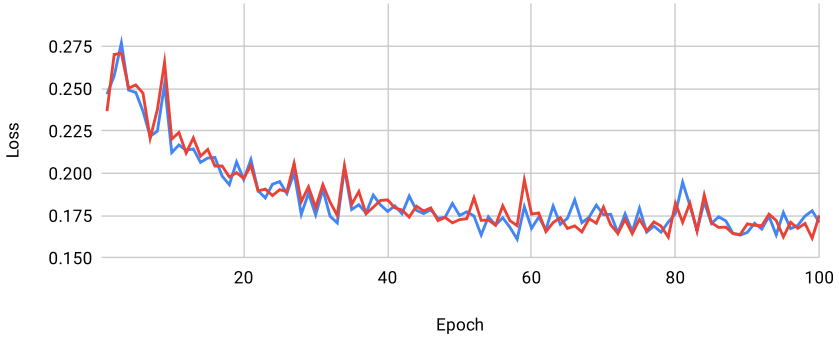


Figure 4.11: Validation loss during training. Vehicle-focused approach: blue, unfocused approach: red.

non-converging validation loss. While the training loss continues to fall, the validation loss reaches its minimum at epoch 16 and then rises again. This problem, possibly caused by overfitting, disappears when using full data augmentation, as is shown by the red lines, which continue to decline.

4.3.4 Learning Rate Decay

The effect of learning rate decay is analyzed by comparing the validation losses during the training of two models: one with learning rate decay and one with a constant learning rate. For this experiment, both models start with a learning rate of 0.004. While the model without learning rate decay stays at that learning rate, the model with learning

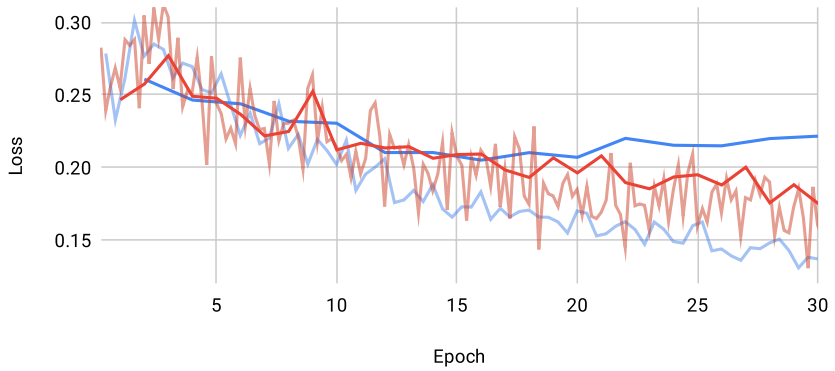


Figure 4.12: Validation (dark blue) and training loss (light blue) for a training run without full data augmentation (i.e. only horizontal flipping). For comparison: the validation (dark red) and training (light red) losses of a training run with full data augmentation.

rate decay multiplies the learning rate by 0.5 every 25 epochs (i.e. from epoch 26 the learning rate is 0.002 and beginning with epoch 51 it is 0.001).

In Figure 4.13 the validation losses of both models can be seen and it is clear that the difference between them is marginal. Nonetheless, the model without learning rate decay performs slightly better: In total it is 62 out of 90 epochs better and on average the loss is 2.05% lower than with learning rate decay. In the epochs where the difference in the learning rate is larger, the difference in the validation loss increases: for the epochs 51 to 75 (learning rate because of decay: 0.001) the difference is already 2.37% and for the epochs 76 to 90 (learning rate because of decay: 0.0005) it is 3.43%.

This experiment indicates that learning rate decay does not lead to an improved performance in this particular case. Although only one learning rate decay configuration is analyzed, it is highly unlikely that a different interval for changing the learning rate would provide better results. A higher interval would lead to smaller learning rates faster, but the experiment indicates that smaller learning rates do not benefit this model, so this is not a promising direction. On the other hand, a slower interval would likely cause no significant change either, because the tested configuration already changes the learning rate only three times. Reducing the number of changes would only minimize the difference between the two configurations. The experiment already demonstrates that the model does not perform better with the learning rate decayed to 0.002 and 0.001, thus it would be counterintuitive to keep this state longer by increasing the decay interval.

Similarly, it is possible to argue against a higher or lower decay factor than 0.5. On one hand it is already shown that only slightly lower learning rates (e.g. 0.002 for the epochs 26 to 50) do not appear to improve the results. On the other hand, even lower learning rates (e.g. 0.0005 for the epochs 76 to 90) also produce worse validation losses. As a

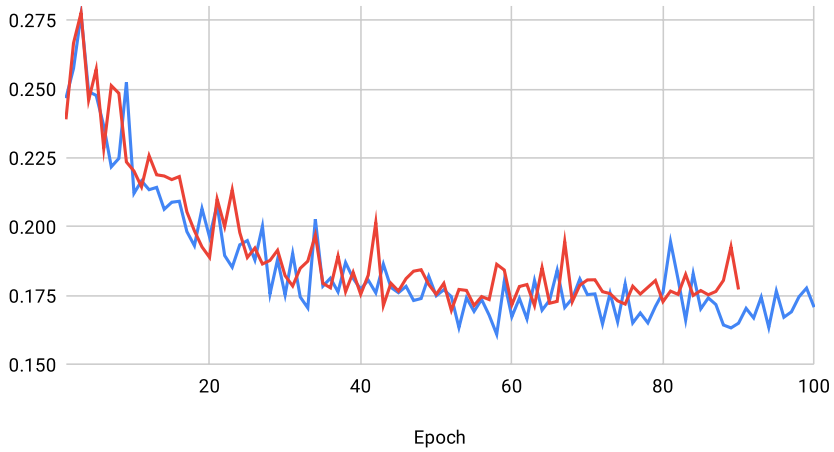


Figure 4.13: Validation loss for training with learning rate decay (in red) and with constant learning rate (in blue). Both are trained on vehicle-focused data.

result, no learning rate decay is used for training of vehicle detection models.

4.3.5 Comparison with HR and LR

The influence of the SR models on vehicle detection can be further understood when seen in relation to two more training sets. The ideal output of a SR model would equal the ground truth HR data. Therefore, when training and testing a vehicle detection algorithm, the results represent the upper bound of performance achievable by improving SR. The other extreme is to perform no SR at all and instead use the LR data, producing the lower bound of the performance.

See Figure 4.14 for the comparison of four metrics for these two additional training sets and the two main approaches (vehicle detection on vehicle-focused and unfocused SR data). As expected, all metrics of the SR-based approaches lie in-between the LR- and HR-based approaches, with the exception of the recall, which is higher for the vehicle-focused vehicle-detection than for the HR-based model. In all metrics, both SR-based approaches are nearer to the HR-based than to the LR-based approach[15].

The fact that the results of both SR-based approaches are close to the HR approach indicates that improvements of the SR preprocessing step do not have a substantial potential to have a positive effect on the vehicle detection. Instead, the vehicle detection step itself could be a target of improvement.

[15]On average (i.e. averaged over all four metrics), the unfocused approach is 1.24% worse than the model trained and tested on HR data and 5.16% better than the LR model. The vehicle-focused approach is 1.19% below the HR approach and 5.22% above the LR approach

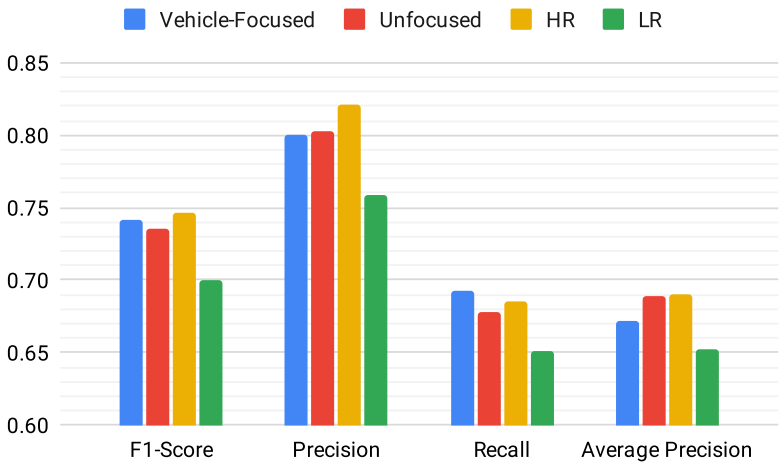


Figure 4.14: Comparison of the vehicle detection results trained on SR, LR, and HR data.

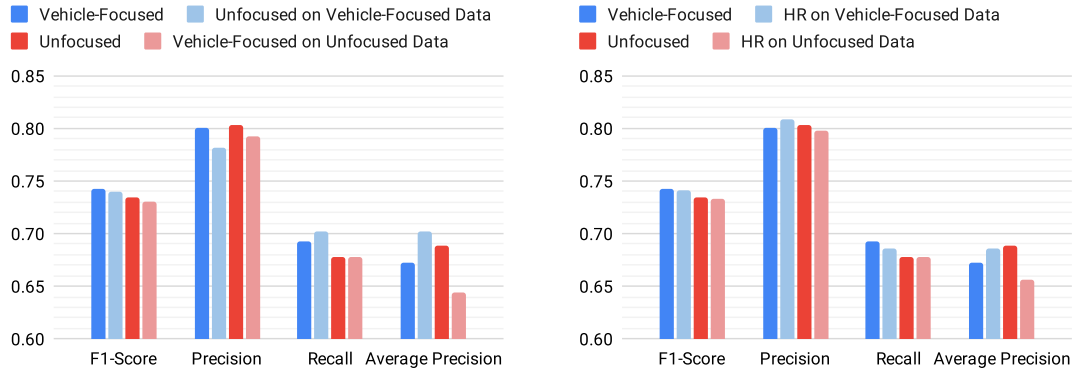
This experiment confirms the results of Shermeyer and Van Etten [SVE19] and Ferdous et al. [FMN19], who observe improved vehicle detection results when using SR as a pre-processing step. This matches with the LR-based model, which shows inferior detection performance in this experiment.

4.3.6 Cross-Dataset Vehicle Detection Evaluation

How much does the vehicle detection model adapt to the slight differences between the two SR-generated datasets? In order to answer this question, the model trained on the vehicle-focused vehicle detection dataset is tested on the unfocused one, and vice-versa. Additionally, this inter-dataset testing is also done on HR data and with the vehicle detection model trained on HR data. This makes answering another question possible: How should the model be trained for optimal vehicle detection results, in a scenario where only unseen LR data is available at inference time, but during training time, HR data would be available too? Should the vehicle detection model be trained on HR data or on a SR version of the data, because during inference it will be applied to SR data?

Figure 4.15a is the basis for answering the first question. It shows a comparison of the results on the vehicle-focused version of the vehicle detection dataset (in blue) between the model trained on that same version of the dataset (dark blue) and trained on the unfocused version (light blue). As it can be seen, the first model produces a better F_1-Score (and precision), but the second one is superior at average precision (and recall). The diagram also shows a comparison on the unfocused version of the dataset, depicting the results of the model trained on the same version (dark red) and the model trained on the vehicle-focused version (light red). The first model performs better in every metric, except recall, where the difference is only 0.0004.

The second question (i.e. if training on HR data would be better) can be answered by the data of Figure 4.15b. It shows (in light blue and light red) the performance of the HR model in comparison to the two models trained specifically on the dataset versions on which the tests are done. The HR model has a higher average precision, but a lower F_1-Score on the vehicle-focused version of the vehicle detection dataset. For the unfocused version of that dataset, all four metrics are lower with the HR model.



(a) Results of the models trained on the vehicle-focused and unfocused versions of the vehicle detection dataset on the same and each others dataset version.

(b) Results of the model trained on HR data for the vehicle-focused and unfocused data.

Figure 4.15: Inter-Dataset Evaluation: Evaluation of models on dataset versions they are not trained on.

Both questions cannot be answered with a general rule. None of the results for the different approaches is substantially better or worse and they vary between the metrics. One observation is that the two main approaches (i.e. training on the two SR versions of the vehicle detection dataset) do generalize and perform similarly well on the respective other version of the dataset. The HR model does not produce substantially different results to the two main approaches and thus is a valid option to train.

4.3.7 Vehicle Detection on pre-trained SR

As already discussed, the SR model is trained from scratch, but in an experiment the training is also done based on a model which is pre-trained on non-remote sensing imagery. The results of the PSNR analysis of the pre-trained model show inferior performance, thus the model trained from scratch is used. In this experiment, the vehicle detection performance when training and testing the model on data predicted by the pre-trained SR model is analyzed.

The four metrics used to assess the performance of those models based on pre-trained SR are depicted in Figure 4.16. The superiority of the SR models without pre-training is

confirmed, as for both approaches (i.e. vehicle-focused and unfocused) all four metrics are lower when using the pre-trained SR.

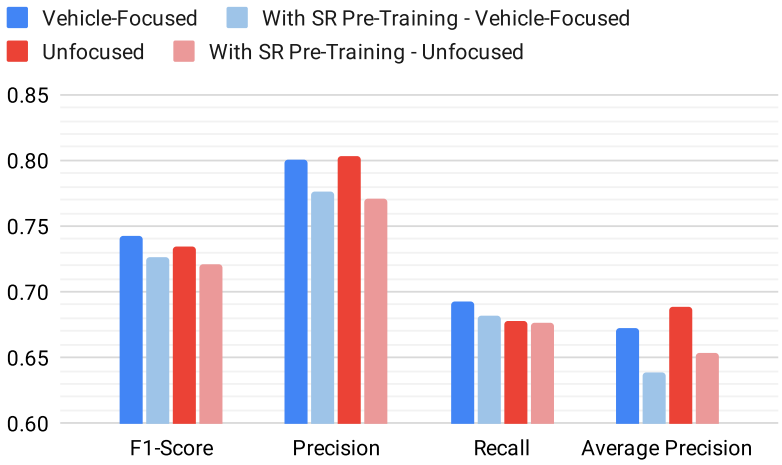


Figure 4.16: Performance of the vehicle detection model when trained on SR data from a pre-trained model.

4.3.8 Results on Subsets and completely unseen Datasets

The datasets used for training and testing the SR model as well as the vehicle detection model consist of six source datasets. One of them (i.e. COWC [MKSB16]) is fully contained in the training datasets and three of them (i.e. VEDAI [RJ16], DLR-MVDA [LM15], and OIRDS [TCP+09]) are only in the test subsets. The remaining two datasets (i.e. DOTA [XBD+18] and xView [LKM+18]) are split into the training (and validation) as well as test subsets. Additionally, the PaCaBa [ZLP+20] dataset, which is not at all contained in the other datasets, serves as a completely unseen dataset for testing.

The results for the vehicle detection prediction on these source datasets are shown in Figure 4.17. One observation is that the vehicle detection generalizes well to unseen datasets. The performances of the models on the three unseen datasets, which are not used for training (i.e. VEDAI, DLR-MVDA, and OIRDS), is better (i.e. F_1-Score and recall are higher) or partially better (i.e. for precision only OIRDS has significantly higher values; for average precision VEDAI and OIRDS have the best values) than on both of the datasets also involved in training (DOTA and xView).

The observation that the generalization to unseen datasets is fully achieved is not valid for the also unseen PaCaBa dataset [ZLP+20]. This dataset seems to be more difficult as it has the lowest F_1-Score, recall, and average precision of all datasets. Only the precision seems to be excellent. This means that a high percentage of vehicles in the dataset are not found, but if the model detects a vehicle, it is usually correct as there are only few false positives.

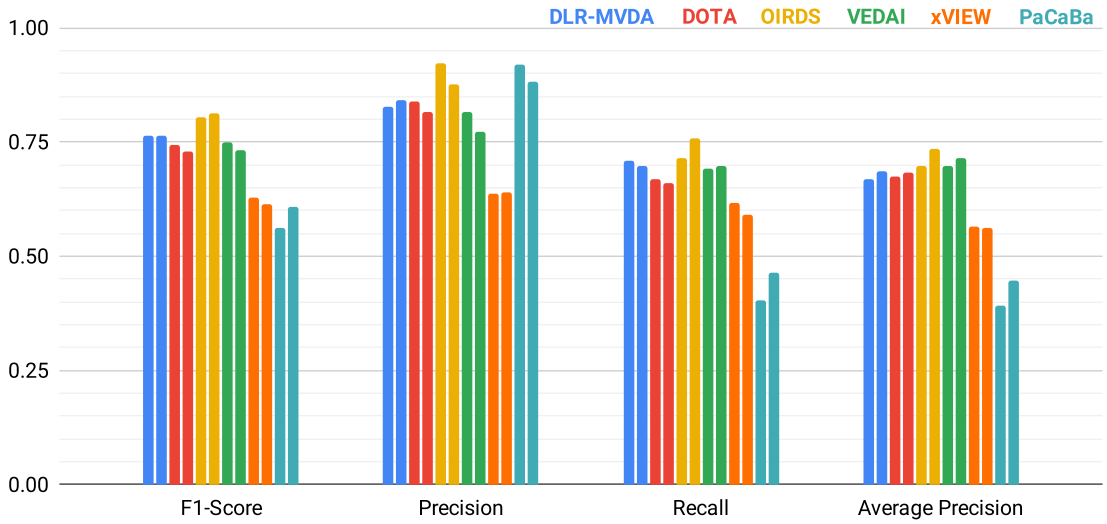


Figure 4.17: Results of the vehicle detection algorithm per source dataset. In the pairwise columns with the same color, the left column is always the result using the vehicle-focused model, on the right is the result of the unfocused model.

When looking into the reasons for the low performance on the PaCaBa dataset, the analysis of the images of this dataset show differences in sharpness. Figure 4.18 is a comparison of a vehicle in PaCaBa with a vehicle in the xView dataset. While the vehicle in Figure 4.18b is as sharp as the GSD of 0.3 m allows, the vehicle from PaCaBa (see Figure 4.18a) appears to be a rounded white blob. One notable difference between those two datasets is the original GSD (i.e. before scaling both to a uniform GSD of 0.3 m), which is 0.34 m for PaCaBa and 0.3 m for xView dataset. This difference is unlikely to cause this significant discrepancy in sharpness because it is relatively small. Another potential reason is that PaCaBa is captured by a satellite from 620 km height [ZLP+20], potentially losing sharpness due to the long distance.

Another possible reason for the lower performance on PaCaBa is the high number of trees, often (as in the example of Figure 4.18a) occluding parts of vehicles and thus making it harder to detect them. When looking at the PaCaBa imagery, the subjective impression is created that the vehicles in the dataset are more difficult to recognize, matching the metrics in Figure 4.17.

Whether or not a part of a dataset is used for training apparently does not heavily influence the results for a dataset, the results may instead be caused by how "difficult" a dataset is. If so, the "easiest" dataset is OIRDS [TCP+09], on which the highest F_1-Score, recall, and average precision are achieved.

Note the variation across datasets if the vehicle-focused or the unfocused model achieve better results for a given metric. For example, the F_1-Score of the vehicle-focused

(a) PaCaBa [ZLP+20] (b) xView [LKM+18]

Figure 4.18: Comparison of vehicle sharpness.

approach is better for DLR-MVDA, DOTA, VEDAI, and xView, but not for the other two datasets. This shows that it can not be said that one approach "has a better precision" (or any other metric) than the other approach. The differences are marginal and thus vary even between datasets.

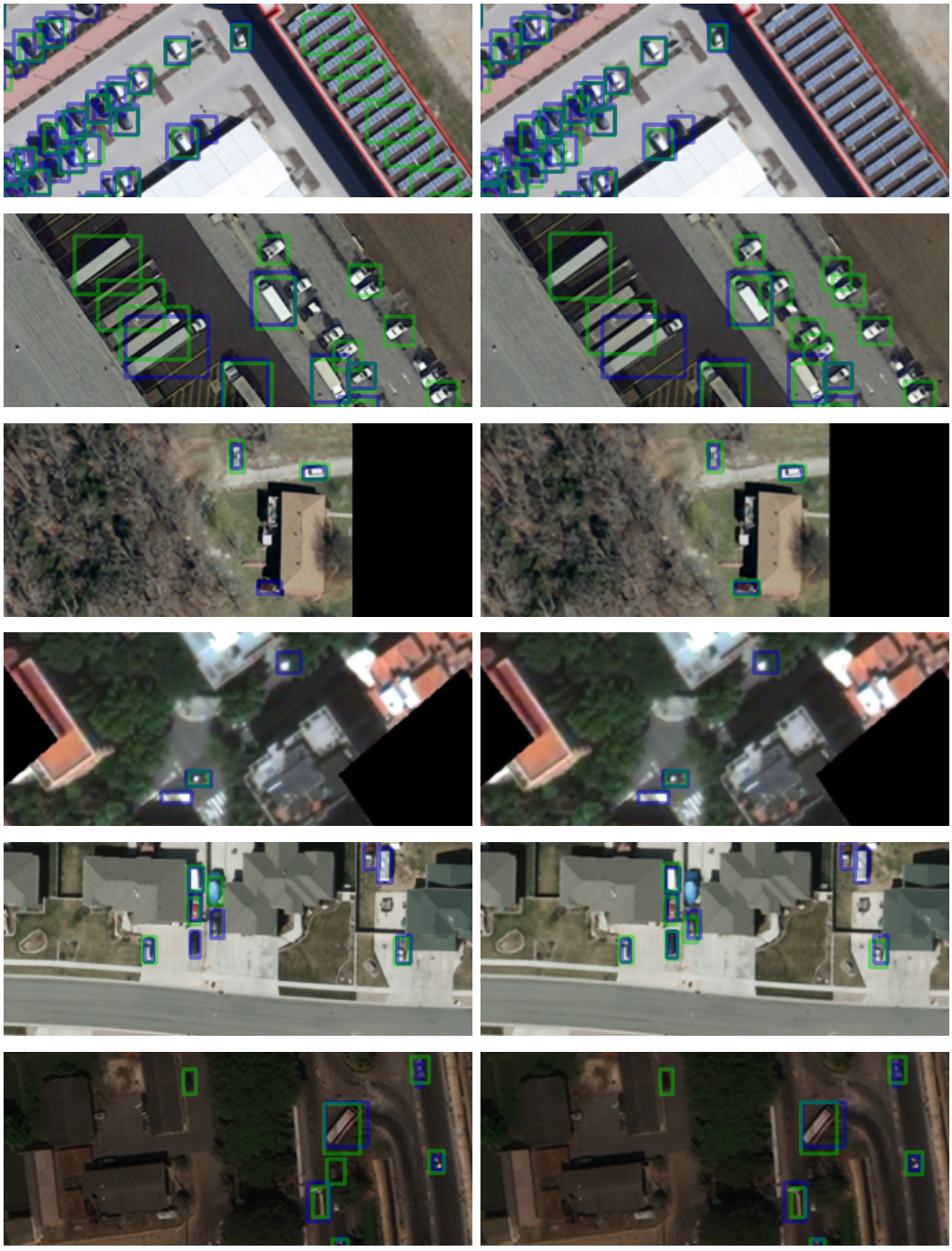
4.3.9 Visual Comparison

In order to get an impression of the vehicle detection performance on the variety of remote sensing imagery used, refer to Figure 4.19. The left column visualizes the bounding boxes (and ground truth) of the vehicle-focused approach and the right column shows the output of the unfocused approach on the same source images. Every row contains an image from a different source dataset. In general it can be seen that both approaches produce mostly similar results.

The first row shows an example where the vehicle-focused approach falsely recognizes rectangular structures[16] as vehicles. The unfocused approach correctly disregards them.

One notably difficult case is shown in the second row. In the left halves of the images, five truck trailers can be seen, but only one is attached to a truck. The ground truth interprets only the complete truck to be a vehicle and the rest is not annotated. Both models fail to match that logic and produce false positives. On the right side in the same image, several more false positives in both approaches can be seen. To the surprise of the author of this thesis, according to the ground truth, the white structures on both sides of the street are not vehicles. However, both approaches interpret some of them as vehicles.

[16]The rectangular structures are probably solar cells.



Vehicle-Focused Unfocused

Figure 4.19: Visualized vehicle detection results. Blue boxes: ground truth, green boxes (lighter green encodes higher certainty): detections. Source datasets in rows from top to bottom: DLR-MVDA [LM15], DOTA [XBD+18], OIRDS [TCP+09], PaCaBa [ZLP+20], VEDAI [RJ16], xView [LKM+18].

4.4 Summary

The key findings of the experiments presented in this chapter are:

- The SR training and validation losses vary more during the training process of the unfocused approach, hindering easy monitoring of the training progress.
- The vehicle-focused SR training is faster and delivers more accurate results in the early phases of the training.
- The SR models achieve better results if trained from scratch, instead of using a pre-trained model. Freezing layers of the pre-trained model does not change this. The vehicle detection performance on imagery processed by pre-trained SR is also worse than without pre-training.
- The SR models achieve a lower PSNR on the vehicle-focused dataset than on the unfocused dataset, indicating that it is more difficult.
- The SR model trained on unfocused data is worse at reconstructing vehicles than the vehicle-focused approach.
- Both vehicle detection models (trained vehicle-focused and unfocused respectively) perform similarly. The F_1-Score is insignificantly higher for the vehicle-focused approach, the average precision for the unfocused approach.
- Using HR data instead of SR data for vehicle detection produces approximately similar results. This indicates that improving the SR models would not benefit vehicle detection any further.
- Vehicle detection is improved by using SR as a preprocessing step.
- Training the vehicle detection on HR data and applying it to SR data is a valid option and produces similar results as training on SR data.
- Training the vehicle detection on unfocused SR data and testing it on vehicle-focused SR data results in approximately the same performance as vice-versa.
- Both vehicle detection approaches generalize without problems to unseen datasets (except PaCaBa [ZLP+20]). They partially produce more accurate predictions on unseen datasets, than on the test sets of datasets which are included in training.
- The PaCaBa dataset is difficult and the models do not achieve good results on it (the average precision is more than 10 percent points lower than for any other dataset).

In summary, the proposed vehicle-focused SR training leads to competitive results while converging faster than training conventionally. Analysis of the SR results implies that

vehicles are reconstructed more accurately with the vehicle-focused approach. This slight advantage does not translate to an increased vehicle detection performance. A possible cause is the quality of both SR approaches, which provide sharp imagery to the vehicle detection approach in either case. Failed detections could be instead attributed to difficult cases in the datasets or a potentially suboptimal vehicle detection model.

CHAPTER 5

Conclusion & Future Work

This thesis proposes a novel strategy to improve on SR-enhanced vehicle detection, by training the SR model in a vehicle-focused manner. This is enabled by creating two separate datasets, for SR training: a vehicle-focused dataset (i.e. consisting of crops centered around vehicles) and a conventional unfocused dataset. By training one SR and subsequently one vehicle detection model for each of the two datasets, the vehicle-focused and the conventional method can be compared. The datasets are created from six source datasets, with an additional seventh dataset used only for testing. SR is performed by the RDN [ZTK+18] architecture and for vehicle detection, the Faster R-CNN [RHGS15] network is applied.

While both approaches deliver similar results after extended SR training times (e.g. after 150 epochs, which take 22.2 hours for vehicle-focused and 25.5 hours for unfocused training), the vehicle-focused method exhibits slightly shorter training times per epoch and, more importantly, earlier convergence. This makes the proposed vehicle-focused approach a superior option to conventional training, especially if short training duration is important, as in cases of strictly limited budget or for large models with high complexity[1].

Although the improvement in speed is a progress on the state of the art, the question remains as to why the vehicle detection performance itself is not benefiting from the vehicle-focused approach. Possible reasons for that and how to eliminate them in future work, are discussed in the following paragraphs.

The performance of the vehicle detection is similar on both versions of the SR imagery. This could be the case because the SR task is too easy and is solved sufficiently well for vehicle detection by both approaches (as demonstrated by the comparison with vehicle detection on HR imagery). Thus, there is no potential to show the superiority of vehicle-focused SR, as unfocused SR is already near the optimum. Future work could

[1] If models take weeks to train (e.g. Liebel and Körner's model [LK16] takes 20 days to train SR), a reduction in training duration by a few percent saves many hours or even days.

investigate vehicle-focused training on more challenging versions of this problem, namely $\times 4$ (e.g. GSD 1.2 m $\Rightarrow$ 0.3 m) or $\times 8$ (e.g. GSD 2.4 m $\Rightarrow$ 0.3 m) SR. This increase in resolution is more difficult and the resulting imagery is less sharp, which results in reduced vehicle detection performance [SVE19]. In those cases, vehicle-focused SR could enable superior vehicle detection performance, as the task is more demanding and it is unlikely that both approaches would again result in SR imagery that is close to ground truth HR imagery.

Further improvements could be achieved by testing more hyperparameters. Finding the right hyperparameters consumes a high quantity of time and resources, as it requires experimentally checking many combinations of them (e.g. by grid search [Nie15]). Only some are checked in this thesis (e.g. learning rate) and thus the performance could be hindered by a suboptimally chosen parameter. Therefore, the models could benefit from finding more suitable parameters. Specifically, experiments with a wider range of learning rates and their decays, different model sizes (e.g. the number of RDBs in the RDN [ZTK+18]), varying batch sizes, and different numbers of RPN proposals (as by Sommer et al. [SSB17]) should be considered to be conducted in future research.

A recent idea in the vehicle detection state of the art, as proposed by Mostofa et al. [MFRN20], is learning a SR and a vehicle detection step jointly in one network. Modifying the training of this joint network to use the vehicle-focused training strategy of this thesis could improve vehicle detection performance. In order to implement this idea, the concept of vehicle-focused learning would have to be slightly changed, as the vehicle detection part of the joint network would not see enough variability in data if the vehicle is always in the center of the training image (which is the case with the vehicle-focused training strategy). This problem could be solved by training with a dataset where one image crop is created for every vehicle, but with the vehicle at a random position in the image instead of being centered.

In this thesis, the anchor size for bounding box proposals is reduced in order to cope with the difference in expected object sizes between general purpose object detection and vehicle detection. Still, Faster R-CNN [RHGS15] may not be optimally suited for the task of vehicle detection, due to the low resolution of the last feature maps [SSSB18]. Several solutions to this problem are proposed in the literature: using a shallower base network [SSB17], additionally using features of earlier (i.e. larger) layers [TZD+17b], and applying a transposed convolution layer to increase the resolution again [SSSB18]. Implementing one of those improvements could increase the general vehicle detection performance in followup work of this thesis. Instead of evaluating modifications to Faster R-CNN [RHGS15], it would also be interesting to see the effect of vehicle-focused SR on other object detection algorithms (e.g. SSD [LAE+16]).

Acronyms

COCO Common Objects in COntext. 15

COWC Cars Overhead With Context. 25, 26, 28, 29, 32, 33, 62

DFF Dense Feature Fusion. 35, 36

DLR-MVDA DLR Multi-class Vehicle Detection and Orientation in Aerial Imagery. 1, 25–27, 29, 33, 54, 62, 64, 65

DOTA Dataset for Object deTection in Aerial images. 24–27, 29, 32–34, 50, 54, 62, 64, 65

DRCN Deeply-Recursive Convolutional Network. 7, 8

DRRN Deep Recursive Residual Network. 7, 8

EDSR Enhanced Deep Super-Resolution. 9, 10

ESPCN Efficient Sub-Pixel Convolutional Neural Network. 9, 36

ESRGAN Enhanced Super-Resolution Generative Adversarial Network. 10

Fast R-CNN Fast Regions with CNN features. 12–14, 16, 38, 39, 41, 56

Faster R-CNN Faster Regions with CNN features. 3, 4, 13–17, 19, 21, 38–41, 69, 70

FPN Feature Pyramid Network. 13, 15

FSRCNN Fast Super-Resolution Convolutional Neural Network. 8

GAN Generative Adversarial Network. 9–11, 18

GSD Ground Sampling Distance. 16, 18, 21, 23, 25, 27–32, 63, 70

HR High Resolution. 3, 6, 7, 9, 49, 59–61, 66, 69, 70

IoU Intersection over Union. 14, 40, 44

LapSRN Laplacian Pyramid Super-Resolution Network. 8, 9

LR Low Resolution. 6–9, 18, 35, 49, 53, 54, 59, 60

mAP mean Average Precision. 18

MISR Multi-Image Super Resolution. 6

OIRDS Overhead Imagery Research Data Set. 25, 26, 28, 29, 31–33, 50, 54, 62, 63, 65

PaCaBa Parking Cars Barcelona. 25, 26, 28, 62–66

PNG Portable Network Graphics. 31

PSNR Peak Signal-to-Noise Ratio. 9, 37, 43, 46–54, 61, 66

R-CNN Regions with CNN features. 12, 14, 16

R-FCN Region-based Fully Convolutional Network. 13

RAM Random-Access Memory. 37, 52

RCAN Residual Channel Attention Network. 10

RDB Residual Dense Block. 10, 35–37, 70

RDN Residual Dense Network. 3, 4, 10, 19, 21, 34–37, 41, 69, 70

ReLU Rectified Linear Unit. 5, 17, 35

RoI Region of Interest. 12, 13, 39, 40

RPN Region Proposal Network. 13, 17, 38–41, 56, 70

SAN Second-order Attention Network. 10

SFENet Shallow Feature Extraction Net. 35

SISR Single Image Super Resolution. 6, 34

SPP-Net Spatial Pyramid Pooling Network. 12

SR Super-Resolution. 1–4, 6–11, 18, 19, 21–23, 28–32, 34, 36, 38, 41, 43, 45–47, 49–56, 59–62, 66, 67, 69, 70

SRCNN Super-Resolution Convolutional Neural Network. 7

SRGAN Super-Resolution Generative Adversarial Network. 9, 10, 18

SRResNet Super-Resolution Residual Network. 9

SSD Single Shot Detector. 15, 18, 70

SVM Support Vector Machine. 12, 18

UAV Unmanned Aerial Vehicle. 16

UPNet Up-Sampling Net. 35, 36

VDSR Very Deep Super Resolution. 7

VEDAI Vehicle Detection in Aerial Imagery. 25, 26, 29, 32–34, 62, 64, 65

VSC-3 Vienna Scientific Cluster 3. 37, 52

XML Extensible Markup Language. 33

xView DIUx xView 2018 Detection Challenge Dataset. 25–27, 29, 31–34, 62–65

YOLO You Only Look Once. 14, 15, 17

Bibliography

[AH17] Hamed Habibi Aghdam and Elnaz Jahani Heravi. *Guide to Convolutional Neural Networks*. Springer, New York, NY, 2017.

[AKB20] Saeed Anwar, Salman Khan, and Nick Barnes. A deep journey into super-resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(3):1–34, 2020.

[AT17] Eirikur Agustsson and Radu Timofte. Ntire 2017 challenge on single image super-resolution: Dataset and study. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 126–135, 2017.

[CA17] Jennifer Carlet and Bernard Abayowa. Fast vehicle detection in aerial imagery. *arXiv preprint arXiv:1709.08666*, 2017.

[CBFAB94] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *Proceedings of 1st International Conference on Image Processing*, volume 2, pages 168–172. IEEE, 1994.

[CH16] Gong Cheng and Junwei Han. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11–28, 2016.

[CJ20] Davide Chicco and Giuseppe Jurman. The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation. *BMC Genomics*, 21(1):1–13, 2020.

[CJWL16] Liujuan Cao, Rongrong Ji, Cheng Wang, and Jonathan Li. Towards domain adaptive vehicle detection in satellite image by supervised super-resolution transfer. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[DCZ+19] Tao Dai, Jianrui Cai, Yongbing Zhang, Shu-Tao Xia, and Lei Zhang. Second-order attention network for single image super-resolution. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11065–11074, 2019.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009.

[DLHS16] Jifeng Dai, Yi Li, Kaiming He, and Jian Sun. R-fcn: Object detection via region-based fully convolutional networks. In *Advances in Neural Information Processing Systems*, pages 379–387, 2016.

[DLHT15] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(2):295–307, 2015.

[DLT16] Chao Dong, Chen Change Loy, and Xiaoou Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision*, pages 391–407. Springer, 2016.

[ESTA14] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154, 2014.

[EVGW+10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010.

[FMN19] Syeda Nyma Ferdous, Moktari Mostofa, and Nasser M Nasrabadi. Super resolution-assisted deep aerial vehicle detection. In *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, volume 11006. International Society for Optics and Photonics, 2019.

[FRC13] Esmaeil Faramarzi, Dinesh Rajan, and Marc P Christensen. Unified blind method for multi-image super-resolution and single/multi-image blur deconvolution. *IEEE Transactions on Image Processing*, 22(6):2101–2114, 2013.

[GBB11] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 315–323. JMLR Workshop and Conference Proceedings, 2011.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[GBH11] Anna Gąszczak, Toby P Breckon, and Jiwan Han. Real-time people and vehicle detection from uav imagery. In *Intelligent Robots and Computer Vision XXVIII: Algorithms and Techniques*, volume 7878, page 78780B. International Society for Optics and Photonics, 2011.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014.

[Ger74] RW Gerchberg. Super-resolution through error energy reduction. *Optica Acta: International Journal of Optics*, 21(9):709–720, 1974.

[Gir15] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[GNB+11] Joshua Gleason, Ara V Nefian, Xavier Bouyssounousse, Terry Fong, and George Bebis. Vehicle detection from aerial imagery. In *2011 IEEE International Conference on Robotics and Automation*, pages 2065–2070. IEEE, 2011.

[GPAM+14] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems-Volume 2*, pages 2672–2680, 2014.

[GWK+18] Jiuxiang Gu, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, Xingxing Wang, Gang Wang, Jianfei Cai, et al. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[HF16] Paul Henderson and Vittorio Ferrari. End-to-end training of object class detectors for mean average precision. In *Asian Conference on Computer Vision*, pages 198–213. Springer, 2016.

[HGD19] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4927, 2019.

[HXX+18] Xiaowei Hu, Xuemiao Xu, Yongjie Xiao, Hao Chen, Shengfeng He, Jing Qin, and Pheng-Ann Heng. Sinet: A scale-insensitive convolutional neural network for fast vehicle detection. *IEEE Transactions on Intelligent Transportation Systems*, 20(3):1010–1019, 2018.

[HZ10] Alain Hore and Djemel Ziou. Image quality metrics: Psnr vs. ssim. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369. IEEE, 2010.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015.

[JWY+19] Kui Jiang, Zhongyuan Wang, Peng Yi, Guangcheng Wang, Tao Lu, and Junjun Jiang. Edge-enhanced gan for remote sensing image superresolution. *IEEE Transactions on Geoscience and Remote Sensing*, 57(8):5799–5812, 2019.

[JZL+19] Licheng Jiao, Fan Zhang, Fang Liu, Shuyuan Yang, Lingling Li, Zhixi Feng, and Rong Qu. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[KLL16a] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Accurate image super-resolution using very deep convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1646–1654, 2016.

[KLL16b] Jiwon Kim, Jung Kwon Lee, and Kyoung Mu Lee. Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1637–1645, 2016.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, 25:1097–1105, 2012.

[LAE+16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European Conference on Computer Vision*, pages 21–37. Springer, 2016.

[LBH15] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[LDG+17] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[LGG+17] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

[LHAY17] Wei-Sheng Lai, Jia-Bin Huang, Narendra Ahuja, and Ming-Hsuan Yang. Deep laplacian pyramid networks for fast and accurate super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 624–632, 2017.

[LK16] Lukas Liebel and Marco 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*, 41:883–890, 2016.

[LKM+18] Darius Lam, Richard Kuzma, Kevin McGee, Samuel Dooley, Michael Laielli, Matthew Klaric, Yaroslav Bulatov, and Brendan McCord. xview: Objects in context in overhead imagery. *arXiv preprint arXiv:1802.07856*, 2018.

[LM15] Kang Liu and Gellert Mattyus. Fast multiclass vehicle detection on aerial images. *IEEE Geoscience and Remote Sensing Letters*, 12(9):1938–1942, 2015.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European Conference on Computer Vision*, pages 740–755. Springer, 2014.

[LOW+20] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International Journal of Computer Vision*, 128(2):261–318, 2020.

[LRK+14] Jens Leitloff, Dominik Rosenbaum, Franz Kurz, Oliver Meynberg, and Peter Reinartz. An operational system for estimating road traffic information from aerial images. *Remote Sensing*, 6(11):11315–11341, 2014.

[LSK+17] Bee Lim, Sanghyun Son, Heewon Kim, Seungjun Nah, and Kyoung Mu Lee. Enhanced deep residual networks for single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 136–144, 2017.

[LTH+17] Christian Ledig, Lucas Theis, Ferenc Huszár, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, et al. Photo-realistic single image super-resolution using a generative adversarial network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4681–4690, 2017.

[LWZ+19] Xiaofang Li, Yirui Wu, Wen Zhang, Ruichao Wang, and Feng Hou. Deep learning methods in real-time image super-resolution: A survey. *Journal of Real-Time Image Processing*, pages 1–25, 2019.

[LYL18] Xiaofei Liu, Tao Yang, and Jing Li. Real-time ground vehicle detection in aerial infrared imagery based on convolutional neural network. *Electronics*, 7(6):78, 2018.

[MA18] Seyed Majid Azimi. Shuffledet: Real-time vehicle detection network in on-board embedded uav imagery. In *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, pages 0–0, 2018.

[MFRN20] Moktari Mostofa, Syeda Nyma Ferdous, Benjamin S Riggan, and Nasser M Nasrabadi. Joint-srvdnet: Joint super resolution and vehicle detection network. *IEEE Access*, 8:82306–82319, 2020.

[MKSB16] T Nathan Mundhenk, Goran Konjevod, Wesam A Sakla, and Kofi Boakye. A large contextual dataset for classification, detection and counting of cars with deep learning. In *European Conference on Computer Vision*, pages 785–800. Springer, 2016.

[Mor11] Nelson Morgan. Deep and wide: Multiple layers in automatic speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):7–13, 2011.

[NHKML17] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep multi-scale convolutional neural network for dynamic scene deblurring. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3883–3891, 2017.

[Nie15] Michael A Nielsen. *Neural Networks and Deep Learning*. Determination Press San Francisco, CA, USA, 2015.

[NM14] Kamal Nasrollahi and Thomas B Moeslund. Super-resolution: A comprehensive survey. *Machine Vision and Applications*, 25(6):1423–1468, 2014.

[RDGF16] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 779–788, 2016.

[RDS+15] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[RF17] Joseph Redmon and Ali Farhadi. Yolo9000: Vetter, faster, stronger. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7263–7271, 2017.

[RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39, 06 2015.

[RJ16] Sebastien Razakarivony and Frederic Jurie. Vehicle detection in aerial imagery: A small target detection benchmark. *Journal of Visual Communication and Image Representation*, 34:187–203, 2016.

[SCH+16] Wenzhe Shi, Jose Caballero, Ferenc Huszár, Johannes Totz, Andrew P Aitken, Rob Bishop, Daniel Rueckert, and Zehan Wang. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1874–1883, 2016.

[SIVA17] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI Conference on Artificial Intelligence*, 2017.

[SKM17] Wesam Sakla, Goran Konjevod, and T Nathan Mundhenk. Deep multimodal vehicle detection in arial isr imagery. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 916–923. IEEE, 2017.

[SLJ+15] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.

[SSB17] Lars Wilko Sommer, Tobias Schuchert, and Jürgen Beyerer. Fast deep vehicle detection in aerial images. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 311–319. IEEE, 2017.

[SSSB18] Lars Sommer, Arne Schumann, Tobias Schuchert, and Jürgen Beyerer. Multi feature deconvolutional faster r-cnn for precise vehicle detection in aerial imagery. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 635–642. IEEE, 2018.

[SSSG17] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 843–852, 2017.

[SVE19] Jacob Shermeyer and Adam Van Etten. The effects of super-resolution on object detection performance in satellite imagery. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1432–1441. IEEE, 2019.

[SWG+18] Jun Sang, Zhongyuan Wu, Pei Guo, Haibo Hu, Hong Xiang, Qian Zhang, and Bin Cai. An improved yolov2 for vehicle detection. *Sensors*, 18(12):4272, 2018.

[SZ15] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015.

[TCP+09] Franklin Tanner, Brian Colder, Craig Pullen, David Heagy, Michael Eppolito, Veronica Carlan, Carsten Oertel, and Phil Sallee. Overhead imagery research data set – an annotated data library & tools to aid in the development of computer vision algorithms. In *2009 IEEE Applied Imagery Pattern Recognition Workshop (AIPR 2009)*, pages 1–8. IEEE, 2009.

[TYL17] Ying Tai, Jian Yang, and Xiaoming Liu. Image super-resolution via deep recursive residual network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3147–3155, 2017.

[TZD+17a] Tianyu Tang, Shilin Zhou, Zhipeng Deng, Lin Lei, and Huanxin Zou. Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks. *Remote Sensing*, 9(11):1170, 2017.

[TZD+17b] Tianyu Tang, Shilin Zhou, Zhipeng Deng, Huanxin Zou, and Lin Lei. Vehicle detection in aerial images based on region convolutional neural networks and hard negative example mining. *Sensors*, 17(2):336, 2017.

[UVDSGS13] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2):154–171, 2013.

[van06] J.D. van Ouwerkerk. Image super-resolution survey. *Image and Vision Computing*, 24(10):1039–1052, 2006.

[WCH20] Zhihao Wang, Jian Chen, and Steven CH Hoi. Deep learning for image super-resolution: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[WGGH18] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7794–7803, 2018.

[WLH+20] Xin Wu, Wei Li, Danfeng Hong, Jiaojiao Tian, Ran Tao, and Qian Du. Vehicle detection of multi-source remote sensing data using active fine-tuning network. *ISPRS Journal of Photogrammetry and Remote Sensing*, 167:39–53, 2020.

[WSH20] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.

[WYW+18] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution generative adversarial networks. In *European Conference on Computer Vision*, pages 63–79. Springer, 2018.

[XBD+18] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3974–3983, 2018.

[YWHM10] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE Transactions on Image Processing*, 19(11):2861–2873, 2010.

[YZT+19] Wenming Yang, Xuechen Zhang, Yapeng Tian, Wei Wang, Jing-Hao Xue, and Qingmin Liao. Deep learning for single image super-resolution: A brief review. *IEEE Transactions on Multimedia*, 21(12):3106–3121, 2019.

[ZF14] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conference on Computer Vision*, pages 818–833. Springer, 2014.

[ZJH19] Junpeng Zhang, Xiuping Jia, and Jiankun Hu. Error bounded foreground and background modeling for moving object detection in satellite videos. *IEEE Transactions on Geoscience and Remote Sensing*, 58(4):2659–2669, 2019.

[ZLL+18] Yulun Zhang, Kunpeng Li, Kai Li, Lichen Wang, Bineng Zhong, and Yun Fu. Image super-resolution using very deep residual channel attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 286–301, 2018.

[ZLP+20] Sebastian Zambanini, Ana-Maria Loghin, Norbert Pfeifer, Elena Màrmol Soley, and Robert Sablatnig. Detection of parking cars in stereo satellite images. *Remote Sensing*, 12(13):2170, 2020.

[ZTK+18] Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. Residual dense network for image super-resolution. In *Proceedings of the IEEE*

Conference on Computer Vision and Pattern Recognition, pages 2472–2481, 2018.

[ZZW+13] Zezhong Zheng, Guoqing Zhou, Yong Wang, Yalan Liu, Xiaowen Li, Xiaoting Wang, and Ling Jiang. A novel vehicle detection method with high resolution highway aerial image. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 6(6):2338–2343, 2013.

[ZZXW19] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.